This book presents the theory of output-driven maps and provides a fresh perspective on the extent to which phonologies can be characterized in terms of restrictions on outputs. Closely related to traditional conceptions of process opacity, but differing in notable ways, the theory of output-driven maps applies equally to SPE-style ordered rules, Optimality Theory, and other phonological theories. It permits a formally rigorous analysis of the issues in Optimality Theory that is not possible with traditional process opacity.

Also presented is a theory of phonological learning. Building on prior work on learning in Optimality Theory, the learning theory exploits the formal structure of output-driven maps to achieve learning that is far more computationally efficient than comparable prior approaches.

In this book Bruce Tesar, one of the founders of the study of learnability in Optimality Theory, presents fresh perspectives in an accessible way for graduate students and academic researchers.

BRUCE TESAR is Associate Professor in the Department of Linguistics and the Center for Cognitive Science at Rutgers University, New Brunswick.
Output-Driven Phonology: Theory and Learning
In this series

106 Sharon Inkelas and Cheryl Zoll: Reduplication: doubling in morphology
107 Susan Edwards: Fluent aphasia
108 Barbara Dancygier and Eve Sweetser: Mental spaces in grammar: conditional constructions
110 Marcus Tomalin: Linguistics and the formal sciences: the origins of generative grammar
111 Samuel D. Epstein and T. Daniel Seely: Derivations in minimalism
112 Paul de Lacy: Markedness: reduction and preservation in phonology
113 Yehuda N. Falk: Subjects and their properties
114 P. H. Matthews: Syntactic relations: a critical survey
115 Mark C. Baker: The syntax of agreement and concord
116 Gillian Catriona Ramchand: Verb meaning and the lexicon: a first phase syntax
117 Pieter Muysken: Functional categories
118 Juan Uriagereka: Syntactic anchors: on semantic structuring
119 D. Robert Ladd: Intonational phonology second edition
120 Leonard H. Babby: The syntax of argument structure
121 B. Elan Dresher: The contrastive hierarchy in phonology
122 David Adger, Daniel Harbour and Laurel J. Watkins: Mirrors and microparameters: phrase structure beyond free word order
123 Niina Ning Zhang: Coordination in syntax
124 Neil Smith: Acquiring phonology
125 Nina Topintzi: Onsets: suprasegmental and prosodic behaviour
126 Cedric Boeckx, Norbert Hornstein and Jairo Nunes: Control as movement
127 Michael Israel: The grammar of polarity: pragmatics, sensitivity, and the logic of scales
128 M. Rita Manzini and Leonardo M. Savoia: Grammatical categories: variation in romance languages
129 Barbara Citko: Symmetry in syntax: merge, move and labels
130 Rachel Walker: Vowel patterns in language
131 Mary Dalrymple and Irina Nikolaeva: Objects and information structure
132 Jerrold M. Sadock: The modular architecture of grammar
133 Dunstan Brown and Andrew Hippisley: Network morphology: a defaults-based theory of word structure
134 Bettelou Los, Corrien Blom, Geert Booij, Marion Elenbaas and Ans van Kemenade: Morphosyntactic change: a comparative study of particles and prefixes
135 Stephen Crain: The Emergence of Meaning
136 Hubert Haider: Symmetry Breaking in Syntax
137 José A. Camacho: Null Subjects
138 Gregory Stump and Raphael A. Finkel: Morphological Typology: From Word to Paradigm
139 Bruce Tesar: Output-Driven Phonology: Theory and Learning

Earlier issues not listed are also available
OUTPUT-DRIVEN PHONOLOGY

THEORY AND LEARNING

BRUCE TESAR

Rutgers University, New Brunswick.

Cambridge University Press
This book is dedicated to Heidi and Amanda, for all they have done to inspire my work and my life.

“Who do we ask for help when we don’t know which way to go? The map!”

Dora the Explorer
# Contents

List of figures  page xvii  
Acknowledgments  page xix

1 Characterizing surface orientedness in phonology  1  
1.1 Surface orientedness  1  
1.1.1 Surface orientedness in phonology  1  
1.1.2 Formalizing surface orientedness  3  
1.2 Surface orientedness in Optimality Theory  6  
1.2.1 Markedness violations as opacity  7  
1.2.2 Markedness constraints cause disparities  10  
1.2.3 Something more general  11  
1.3 Formalizing surface orientedness: output-driven maps  13  
1.4 Output drivenness and Optimality Theory  15  
1.5 Output drivenness and learning  16  
1.6 The relationship between learnability and linguistic theory  18

2 Output-driven maps  20  
2.1 The main idea  20  
2.1.1 Terminology: candidates and correspondence  20  
2.1.2 Inputs of greater similarity yield the same output  21  
2.1.3 Unifying surface orientedness  23  
2.2 Relative similarity  26  
2.2.1 Relating the disparities of two candidates  26  
2.2.2 Individuating disparities  28  
2.2.3 Relative similarity is a relational notion  30  
2.2.4 The importance of input–output correspondence  33  
2.3 Output drivenness is not process opacity  37  
2.3.1 One map, multiple generalizations  37  
2.3.2 Epenthesis and assimilation in Lithuanian  39  
2.3.3 Closeness with processes  42  
2.4 Formal analysis with segmental IO correspondence  44
2.4.1 Maps from inputs to candidates 44
2.4.2 The internal structure of candidates 46
2.4.3 Relating candidates to each other 48
2.4.4 The non-uniqueness of input–input correspondence 53
2.4.5 Removing disparities by changing the input 54
2.4.6 The identical disparity requirement and surface orientedness 55
2.4.7 Individuating disparities (again) 57
2.5 Expanding to other representational theories 59
2.5.1 Non-identical corresponding representational elements 59
2.5.2 Non-unique correspondence 60
2.5.3 Autosegmental representation 60
2.6 The map 61

3 Output-driven maps in Optimality Theory 63
3.1 Background: ERC entailment in Optimality Theory 64
3.1.1 Elementary ranking conditions 64
3.1.2 Single ERC entailment: L-retraction and W-extension 66
3.1.3 Joint ERC entailment: fusion 67
3.2 Relating output-driven maps to Optimality Theory 69
3.2.1 Output-driven maps and optimization 69
3.2.2 A designated competitor: aoy 70
3.2.3 Relationships among the disparities 74
3.2.4 As goes bpy, so goes aoy 78
3.2.5 Output-driven maps and constraints 81
3.3 Sufficient conditions for output-driven maps 83
3.3.1 Properties of GEN: correspondence uniformity 84
3.3.2 Properties of constraints: output-driven preserving 86
3.3.3 Proof of sufficient conditions for output-driven maps 87
3.4 Basic constraints: overview of the results 88
3.4.1 Terminology: faithfulness and input-referring constraints 88
3.4.2 Markedness constraints 89
3.4.3 Value-independent input-referring constraints 89
3.4.4 Value-restricted input-referring constraints 90
3.5 Analysis of relationships between disparities 91
3.5.1 The set-up 91
3.5.2 Deletion disparities 93
3.5.3 Insertion disparities 94
3.5.4 Identity disparities 96
3.5.5 Comments/discussion 101
3.6 Output-driven preserving constraints: the proofs 101
3.6.1 Outline of the proof structure 101
3.6.2 MAX 102
3.6.2.1 Partition of the deletion disparities 102
3.6.2.2 Corresponding deletion disparities for \( \text{aoy} \) and \( \text{bpy} \) 103
3.6.2.3 Corresponding deletion disparities for \( \text{akx} \) and \( \text{bmx} \) 103
3.6.2.4 Non-corresponding deletion disparities for \( \text{aoy} \) and \( \text{akx} \) 104
3.6.2.5 \( \text{Max}(\text{bpy}) < \text{Max}(\text{bmx}) \) entails \( \text{Max}(\text{aoy}) < \text{Max}(\text{akx}) \) 104
3.6.2.6 \( \text{Max}(\text{bpy}) = \text{Max}(\text{bmx}) \) entails \( \text{Max}(\text{aoy}) \leq \text{Max}(\text{akx}) \) 105

3.6.3 \( \text{Dep} \)
3.6.3.1 Partition of the insertion disparities 105
3.6.3.2 Corresponding insertion disparities for \( \text{aoy} \) and \( \text{bpy} \) 106
3.6.3.3 Corresponding insertion disparities for \( \text{akx} \) and \( \text{bmx} \) 106
3.6.3.4 Non-corresponding insertion disparities for \( \text{aoy} \) and \( \text{akx} \) 106
3.6.3.5 Proof: \( \text{Dep}(\text{bpy}) < \text{Dep}(\text{bmx}) \) entails \( \text{Dep}(\text{aoy}) < \text{Dep}(\text{akx}) \) 107
3.6.3.6 Proof: \( \text{Dep}(\text{bpy}) = \text{Dep}(\text{bmx}) \) entails \( \text{Dep}(\text{aoy}) \leq \text{Dep}(\text{akx}) \) 107

3.6.4 \( \text{Ident}[\text{F}_\text{in} \in V] \)
3.6.4.1 Partition of the identity disparities 108
3.6.4.2 Identical corresponding identity disparities for \( \text{aoy} \) and \( \text{bpy} \) 109
3.6.4.3 Corresponding identity disparities for \( \text{akx} \) and \( \text{bmx} \) 110
3.6.4.4 Non-Corresponding and non-identical corresponding identity disparities for \( \text{aoy} \) and \( \text{akx} \) 110
3.6.4.5 Proof: \( \text{Ident}[\text{F}_\text{in} \in V](\text{bpy}) < \text{Ident}[\text{F}_\text{in} \in V](\text{bmx}) \) entails \( \text{Ident}[\text{F}_\text{in} \in V](\text{aoy}) < \text{Ident}[\text{F}_\text{in} \in V](\text{akx}) \) 110
3.6.4.6 Proof: \( \text{Ident}[\text{F}_\text{in} \in V](\text{bpy}) = \text{Ident}[\text{F}_\text{in} \in V](\text{bmx}) \) entails \( \text{Ident}[\text{F}_\text{in} \in V](\text{aoy}) \leq \text{Ident}[\text{F}_\text{in} \in V](\text{akx}) \) 111

3.6.5 \( \text{Ident}[\text{F}_\text{out} \in V] \)
3.6.5.1 Partition of the identity disparities 112
3.6.5.2 Corresponding identity disparities for \( \text{aoy} \) and \( \text{bpy} \) 113
3.6.5.3 Corresponding identity disparities for \( \text{akx} \) and \( \text{bmx} \) 114
3.6.5.4 Non-corresponding identity disparities for \( \text{aoy} \) and \( \text{akx} \) 114
3.6.5.5 Proof: \( \text{Ident}[\text{F}_\text{out} \in V](\text{bpy}) < \text{Ident}[\text{F}_\text{out} \in V](\text{bmx}) \) entails \( \text{Ident}[\text{F}_\text{out} \in V](\text{aoy}) < \text{Ident}[\text{F}_\text{out} \in V](\text{akx}) \) 114
3.6.5.6 Proof: \( \text{Ident}[\text{F}_\text{out} \in V](\text{bpy}) = \text{Ident}[\text{F}_\text{out} \in V](\text{bmx}) \) entails \( \text{Ident}[\text{F}_\text{out} \in V](\text{aoy}) \leq \text{Ident}[\text{F}_\text{out} \in V](\text{akx}) \) 115

3.7 The map 116
4 Analysis of constraint behavior

4.1 Non-ODP constraints and non-output-driven maps 117
4.2 Illustrating the three ways of non-ODP
4.2.1 Distinction only at lesser similarity 119
4.2.2 Distinction only at greater similarity 122
4.2.3 Distinction conflict 125
4.3 Relating non-ODP constraint behaviors to non-ODM map patterns 127
4.4 Faithfulness conditioned on output context
4.4.1 Positional faithfulness I: position-specific Dep 129
4.4.2 Positional faithfulness II: position-specific Ident[F]\square 132
4.4.3 Conjoined markedness and faithfulness 134
4.4.4 Value-restricted Dep 137
4.4.5 Summary: independent context 138
4.5 Faithfulness conditioned on input context
4.5.1 Value-restricted Max 139
4.5.2 Relation to output-conditioned faithfulness 142
4.6 Multiply conditioned faithfulness
4.6.1 Joint input–output value restrictions 143
4.6.2 Conditioning on disparities 144
4.7 Conditioned antifaitfulness
4.7.1 Material implication constraints 145
4.7.2 F \rightarrow M as output-conditioned antifaitfulness 146
4.8 Reference to other forms: sympathy 148
4.9 Eventual idempotency
4.9.1 Classical OT grammars are eventually idempotent 151
4.9.2 Absolute vs. relative satisfaction of constraints 154
4.9.3 Derived environment exchanges 155
4.9.4 Absolute vs. relative characterizations of similarity 159
4.10 The role of relative similarity
4.10.1 Distinguishing faithfulness from faithfulness constraints 161
4.10.2 The rotation theorem isn’t about faithfulness 162
4.10.3 Relative similarity links grammars to maps 166
4.11 More on relating non-ODP behaviors to non-ODM patterns 167
4.12 Maps and grammars 170
4.13 The map 171

5 Learning phonotactics

5.1 An overview of the learning problem 174
5.2 Stress and length: a system for illustration 175
5.3 Constructing a constraint hierarchy from winner–loser pairs
5.3.1 Recursive Constraint Demotion 177
5.3.2 Stratified constraint hierarchies 181
5.3.3 Constraint conflict and ranking relations 184
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.4</td>
<td>Selecting winner–loser pairs</td>
<td>185</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Error detection</td>
<td>186</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Production-directed parsing with stratified hierarchies</td>
<td>187</td>
</tr>
<tr>
<td>5.4.2.1</td>
<td>Mark pooling</td>
<td>188</td>
</tr>
<tr>
<td>5.4.2.2</td>
<td>Conflicts Tie</td>
<td>191</td>
</tr>
<tr>
<td>5.4.2.3</td>
<td>Variationist EDCD</td>
<td>194</td>
</tr>
<tr>
<td>5.4.3</td>
<td>MultiRecursive Constraint Demotion (MRCD)</td>
<td>195</td>
</tr>
<tr>
<td>5.4.4</td>
<td>MRCD step by step</td>
<td>197</td>
</tr>
<tr>
<td>5.4.4.1</td>
<td>First pair</td>
<td>198</td>
</tr>
<tr>
<td>5.4.4.2</td>
<td>Second pair</td>
<td>198</td>
</tr>
<tr>
<td>5.4.4.3</td>
<td>Third pair</td>
<td>198</td>
</tr>
<tr>
<td>5.4.4.4</td>
<td>Fourth pair</td>
<td>199</td>
</tr>
<tr>
<td>5.4.4.5</td>
<td>Fifth pair</td>
<td>199</td>
</tr>
<tr>
<td>5.4.4.6</td>
<td>Last pair</td>
<td>200</td>
</tr>
<tr>
<td>5.4.5</td>
<td>Limitations of loser production via stratified hierarchies</td>
<td>200</td>
</tr>
<tr>
<td>5.5</td>
<td>Assessing computational requirements</td>
<td>203</td>
</tr>
<tr>
<td>5.5.1</td>
<td>The computational complexity of MRCD</td>
<td>203</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Grammar space vs. language space</td>
<td>205</td>
</tr>
<tr>
<td>5.6</td>
<td>Restrictiveness biases</td>
<td>207</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Restrictiveness in learning</td>
<td>207</td>
</tr>
<tr>
<td>5.6.2</td>
<td>Phonotactic learning</td>
<td>209</td>
</tr>
<tr>
<td>5.6.3</td>
<td>Language subsets vs. grammar subsets</td>
<td>211</td>
</tr>
<tr>
<td>5.6.4</td>
<td>Biased Constraint Demotion</td>
<td>213</td>
</tr>
<tr>
<td>5.6.4.1</td>
<td>RCD and restrictiveness</td>
<td>213</td>
</tr>
<tr>
<td>5.6.4.2</td>
<td>Estimating the restrictiveness of grammars: the r-measure</td>
<td>215</td>
</tr>
<tr>
<td>5.6.4.3</td>
<td>A restrictiveness bias for RCD</td>
<td>216</td>
</tr>
<tr>
<td>5.6.5</td>
<td>Enforcing restrictiveness in phonotactic learning</td>
<td>220</td>
</tr>
<tr>
<td>5.6.5.1</td>
<td>Enforcing restrictiveness with BCD</td>
<td>220</td>
</tr>
<tr>
<td>5.6.5.2</td>
<td>Restrictiveness with “hierarchy only” learning</td>
<td>221</td>
</tr>
<tr>
<td>5.6.6</td>
<td>Implicit representation of phonotactic restrictions</td>
<td>222</td>
</tr>
<tr>
<td>5.7</td>
<td>Phonotactic contrast</td>
<td>224</td>
</tr>
<tr>
<td>5.7.1</td>
<td>Contrast and the nature of phonotactic learning</td>
<td>224</td>
</tr>
<tr>
<td>5.7.2</td>
<td>A canonical form for phonotactic contrast</td>
<td>226</td>
</tr>
<tr>
<td>5.8</td>
<td>Phonotactic information underdetermines languages</td>
<td>233</td>
</tr>
<tr>
<td>5.9</td>
<td>The map</td>
<td>236</td>
</tr>
<tr>
<td>5.10</td>
<td>Appendix: the Stress/Length typology</td>
<td>237</td>
</tr>
</tbody>
</table>

## 6 Learning with paradigmatic information  

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Paradigmatic information</td>
<td>247</td>
</tr>
<tr>
<td>6.2</td>
<td>The explosive growth of lexical hypothesis spaces</td>
<td>248</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Now that’s big</td>
<td>248</td>
</tr>
<tr>
<td>6.2.2</td>
<td>The basic alternant constraint</td>
<td>248</td>
</tr>
</tbody>
</table>
## 6

6.2.3 Selected prior work 250
6.2.4 Combinatorics of the Stress/Length linguistic system 252
6.3 An aside on methodology 254
6.4 Inconsistency detection 256
6.5 Setting underlying features via inconsistency detection 258
  6.5.1 Feature setting 259
  6.5.2 Setting a single unset feature 260
  6.5.3 Multiple unset features 263
  6.5.4 Multiple words and local lexica 266
6.6 Non-phonotactic ranking information 269
  6.6.1 Ranking information in local lexica 269
  6.6.2 The join operation 270
  6.6.3 Extracting shared ranking information 272
6.7 The Contrast Pair and Ranking (CPR) algorithm 274
6.8 Computational issues for CPR 276
6.9 The map 277

## 7

### 7.1 Contrast with Richness of the Base
  7.1.1 Contrastive for an input 280
  7.1.2 Contrastive for a morpheme 282
  7.1.3 Contrast in output-driven maps 283

### 7.2 Relative similarity lattices 284

### 7.3 Limiting lexical search in output-driven maps 287

### 7.4 Phonotactic contrast and underlying feature values 291

### 7.5 Morphemic alternation and non-phonotactic ranking information 294

### 7.6 Contrast pairs
  7.6.1 When one word isn’t enough 299
  7.6.2 A disjunction of disparities 302
  7.6.3 Restricted rankings create inconsistencies 305
  7.6.4 The roles of alternation and contrast in contrast pairs 308
  7.6.5 Multiple words and relative similarity 310
  7.6.6 Another illustration: setting the stress feature for r1 316
  7.6.7 Setting environment morpheme features 317

### 7.7 Beyond error-driven learning
  7.7.1 Uncertainty in the lexicon 323
  7.7.2 Uncertainty about the ranking 328
  7.7.3 Single form learning 332
  7.7.4 Contrast pair learning 334

### 7.8 The Output-Driven Learner (preliminary) 335

### 7.9 Learning language L20
  7.9.1 Phonotactic learning 336
  7.9.2 Initial single form learning 338
8 Paradigmatic subsets

8.1 The phenomenon: paradigmatic subsets
8.1.1 Language L8, the subset language
8.1.2 Language L7, the superset language
8.1.3 L8 is a paradigmatic subset of L7
8.2 The problem: attempting to learn L8
8.3 Paradigmatic restrictiveness and the lexicon
8.4 The solution: Fewest Set Features
8.4.1 The Fewest Set Features procedure
8.4.2 Algorithmic details of Fewest Set Features
8.4.3 The relation to maximum likelihood
8.4.4 Summary: restrictiveness in the lexicon
8.5 Evaluating the Output-Driven Learner
8.5.1 The Output-Driven Learner (revised)
8.5.2 Simulation results
8.5.2.1 No contrast languages
8.5.2.2 Single form learning, no contrast pairs
8.5.2.3 Contrast pairs
8.5.2.4 Fewest Set Features
8.5.2.5 A little perspective
8.5.3 Issues
8.6 The map

9 Linguistic theory and language learnability

9.1 Contrast and the final lexicon
9.2 Forms of restrictiveness enforcement
9.3 Evaluation metrics
9.4 What is stored by the learner (and why)
9.4.1 Ranking information
9.4.2 Lexical information
xvi  Contents

9.4.3  Structural ambiguity and multiple hypotheses  396
9.5   Beyond output drivenness in learning  398
9.6   What has been accomplished  399

References  401
Index  410
## Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Relative similarity relation (upward is greater internal similarity).</td>
<td>27</td>
</tr>
<tr>
<td>2.2</td>
<td>Candidates $akx$ and $bmx$, and their input–input correspondence.</td>
<td>51</td>
</tr>
<tr>
<td>3.1</td>
<td>Candidates $akx$, $bmx$, $aoy$, and $bpy$.</td>
<td>73</td>
</tr>
<tr>
<td>3.2</td>
<td>Relationships among the disparities. Corresponding disparities (same output) are connected with double-sided arrows; analogous disparities (same input) are connected by the square-angled lines.</td>
<td>75</td>
</tr>
<tr>
<td>3.3</td>
<td>Candidates $akx$, $bmx$, $aoy$, and $bpy$.</td>
<td>80</td>
</tr>
<tr>
<td>3.4</td>
<td>Relationships among the disparities when $ouy = [tebig]$.</td>
<td>81</td>
</tr>
<tr>
<td>7.1</td>
<td>Relative similarity lattice for output $[paká:]$ (higher in the diagram means greater internal similarity).</td>
<td>285</td>
</tr>
<tr>
<td>7.2</td>
<td>Relative similarity lattice for output $paká$: (feature matrix version).</td>
<td>286</td>
</tr>
<tr>
<td>7.3</td>
<td>Setting $s4$ to +long. This is the relative similarity lattice for the output of $r1s4$, $[paká:]$. The shaded sublattice contains all candidates with $s4$ underlyingly –long.</td>
<td>289</td>
</tr>
<tr>
<td>7.4</td>
<td>The relative similarity lattice for $r3s4$ (nonviable candidates, with $s4$ underlyingly –long, are marked with shaded diamonds).</td>
<td>296</td>
</tr>
<tr>
<td>7.5</td>
<td>The relative similarity lattice for $r1s1$. The shaded diamond nodes have $r1$ underlyingly +long, counter to what has already been set in the lexicon, and so are not viable candidates.</td>
<td>311</td>
</tr>
<tr>
<td>7.6</td>
<td>The relative similarity lattice for $r1s3$. The shaded diamond nodes have $r1$ underlyingly +long and/or $s3$ underlyingly +long, counter to what has already been set in the lexicon, and so are not viable candidates.</td>
<td>311</td>
</tr>
<tr>
<td>7.7</td>
<td>Relative similarity lattice for $r1s1$ (viable candidates only).</td>
<td>312</td>
</tr>
</tbody>
</table>
Figures

7.8  Relative similarity lattice for r1s3 (viable candidates only). 312
7.9  The joint relative similarity order for the contrast pair r1s1 and r1s3. Each node is labeled with underlying forms for r1s1 and r1s3, in that order. The left suborder contains all local lexica in which r1 is –stress underlyingly, while the right suborder contains all local lexica in which r1 is +stress underlyingly. 313
7.10 Testing the stress feature for s1. The shaded nodes are the local lexica with +stress assigned to s1 underlyingly. 315
8.1  Relative similarity lattice for r1s1, where s1 has been set to –long. 366
8.2  Viable sublattice for r1s1. 367
8.3  Viable sublattice for r1s1 with inconsistent inputs shaded. 368
Acknowledgments

The work in this book was conducted over a period of several years. During that time, I have benefitted significantly from conversations with a number of people, including Crystal Akers, Eric Baković, Karen Campbell, Paul de Lacy, Jane Grimshaw, Fernando Guzman, Brett Hyde, Gaja Jarosz, John McCarthy, Nazarré Merchant, Alan Prince, Jason Riggle, and Paul Smolensky. I also received useful feedback from several audiences who kindly listened to various incarnations of pieces of this work: the Rutgers Optimality Reading Group (RORG), the Mathematics Association of America Seaway Section and New York Association of Two Year Colleges Fall 2007 Meeting, several meetings of the Northeast Computational Phonology Circle, the Learnability Meets Acquisition Workshop at the 31st Jahrestagung der Deutsche Gesellschaft für Sprachwissenschaft in Osnabrück, NELS39, the Seventh North American Phonology Conference, the Linguistics Department at SUNY Stony Brook, and the Linguistics and Cognitive Science Department at the University of Delaware. Useful comments were also provided by three Cambridge University Press reviewers. Exceptional patience was exhibited by the students of two incarnations of my Seminar on Learnability and Linguistic Theory (Ling670), as well one incarnation of Phonology III (Ling522).

All factual errors and bizarre-sounding conceptions are the sole responsibility of the author.

Special thanks are due to Alan Prince for suggesting the term “output-driven.”

I wish to thank, for institutional support, the Department of Linguistics and the Rutgers Center for Cognitive Science (RuCCS), at Rutgers, the State University of New Jersey, New Brunswick.

I would like to thank Helen Barton, of Cambridge University Press, for her support of this project and her assistance in getting this work into print.

I wish to express gratitude to my wife, Esther, and my daughters, Heidi and Amanda, for their support and exceptional tolerance of the countless hours I spent in my basement office, working on “the book.” Special thanks to Heidi and Amanda for loving me despite the fact that I do not like ketchup.
1 Characterizing surface orientedness in phonology

1.1 Surface orientedness

1.1.1 Surface orientedness in phonology

A long-standing issue of interest in phonology is the extent to which phonological systems can be characterized in terms of conditions on the surface forms (Chomsky 1964, Kiparsky 1971, Kiparsky 1973, Kisseberth 1970). One way to express the issue is in terms of disparities between the (underlying) input representation and the (surface) output representation. If one supposes that the input will be preserved in the output “by default,” to what extent are the disparities that are introduced motivated by restrictions on the output form? I will use the term surface orientedness to refer to the intuitive property of all disparities being motivated by output restrictions. This is expressed slightly differently in (1.1), which asserts that disparities are introduced only to the extent necessary to satisfy output restrictions.

(1.1) Surface orientedness: disparities between input and output are only introduced to the extent necessary to satisfy output restrictions.

Many if not most phonological phenomena seem to be characterizable in terms of output restrictions. Assimilation acts to ensure that, in the output, adjacent segments of certain kinds (e.g., obstruents) are alike in certain ways (e.g., agree in voicing). Positional neutralization acts to ensure that, in the output, segments of certain kinds (e.g., non-glottal obstruents) do not appear in the neutralized positions (e.g., syllable codas). Minimal-size phenomena act to ensure that, in the output, constituents of certain kinds (e.g., prosodic words) are never smaller than a certain size (e.g., a properly formed metrical foot).

The output restrictions themselves are only part of the story. There is also the matter of which disparities serve to enforce the output restrictions. Nearly any potential violation of an output restriction can be resolved in more than one way, with different ways involving the introduction of different disparities. If

1 For a broader overview of the issue, see McCarthy 2007a.
an input form has adjacent obstruents that differ in voicing, an output restriction requiring adjacent obstruents to agree in voicing can be satisfied by voicing the underlyingly voiceless obstruent, or by devoicing the underlyingly voiced obstruent. If an input form has a labial consonant that is at risk of being syllabified as a coda, an output restriction forbidding labials in coda position can be satisfied by deletion of the consonant (so that it does not appear at all in the output), by changing the place of the consonant to something that is permitted in coda position (e.g., glottal place), or by using an alternative syllabification so that the labial is not in a coda position in the output (e.g., by epenthesizing a vowel immediately after the labial consonant).

Further, the choice of disparities for enforcing output restrictions is not just a matter of cross-linguistic variation. One of the more powerful arguments made for the value of output restrictions in understanding phonology is the existence of “conspiracy” patterns, where different disparities are introduced in different contexts, all to the effect of enforcing the same output restriction (Kisseberth 1970). The use of different strategies in different contexts attests to the presence of multiple output restrictions within a language, so that the preferred disparity for enforcing an output restriction might be abandoned in favor of another one if, in that context, the preferred disparity would run afoul of another output restriction.

An example can be found in Prince and Smolensky’s (1993/2004) analysis of Lardil. The presentation here is greatly simplified for expositional purposes. In Lardil, there is a restriction on segments in codas, which Prince and Smolensky label the “Coda Condition”: “A coda consonant can have only Coronal place or place shared with another consonant.” This is an output restriction, and it is in some cases enforced by the introduction of a deletion disparity. This can be seen in nominative case words. The nominative case itself has no overt phonological realization, so the underlying stem receives no further inflection. In longer stems that end in a consonant, that final consonant is deleted in the output if it would violate the Coda Condition when syllabified as a coda unaltered, as shown in (1.2).

(1.2) Truncation of nominative case forms in Lardil
/naluk/ → [nalu]
/wunjunuŋ/ → [unjunuŋ]
/wanjalk/ → [wanjal]

Deletion is not the only imaginable way of enforcing the Coda Condition. Epenthesizing a vowel at the end of the word and creating a new syllable would

also avoid violation of the Coda Condition, as would altering the final consonant to one with coronal place. But in these contexts, Lardil clearly prefers to delete the final consonant.

The situation is rather different in some other contexts. Another distributional observation in Lardil is the Minimal Word Condition: lexical words (such as nouns) always appear as at least bimoraic. This output restriction can be related to the output of short stems ending in a potentially offending consonant: instead of deleting the final consonant, a vowel is epenthesized following the consonant, and the two segments form a separate, final syllable in the output, as shown in (1.3).

(1.3) Augmentation of nominative case forms in Lardil
/yak/ → [yaka]
/relk/ → [relka]

Clearly, deletion and insertion are distinct sorts of disparities. Yet, in Lardil, both serve to enforce the Coda Condition and the Minimal Word Condition. This is a conspiracy of the Kisseberthian sort: deletion and insertion “conspire” to enforce the output restrictions.

It is imaginable that both kinds of disparities would be introduced for subminimal stems such as in (1.3), with the underlying final consonant being deleted, and then both a consonant and a vowel being inserted to form a second syllable. But that is not what happens. Abandoning the use of deletion in this instance allows the introduction of a minimal amount of insertion (only one segment, the output-final vowel) while simultaneously satisfying both output restrictions: the word is bimoraic in the output and there are no non-coronal codas in the output. Given that some insertion is unavoidable, deletion would simply necessitate yet more insertion. This illustrates the sense in which disparities are introduced “only to the extent necessary,” with respect to surface orientedness. Output restrictions may motivate the introduction of disparities between input and output, but they do not open the floodgates for arbitrary numbers of disparities: only those that contribute to the better satisfaction of the output restrictions will possibly be introduced.

This example, and many others like it, motivate the investigation of surface orientedness in phonological theory. That in turn raises the question of how to properly formalize the intuitive notion, so that the motivating patterns in fact follow as predicted consequences of the theory.

1.1.2 Formalizing surface orientedness

While the statement in (1) may seem intuitively clear, formalizing it in a satisfactory way is not so straightforward. It isn’t immediately clear what kind
of thing an output restriction is, or what exactly it means for disparities to be “to the extent necessary” to satisfy output restrictions. Furthermore, it is not immediately obvious how to give such a formalization without delving deeply into the mechanics of a particular theory.

We can abstract away from the mechanics of particular theories by considering the map defined by a phonology. A phonological map is a collection of annotated pairings of inputs with outputs (this definition will be revised in Chapter 2, but will work for now). For each linguistic input, a map contains exactly one input–output pair (like a mathematical function), and for each pair an indication of the disparities between them. A map does not make reference to how the grammar determines the output for an input, it only lists the disparities that are applied to the input along with the resulting output. An individual annotated pair is referred to as a mapping; a map is a collection of mappings.

A more narrowly focused intuition along the lines of (1.1) is that inputs that are identical to well-formed outputs will map to themselves (with no disparities). If disparities are introduced only to the extent necessary, then if you can do without any disparities, you will do without any disparities. This has sometimes been called the identity map property (Prince and Tesar 2004): a well-formed output, when used as an input, maps to itself. Formally, this is equivalent to the mathematical property of idempotency for a unary operation: applying the operation twice yields the same value as applying it only once.  

The canonical example of a phenomenon that violates the identity map property is a chain shift. This can be illustrated with an example based on an analysis by Kirchner (1995) of Etxarri Basque vowel raising in hiatus (this example is examined in greater detail in Section 4.2.1). In the hiatus environment, an input mid vowel e changes to the high vowel i: /e/ → [i]. However, in the same environment, an input high vowel i changes to the raised vowel i ː /i/ → [i ː]. The high vowel [i] is a grammatical output (setting aside the conditioning environment for purposes of presentation), but when used as an input it does not emerge unaltered; instead, the raised vowel [i ː] is the output. This violates idempotency, which would require that if [i] is the output for some input, like /e/, then it must map to itself.

Intuitively, a chain shift is not output oriented. In the Etxarri Basque vowel raising case, if the single disparity in /e/ → [i] is sufficient to satisfy the output restrictions (whatever they are), then no disparities should be needed for input /i/: it should be the case that /i/ → [i]. If, on the other hand, the single disparity in /i/ → [i ː] is necessary to satisfy the output restrictions, then the disparity of

---

3 Idempotency has been defined this way in the context of lattice theory, for example (Szász 1963).
/e/ → [i] ought to be insufficient: it should be the case that /e/ → [i^2]. The failure to be output oriented makes chain shifts interesting and raises questions about how to best analyze and understand them.

Within the theory of SPE (Chomsky and Halle 1968), input–output disparities result from the application of phonological processes, implemented as rewrite rules. The issue of surface orientedness can then be approached by examining the extent to which the behavior of the processes can be characterized in terms of the final output of the derivation. In this view, surface orientedness is a property of processes with respect to derivations. The conditions determining where a process can apply are solely conditions on the structure before the rule has applied, with no explicit reference to the endpoint of the derivation. Not much can be concluded solely on the basis of the definition of the process itself, in isolation. But one can examine the relationship of a process to an output in the context of a derivation for that output.

Interestingly, in rule-based phonology the focus has been not on directly characterizing what it means to be surface oriented, but on directly characterizing what it means to not be surface oriented. The concept of phonological opacity has conventionally been used to characterize situations in which the behavior of a phonological process in a derivation fails to be clearly reflected in the output of that derivation (Kiparsky 1971, Kiparsky 1973). Expanding, a process would be opaque in a phonological map if it was opaque in the derivation associated with at least one input–output mapping of the map. Roughly put, a process is said to be opaque if it contributes meaningfully to the analysis of a phonological map (either by applying or by not applying), but the conditions for its (non-)application are not reflected in the output for some derivation. Surface orientedness is then characterized by complementarity: surface orientedness is identified with a lack of opacity.

This can involve either of two somewhat different types of circumstances. In one circumstance, the output contains a component satisfying the conditions for the application of a process. The fact that the component is in the output means that the process has not applied to that component, despite meeting the conditions of the process. In the terminology of McCarthy 1999, such a process is not surface-true in that derivation (for that component). In the

4 This suggests a view in which surface orientedness is the “default” situation, holding of most of phonology, and in which the phenomena that seem not to be surface oriented are the more unusual.

5 Processes which are not opaque are often labeled “transparent.”

6 Or at least, if the process applied to an earlier form of that component during the derivation, then the component was subsequently returned to the state satisfying the conditions of the process, and the process did not apply again to that component prior to the completion of the derivation.
other circumstance, a process has applied, and the product of that process is present in the output. However, some relevant aspect of the conditions necessary for the application of the process no longer remains. In the terminology of McCarthy, such a process is not surface-apparent (in that derivation, for that component). One could then take the view that a phonological map is surface oriented if none of the processes used in defining it are opaque in any of the derivations giving rise to the map. On this view, evaluating the surface orientatedness of a phonological map is dependent on the particular processes used to characterize it.

Using phonological processes to assess surface orientatedness can be particularly awkward in a theory like Optimality Theory (Prince and Smolensky 1993/2004), in which processes are not primitives of the theory, but are at best descriptive commentaries on the theory, subject to equivocation. Observing that there are no processes, opaque or otherwise, and claiming therefore that OT is vacuously surface oriented, seems merely to duck the issue entirely, rather than address it. OT may not have processes, but it does introduce disparities between inputs and outputs and does so by categorizing (in various ways) both disparities and potential occasions for their introduction. Pursuit of intuitions regarding surface orientatedness in such theories seems to require a characterization of surface orientatedness that is not dependent on processes. Two possible approaches come to mind: (a) a different characterization in terms of the theoretical constructs of another particular theory (like Optimality Theory); (b) a more abstract characterization that makes reference only to the inherent properties of maps themselves. This book is the result of pursuing option (b).

1.2 Surface orientatedness in Optimality Theory

Before outlining the proposed abstract characterization of surface orientatedness, I will in this section briefly examine a couple of attempts to characterize surface orientatedness in terms of the theoretical constructs of Optimality Theory. I argue that both attempts are ultimately unsatisfactory in that they fail to capture some key aspects of surface orientatedness. This provides further motivation for pursuing the more general characterization described in the next section.

7 By “process” I explicitly mean a part of a theory which bundles together specification of a conditioning environment and a change to be made. The term is also sometimes used in a descriptive sense, referring to phonological patterns relating changes to conditioning environments. The latter sense is applicable to Optimality Theory, as it is to any theory which involves mapping inputs to outputs. The descriptive notion of process refers to patterns to be explained by phonological theory, not components of phonological theory which explain patterns.
1.2 Surface orientedness in Optimality Theory

The two attempts to characterize surface orientedness in terms of Optimality Theory are an interesting pair, because they lead to opposite conclusions about the capacity of Optimality Theory to realize maps that are surface oriented. Both focus on markedness constraints as the particularly important theoretical constructs, because markedness constraints by definition only evaluate outputs, and are quite easily understood as restrictions on outputs.

1.2.1 Markedness violations as opacity

One attempt builds on a general notion of opacity as a matter of generalizations that are not surface-true. This is the view put forth by Idsardi (2000): “An opaque generalization is a generalization that does crucial work in the analysis, but which does not hold of the output form.” Note that opacity is a property of generalizations, not maps. But one could by extension suggest that a particular analysis of a map is not surface oriented if it involves one or more opaque generalizations. This view of surface orientedness can be taken to suggest that OT is inherently not surface oriented: generalizations in OT are expressed by constraints, constraints are inherently violable, and in practice nearly any worthwhile OT analysis involves at least some constraint violation. This includes violation in optimal candidates of constraints that are active in the grammar (constraints that eliminate competitors), active constraints being ones that are clearly doing crucial work. Because markedness constraints only evaluate outputs, violations of markedness constraints in particular constitute instances where generalizations do not hold of the output form. The view that OT markedness constraints which are violated in grammatical outputs constitute generalizations which do not hold of the output form has been stated explicitly by McCarthy (1999: 332) and Idsardi (2000: 342).

The analogy between processes and markedness constraints is less than perfect. Processes are generalizations about derivations, not outputs. One cannot determine if a structure in an output is the result of a particular process based on the output alone; it is necessary to recover the relevant aspects of the derivation. Markedness constraints are generalizations about outputs. A structure in an output does or does not constitute a violation of a markedness constraint regardless of whether it is faithfully preserved from the input or it is in part a consequence of disparities between the input and the output.

It should be emphasized that (to the best of my knowledge) neither McCarthy nor Idsardi has claimed that markedness constraints constitute the only linguistically relevant generalizations in an Optimality Theoretic context. But the nature of other generalizations is left unclear, particularly with respect to opacity. The faithfulness constraints might seem like natural instances of such other
Characterizing surface orientedness in phonology

generalizations in Optimality Theory, but it is not immediately obvious how a faithfulness constraint can be said to hold or not hold of an output. Simply asserting that any violation of a faithfulness constraint constitutes a failure of the generalization to be surface-true/apparent leads toward the thoroughly unenlightening classification of maps by whether or not they permit any disparities at all. The class of all and only maps that permit no disparities anywhere, while simple enough to describe, completely fails to correspond to the class of surface-oriented maps in any interesting way. Surface orientedness requires that disparities be introduced only to satisfy output restrictions, not that disparities never be introduced.

The statement that markedness constraints which are violated in grammatical forms constitute generalizations that don’t hold of surface forms is straightforward. However, the surface-trueness of markedness constraints alone doesn’t seem to do justice to the intuition of “surface oriented” as a property of maps. To take an example from McCarthy, he offers, as an example of a generalization in an OT analysis that isn’t surface-true, an analysis of syllable structure involving the constraint Onset, which is violated by syllables that lack an onset, and expresses the generalization “syllables have onsets.” The constraint plays an active role in the analysis, yet there are grammatical outputs with onsetless syllables: the language permits word-initial syllables to lack onsets.\footnote{This is a piece of a more complex illustration given by McCarthy 1999, fn. 1, illustrating different kinds of non-surface-true generalizations.} Suppose, for the sake of discussion, that consonants are inserted into onset position to ensure that non-initial syllables have onsets, so the map includes mappings like the ones in (1.4).

(1.4) Mappings in which non-initial syllables must have onsets
/kika/ → [ki.ka]
/ika/ → [i.ka]
/kia/ → [ki.ka]
/ia/ → [i.ka]

While the onsetless word-initial syllables certainly constitute violations of the constraint Onset, the map itself nevertheless seems surface oriented: consonants are inserted (introducing disparities) only to the extent necessary to ensure that non-initial syllables have onsets (an output restriction). The generalization expressed by Onset is violated on the surface, but relates to a generalization that is not violated on the surface: all non-initial syllables have onsets. The disparities are characterizable with an output restriction, even if that restriction isn’t identical in content to any single markedness constraint.
A useful comparison is with a chain shift. Modifying the previous example, suppose we now have a map in which consonants are inserted to ensure that non-initial syllables have onsets (like the previous example), and additionally in which a consonant is inserted into onsetless initial syllables, but only when all other syllables have onsets from the input (not inserted). Such a map would include mappings like the ones in (1.5).

(1.5) Mappings that include a chain shift

\begin{align*}
/kika/ & \rightarrow [ki.ka] \\
/i ka/ & \rightarrow [ki.ka] \\
/kia/ & \rightarrow [ki.ka] \\
/ia/ & \rightarrow [i.ka]
\end{align*}

The mappings /ia/ $\rightarrow [i.ka]$ and /ika/ $\rightarrow [ki.ka]$ jointly constitute a chain shift pattern. If the introduction of disparities were motivated purely by output restrictions, and /ia/ $\rightarrow [i.ka]$, then [i.ka] ought to be “good enough” with respect to the output restrictions (after all, it is a valid output). It seems that /ika/ should map to [i.ka], a “good enough” output not requiring any disparities with the input (no disparities are necessary). If [i.ka] is not “good enough” as an output, so that the disparity of inserting an initial consonant is warranted in /ika/ $\rightarrow [ki.ka]$, then that disparity should be warranted for input /ia/ also: /ia/ should map to [ki.ka]. As McCarthy puts it, the issue is the failure of /ia/ to make a “fell swoop” all the way to [ki.ka] (McCarthy 1999: 364). The disparity of inserting an initial consonant is unavoidably conditioned by something other than the “goodness” of the output itself. If in one instance [i.ka] is good enough, and in the other instance it isn’t, then something other than restrictions on the output alone must be conditioning the introduction of the disparity (the inserted consonant).

The distinction between the two examples is not simply a matter of distinct phonotactic regularities. Both examples have the same output inventories: the valid outputs are [ki.ka] and [i.ka]. Both examples have outputs in which the initial syllable lacks an onset. Both examples have no outputs in which a non-initial syllable lacks an onset. The difference concerns the relationships between output restrictions and the disparities that are introduced in mappings.

Note that the intuition concerning chain shifts, as stated here, is about the map itself, not about any generalization/output restriction that one might use in describing the map (such as “non-initial syllables must have onsets”). The key mappings (both grammatical and not) and their disparities are given in (1.6). Since /ia/ $\rightarrow [i.ka]$ has the inserted [k] in the second syllable as its only disparity, and the first syllable does not have an onset, the input /ika/, which wouldn’t require any disparities to reach the same output [i.ka], would
be expected under surface orientedness to map to [i.ka]: any introduction of disparities for input /ika/ is not required by output restrictions, because [i.ka] is a valid output. The fact that /ika/ maps to [ki.ka] instead violates the intuitions of surface orientedness. The violation of surface orientedness can be described in terms of the fact that [i.ka] is a valid output, without reference to the content of any particular output restriction. Indeed, it shouldn’t be possible for a pure output restriction to distinguish two cases with identical outputs.

(1.6) Key input–output pairs with their disparities

\[
\begin{align*}
/i\alpha/ & \rightarrow [i.\alpha] & \text{Disparities: inserted} & [k] \text{ in second syllable} \\
*/i\alpha/ & \rightarrow [i.\alpha] & \text{Disparities: none} \\
/i\alpha/ & \rightarrow [k.i.\alpha] & \text{Disparities: inserted} & [k] \text{ in first syllable}
\end{align*}
\]

The discussions of opacity in the work by Kiparsky, McCarthy, and Idsardi all focus on properties of the generalizations expressed by immediate constructs of phonological theories, e.g., the rules of SPE or the markedness constraints of Optimality Theory. Since phonology is in the business of characterizing generalizations about phonological systems, such a focus is (quite reasonably) to be expected. And yet, it ties the characterization of surface orientedness tightly to the constructs of particular theories and in the process misses the intuition that the maps themselves are or are not surface oriented. The generalization that non-initial syllables must have onsets feels quite surface oriented, in a way that is insensitive to whether the relevant theoretical construct is a statement about syllables (which would not be surface-true) or a statement about non-initial syllables (which would be surface-true).

1.2.2 Markedness constraints cause disparities

Another attempt to characterize surface orientedness in terms of Optimality Theory argues that the only constraints that can cause disparities are markedness constraints, and therefore disparities are always a consequence of the enforcement of output restrictions (the markedness constraints). This leads to the (opposite) conclusion that OT is inherently surface oriented.

This view appears to focus to a somewhat greater extent on the map itself. It makes no reference to the content of markedness constraints, save for the fact that they only evaluate output forms. In particular, it says nothing about whether markedness constraints are actually “surface-true” (unviolated) in a language. The argument implicitly rests on the intuitive assumption that candidates with zero disparities satisfy all faithfulness constraints, and thus can only lose to another candidate on the basis of a markedness constraint (whether or not that markedness constraint is surface-true in the language overall).
This characterization is missing the “only introduced to the extent necessary” part of the intuition in (1.1). If disparities are “introduced to the extent necessary,” then the disparities must in fact be introduced whenever “necessity” arises. To respect “to the extent necessary,” it isn’t enough to say that when disparities are introduced, they are necessary to satisfy the output restrictions; one must also say that when disparities are necessary to satisfy the output restrictions, they must be introduced. But nothing about the observation that zero-disparity candidates satisfy all faithfulness constraints addresses any obligation for disparities to be introduced at all.

One could attempt to address this objection by altering the view to distinguish between markedness constraints and the output restrictions that arise from them. The claim would be that disparities are introduced as a result of output restrictions (which are necessarily surface-true), which in turn are derived from markedness constraints (which are not necessarily surface-true). The issue is not the surface-trueness of the individual constraints, but the sufficiency of the surface-true restrictions that the markedness constraints collectively give rise to. The markedness constraint *Onset* may not be surface-true in the language, but the output restriction “non-initial syllables have onsets” is surface-true in the language.

Unfortunately, once one separates output restrictions from output constraints, there is no easy way to reason from output constraints alone back to output restrictions. It may be the case that every introduction of a disparity involves improving satisfaction of a markedness constraint, but that is very different from claiming that the markedness constraint is the only determining factor. If in general the disparities of a map for an OT grammar arise from the interaction of markedness and faithfulness constraints, then the contribution of the faithfulness constraints must be accounted for.

The inadequacy of this view of OT as inherently surface oriented is sharply revealed by the observation that there are OT systems consisting only of markedness and faithfulness constraints that define maps with chain shifts, a canonical case of a map phenomenon that is intuitively not surface oriented, as was discussed in the previous section. Several examples of OT grammars defining chain shifts are presented in later chapters.

### 1.2.3 Something more general

The two intuitive views about OT just described are different in several respects. One respect in which the two are similar, however, is a near-exclusive focus on markedness constraints. I claim that this focus is a significant part of why both views fail to capture the intuitions of surface orientedness. To foreshadow
some of the results of Chapter 3, the surface orientedness of an OT system is largely determined not by the markedness constraints but by the faithfulness constraints.\footnote{More generally, by the constraints that refer to things other than the output.}

Focusing on opaque generalizations ties the characterization of surface-oriented phonological maps to the specific constructs of particular theories. What is missing is a more abstract characterization of surface-oriented phonological maps themselves, expressing the sense that some maps are or are not inherently surface oriented. At first glance, this doesn’t seem surprising. The intuition of surface orientedness in (1.1) makes reference to “output restrictions,” so it seems like one might need to have a theory of what the output restrictions are before being able to assess maps. Nevertheless, I will define a formal property of maps, output drivenness, that captures the intuition of surface orientedness without needing to specify what the output restrictions are. In other words, I will define what it means for disparities in a map to be driven (required) by output restrictions without having to commit to what the output restrictions are. I argue that output drivenness does a much better job of capturing the intuitions of surface orientedness.

In tandem with not needing to specify what the output restrictions are, I am not claiming that the output restrictions whose existence is implied for an output-driven map must themselves be theoretical constructs of any adequate theoretical account of that map. A grammar can provably generate only syllables of the form CV without containing any explicit statements of the sort “all syllables are CV,” or even making explicit reference to the concept of a CV syllable. The theory of output drivenness is less concerned with the specific outputs that a grammar arrives at, and more concerned with how it arrives at them (which disparities are introduced, and under what conditions). The substance of what kinds of output restrictions actually drive phonologies is clearly of fundamental importance to phonological theory, but output drivenness is a property of maps that generalizes across different possible output restrictions. The accomplishment is the factoring out of the “drivenness” from any specific substantive details of the “output.”

It is not a goal of this work to argue against formal linguistic theory, or to suggest that every interesting property of phonologies can be captured without reference to familiar constructs of linguistic theories. That would be, quite simply, self-defeating. Far from seeking to replace or marginalize specific linguistic theories, the point of the concept of output drivenness is to help provide insight into specific linguistic theories, as well as the phenomena they
seek to explain. It is a goal of this work to gain further insight into linguistic theories by understanding how output drivenness relates to the constructs of particular linguistic theories; such understanding is developed for Optimality Theory in Chapter 3 and Chapter 4.

1.3 Formalizing surface orientedness: output-driven maps

Chapter 2 provides a formalization of surface orientedness in purely representational terms, that is, as a property of the phonological map independent of any particular analysis in terms of processes or constraints. The essence of the idea is stated in (1.7). The separate terminology “output-driven” is employed to distinguish the intuition from the formal concept: “surface oriented” refers to the general intuition in (1.1), while “output-driven” refers to the formally defined property of maps outlined in (1.7).

(1.7) A phonological map will be said to be output-driven if, for any mapping from an input to an output, any other input that has greater similarity to the output also maps to the same output.

The statement in (1.7) won’t get you very far without an appropriate definition of “greater similarity.” The sense of similarity that captures the linguistic intuitions of surface orientedness is briefly described in (1.8), and discussed at great length in Chapter 2.

(1.8) One input, B, has greater similarity to an output than another input A if the disparities between B and the output are a subset of the disparities between A and the output.

The concept of output-driven map expressed in (1.7) and (1.8) fits the intuition of surface orientedness given in (1.1). For any grammatical mapping from input to output with some set of disparities, the input identical to that output must map to that same output: a form (as an input) clearly has greater similarity to itself (as an output) than any other input form has. Thus, output-driven maps are necessarily idempotent. If the disparities between an input and its output are introduced only to the extent necessary to satisfy output restrictions, then any input which has only a subset of those disparities with the same output should map to that same output; there are only a subset of the obstacles to the same destination. In an output-driven map, inputs capable of reaching the same output with a subset of the disparities in fact do so, rather than mapping to a different output via a different set of disparities (no matter how well formed that other output might be). Chain shifts are inherently not output-driven because they are not idempotent.
Characterizing surface orientedness in phonology

Output drivenness is a stronger condition than mere idempotency. There are maps that are idempotent but not output-driven. Examples of such maps, involving derived environment effects, will be given in Chapter 2. The distinction between output drivenness and idempotency lies in the patterning of the non-identity mappings: output drivenness imposes conditions on the mapping of all inputs, not just those that are identical to well-formed outputs. The theory of output-driven maps reveals the commonality between chain shifts and derived environment effects: the two turn out to be minor variants of the same phenomenon. In addition to formalizing what it means to be surface oriented, output drivenness sheds new light on things which are not surface oriented.

Output drivenness is independent of the theory used to define particular maps or predict which maps are allowable. It does not presume SPE-type ordered rules, Optimality Theory, or any other such theory. It is dependent on the representational commitments used to define the linguistic inputs and outputs, the representational commitments concerning correspondence relations between the inputs and outputs, and the characterization of input/output disparities. Thus, it would be a large mistake to suggest that evaluating a map as output-driven or not is “theory independent” in the more general sense of theory (lower case); the (non-)output drivenness of a phonological map will be heavily dependent on theoretical representational commitments.

Output drivenness is a property of maps. It is not a property of datasets. Output drivenness makes crucial reference to the inputs assigned to outputs, inputs which must be posited in response to data, not found in data. Further, for any given mapping to which one is committed, output drivenness makes reference to the mapping for every input with greater similarity to the output of the original mapping. A map must specify the mappings for all of the inputs, not just ones that happen to be attested or included in a sample. This is another sense in which output drivenness is theory dependent in the more general sense: output drivenness is a property of a complete map, and in general will depend on commitments concerning both the inputs and the grammaticality of forms that are not actually attested.

Output drivenness is not a representational way of arriving at the same conclusions as traditional process opacity. While it is the case that a number of maps that are often described as involving opaque processes are in fact not output-driven, the two properties are fundamentally distinct. This is particularly clear in cases where the same phonological map can be plausibly characterized in terms of either opaque generalizations or transparent (non-opaque) generalizations. A single map cannot be both output-driven and not; only a single judgment is rendered for the map itself. An example of such a case is...
discussed in Section 2.3. Process opacity is dependent on the particular choice of processes; more generally, opacity is dependent on the particular choice of generalizations, be they processes, markedness constraints, or whatever other constructs are chosen. Output drivenness is a property of maps themselves, not chosen generalizations.

1.4 Output drivenness and Optimality Theory

If interesting properties of maps can be identified, then various theoretical devices can be evaluated with respect to those properties. This is similar to what is accomplished by the Chomsky hierarchy of formal languages (Chomsky 1959): language-generating devices can be evaluated for their generative capacity, based on their (in)ability to generate the languages of the independently defined language classes of the hierarchy. The first goal of this book is to define a formal property, output drivenness, that captures intuitions of surface orientedness of maps. The second goal is to evaluate some theoretical devices, specifically constructs of Optimality Theoretic systems, with respect to their capacity for defining output-driven and non-output-driven maps. Instead of evaluating direct surface-trueness, I will be evaluating theoretical constructs with respect to the output drivenness of the maps that they define.

Optimality Theoretic systems can be evaluated in terms of their ability to generate output-driven and non-output-driven maps. Some OT grammars describe output-driven maps, while others describe non-output-driven maps. If we accept output drivenness as a formal characterization of surface orientedness, then we reach the conclusion that Optimality Theory is neither inherently surface oriented nor inherently not surface oriented. The analysis of OT and output drivenness will reveal that what determines the (non-)output drivenness of the maps definable by an Optimality Theoretic system is the content of the faithfulness constraints (more precisely, the “input-referring” constraints). Thus, the evaluation of OT systems with respect to output drivenness turns out to effectively be a development of the theoretical understanding of faithfulness in OT.

In Chapter 3, conditions on Optimality Theoretic systems are derived that are sufficient to ensure that all maps defined by such a system are output-driven. The conditions ensuring that the phonological maps of an OT system are output-driven are derived directly from the definition of Optimality Theory. These conditions break into separate conditions on GEN (the candidates) and on CON (the constraints). Of particular interest is the fact that the conditions on the constraints apply separately to each constraint. If each constraint can
be shown to individually satisfy the given conditions, then any ranking of any combination of them will yield an output-driven map. Proofs that a variety of basic OT constraints meet these conditions are given in Chapter 3.

Chapter 4 examines the ways in which OT-defined maps can fail to be output-driven. If a constraint fails to meet the conditions guaranteeing output drivenness, then it must exhibit a particular behavior. If a map is not output-driven, and GEN meets the appropriate conditions for output drivenness, then the OT system defining the map must have at least one constraint exhibiting this behavior. The conditions derived in Chapter 3 are shown in Chapter 4 to provide a unified understanding of a number of proposals for handling, within OT, maps that are not surface oriented. Chapter 4 includes several examples of analyses proposed in the literature that define maps that are not output-driven, and in every case the responsible constraints are shown to exhibit the key behavior. The examined proposals include local conjunction, local disjunction, positional faithfulness, simple antifaithfulness, and sympathy theory. Output drivenness makes it possible to unify the understanding of a number of different proposals within Optimality Theory for addressing certain phenomena often characterized in terms of process opacity.

1.5 Output drivenness and learning

Formal conditions that characterize entire phonological maps, in addition to being of interest for phonological theory, have great potential significance for theories of language learning. Interest in idempotent maps arose in the study of phonotactic learning: if it could be assumed that the target grammar mapped each well-formed (grammatical) linguistic form to itself, then a learner could start out assuming that the underlying form for each observed surface form was identical to said surface form and thereby learn non-trivial things about the map (Hayes 2004, Prince and Tesar 2004). Output drivenness is a stronger condition than idempotency and has significant implications not only for phonotactic learning, but also for later, non-phonotactic stages of learning, in particular the learning of underlying forms. Output drivenness imposes restrictive structure on the space of inputs relative to the outputs of a map: the fate of one input can determine the fate of a whole subspace of inputs. As will be shown in Chapter 7 and Chapter 8, this structure can be exploited by a learner quite effectively, making it possible to efficiently contend with the combinatorially vast space of possible lexica of underlying forms that a learner must contend with.

One of the things that makes language learning so interesting is that several aspects of the problem are mutually dependent and interact strongly with
each other. Any account of how phonological underlying forms are learned must eventually make significant contact with the learning of the grammar that determines the output for each input. To fully appreciate how the theory of output drivenness can contribute to language learning, it is necessary to understand some of the prior work that has been done on phonological learnability, to see how output drivenness can successfully mesh with existing accounts of phonological learning. Chapter 5 provides background on learnability in Optimality Theory, including work on obtaining information about constraint rankings, the enforcement of restrictiveness biases, and phonotactic learning. Chapter 6 provides background on the learning of phonological underlying forms, briefly describing several approaches, and focusing in more detail on the approach based on inconsistency detection that is the basis for the later combination with output-driven map theory.

Chapter 7 shows how the theory of output-driven maps can be exploited in language learning. Output drivenness creates a system of entailments between the mappings for different inputs: if one input maps to an output, then any other input with a subset of the disparities of the first input also maps to that output. The logical contrapositive must also hold: if one input cannot map to an output, then any input with a superset of the disparities of the first input cannot map to that output. This allows one input to stand in for an entire subspace of the set of possible inputs. By evaluating an input with only a single disparity, if that input cannot map to the output, then any input containing that disparity (along with others) cannot map to the output, and therefore the correct input must not have that disparity with the output. For the evaluation of individual words, this allows the learner to avoid searching all of the possible inputs for a word (a space which grows exponentially in the number of potential disparities), instead evaluating each input with only a single disparity (which grows only linearly in the number of potential disparities). The primary benefit is a huge reduction in the computational cost of learning. The algorithm described in Chapter 7 can successfully learn any system that its predecessor could, while making the learning far more tractable.

Chapter 8 examines a challenging problem of restrictiveness that arises in the learning of underlying forms in tandem with constraint rankings: the problem of paradigmatic subsets. Like other restrictiveness problems, this one involves having more than one grammar consistent with the same data, where one of the grammars is more restrictive than the other: the language generated by the first grammar is a strict subset of the language generated by the other grammar. Paradigmatic subsets are situations where not only are the generated outputs of one grammar a subset of the generated outputs for the other grammar, but
the paradigmatic relations, including morphemic alternations, that surface with one grammar are a subset of the relations that can surface with the other grammar. The techniques for contending with restrictiveness in phonotactic learning (where morphemic alternations are not under consideration) are generally inadequate to contend with paradigmatic subsets. A technique for contending with paradigmatic subsets, based crucially on the theory of output-driven maps, is presented in Chapter 8, along with some discussion of the relation of this proposal to ideas familiar from statistical learning theory.

1.6 The relationship between learnability and linguistic theory

This book can be loosely divided into three parts. The first part concerns the concept of output-driven maps, and how it can capture linguistically significant properties in a purely representational fashion. The second part concerns the relation between output-driven maps and Optimality Theory, examining the particular properties of OT constructs that do or do not permit non-output-driven maps. The third part concerns language learning, showing the significant benefits to learning if the learner has a linguistic theory consisting of an Optimality Theoretic system that defines only output-driven maps.

The relationship of the final part, on learning, to the earlier parts of the book may strike some readers as a bit odd. The first two parts look at output drivenness as a property of interest and discuss several attested phenomena that are arguably not output-driven. This might seem to question the relevance of the idealizations employed in the section on learning, in which the learner presumes that the grammar it is learning defines an output-driven map. I argue that work under these idealizations is nevertheless highly relevant.

One claim motivating the learning work is that, while there are phenomena that are not output-driven, a large majority of phonological phenomena are output-driven. Thus, output-driven maps are a good first take on characterizing the structure of phonological maps. Another claim motivating the learning work is that the space of grammar hypotheses, including possible lexica of underlying forms for morphemes, is far too vast for the learner to contend with in any simplistic, unconstrained fashion. For language learning to be possible, the linguistic theory must have some kind of non-trivial structure connecting candidates with different inputs that can be exploited by a learner. The definition of output-driven maps is a first cut at identifying that structure, permitting analysis of a wide range of basic phonological phenomena while also contributing significantly to efficient learning.
Several major topics in phonological theory and in learning are addressed in this work. Major topics in phonological theory are listed in (1.9), along with indications of the chapters in which they are most directly addressed. Major topics in language learnability are listed in (1.10).

(1.9) Topics in phonological theory addressed in this book
- Surface orientedness and process opacity/transparency (Chapters 1, 2, 3, 4)
- Correspondence (Chapters 2, 3)
- Contrast (Chapters 5, 6, 7, 9)

(1.10) Topics in language learnability addressed in this book
- Error-driven learning, and moving beyond it (Chapters 5, 6, 7)
- What information is stored by the learner (Chapters 5, 6, 9)
- Restrictiveness (Chapters 5, 7, 8, 9)
- Processing multiple words simultaneously (Chapters 6, 7)

Chapter 9 further discusses some topics where linguistic theory and language learnability interact. The relationship between contemporary notions of contrast and the learning of underlying forms is discussed, as are the implications of this work for the concept of an evaluation metric and the kinds of information structures stored by learners during learning. The work in this book is offered as further evidence in support of the view that, when done properly, linguistic theory and language learnability inform and depend upon each other.
2 Output-driven maps

The main idea of output drivenness, and some of its key properties, are presented in Section 2.1. Section 2.2 discusses the concept of relative similarity in greater detail. Section 2.3 examines the important differences between output drivenness and process opacity. Section 2.4 works through the details of a formal definition of output-driven maps, one based on segmental IO correspondence. The rest of the chapter discusses linguistic issues that come up in light of output drivenness.

2.1 The main idea

2.1.1 Terminology: candidates and correspondence
Discussion of generative linguistics necessarily makes reference to the different possible outputs that a given input could map to, as allowed by the representational theory in use. I will use the term candidate to refer any representation consisting of an input, a possible output for that input, and a correspondence relation between the input and the output. The input–output correspondence relation determines the disparities between the input and the output in a candidate.

The candidates of a theory reveal the possible input–output pairs allowed by that theory, independent of any specific map. The mapping for an input in a map is the candidate for that input that is included in the map; which candidates are mappings depends upon the map being referred to. When referring to a map defined by a grammar, the mapping for an input is the grammatical candidate for that input. A phonological map can be thought of as a set of candidates, one for each input. A candidate with input /a/ and output [i] will commonly be denoted /a/[i], with no particular commitment as to its grammaticality for any given map. A mapping with input /a/ and output [i] will commonly be denoted /a/ → [i], with the arrow indicating the grammaticality of the candidate for some given map.
2.1 The main idea

The terms “candidate” and “correspondence” are familiar from the literature on Optimality Theory. However, the concepts they refer to here are not parochial to OT. As explained in Section 2.2.4, any generative theory works in terms of inputs, outputs, and correspondences between them. The label “candidate” correctly suggests that a grammar is choosing from among distinct possibilities, but does not entail that the choice is made via optimization (or any other particular mechanism for choosing).

2.1.2 Inputs of greater similarity yield the same output

The following illustration characterizes vowels in terms of two height features: +/-low and +/-hi. Following Chomsky and Halle (1968: 305), the feature combination [+low, +hi] is ruled out representationally. Thus, there are three vowels in the illustration: [i] [-low, +hi], [e] [-low, –hi], and [a] [+low, –hi].

Output drivenness requires a notion of similarity between representations. Intuitively, we want to be able to say when one input representation has greater similarity to an output representation than another input has. Similarity between an input and an output is expressed as an individuation of the disparities between the input and the output. For instance, a disparity in feature value is an instance where corresponding input and output segments disagree in the value of a feature. /ε/ has greater similarity to [i] than /a/ has to [i], because /a/ and [i] disagree on every feature that /ε/ and [i] disagree on: there is a disparity between /ε/ and [i] on the feature [hi], a disparity that also exists between /a/ and [i]. Further, /a/ and [i] have an additional disparity between them, on the feature [low], a disparity that does not exist between /ε/ and [i]. Thus, if we have inputs /ε/ and /a/, and output [i], then /ε/ has greater similarity to [i] than /a/ has to [i]. Similarity here is not simply a matter of the number of disparities; the reasons why will be discussed in greater detail in Section 2.2.3.

Input /ε/ has greater similarity to [i] than /a/ has to [i] not because it has fewer disparities, but because the disparities of /ε/ to [i] are a subset of the disparities of /a/ to [i].

The main idea of output drivenness is this: if a given input is mapped to an output, then any other input which has greater similarity to that output must also be mapped to that output. Suppose /a/ maps to [i]: /a/ → [i]. If the map is output-driven, then /ε/ necessarily also maps to [i]: /ε/ → [i]. This is because /ε/ has greater similarity to [i] than /a/ has. The mapping /a/ → [i] also entails the mapping /i/ → [i], as /i/ clearly has greater similarity to [i] than /a/ has. The map given in (2.1) is therefore output-driven.

(2.1) Output-driven: /a/ → [i] /ε/ → [i] /i/ → [i]
The map in (2.2) is also output-driven. Here, there is no required relationship between /a/ and /e/. Given that /a/ maps to [a], /e/ does not have greater similarity to [a] than /a/ has, so the mapping /a/ → [a] has no implications for the output that /e/ is mapped to (with respect to output drivenness). Similarly, given that /e/ maps to [i], /a/ does not have greater similarity to [i] than /e/ has, so the mapping /e/ → [i] has no implications for the output of /a/. The mapping /e/ → [i] does, however, entail the mapping /i/ → [i].

(2.2) Output-driven: /a/ → [a]  /e/ → [i]  /i/ → [i]

Note that the relationship between the mappings in an output-driven map is one of entailment. There is nothing about mapping /e/ → [i] by itself that makes it obligatory in an output-driven map. The map in (2.3) is also output-driven. Output drivenness here is consistent with /e/ not mapping to [i] because /a/ does not map to [i]. In (2.3), candidate /e/[i] is not grammatical, instead candidate /e/[e] is grammatical.

(2.3) Output-driven: /a/ → [a]  /e/ → [e]  /i/ → [i]

One immediate consequence is that all output-driven maps are idempotent: any representation which is the output for some input is the output for itself. This is because any representation necessarily has greater similarity to itself than any other representation has to it. A form can map to itself with no disparities; identity of form is as similar as you can get. If we have /a/ → [e] in an output-driven map, then the mapping /e/ → [e] follows from the definition of output-driven map. This means that output-driven maps automatically disallow chain shifts; any map exhibiting chain shifts is not idempotent and is not output-driven. An example of such a non-output-driven map is given in (2.4).

(2.4) Non-output-driven: /a/ → [e]  /e/ → [i]  /i/ → [i]

A chain shift is a case where the “greater-similarity input” is identical to the output for the “lesser-similarity input.” Given that /a/ → [e], the input /e/ has greater similarity to output [e] than /a/ does, by virtue of being identical. For output [e], /a/ is the lesser-similarity input, and /e/ is the greater-similarity input. In an output-driven map, /a/ → [e] entails /e/ → [e]. The failure of the map in (2.4) to satisfy this entailment makes it non-output-driven.

There are also non-output-driven maps containing patterns in which the greater-similarity input is distinct from the output of the first form. An example is given in (2.5).

(2.5) Non-output-driven: /a/ → [i]  /e/ → [e]  /i/ → [i]
In this map, /a/ maps to [i]. /e/ has greater similarity to [i] than /a/ has, without being identical to [i]. The candidate /a/[i] has two disparities, one in low and one in hi, while the candidate /e/[i] has only a single disparity, the same disparity in hi (–hi in the input, +hi in the output) as in /a/[i]. Contra output drivenness, /e/ does not map to [i] in (2.5); it maps to itself. Łubowicz (2003: 69) labels this kind of map pattern a derived environment effect (inspired by the idea that [e] is an acceptable output only when it results from input /e/, not when it would be derived from some non-identical input, like /a/). The example in (2.5) is significant because the map is idempotent; the grammatical outputs, [i] and [e], each map to themselves. Thus, output drivenness is a stronger property than idempotency. All output-driven maps are idempotent, but not all idempotent maps are output-driven.

A special case of a chain shift is a circular chain shift; an example is given in (2.6). Given that /e/ → [i], the fact that the greater-similarity input /i/ does not map to [i] makes it a chain shift; the fact that /i/ maps to [e], identical to the lesser-similarity input, makes it a circular chain shift.

(2.6) Non-output-driven: /a/ → [a] /e/ → [i] /i/ → [e]

In a prototypical derived environment effect pattern, such as (2.5), the greater-similarity input, /e/, maps to itself. However, it is possible to have maps which are not chain shifts, and where the greater-similarity input maps not to itself but to some other form entirely. Illustrating such a case requires at least four forms. Such a case is depicted in (2.7), where the form [o] is included, and the feature [round] is presumed to distinguish +round [o] from –round {[a], [e], [i]}.

(2.7) Non-output-driven: /a/ → [i] /e/ → [o] /o/ → [o] /i/ → [i]

The mapping /a/ → [i] would entail /e/ → [i] in an output-driven map. Instead, in (2.7), /e/ maps to /o/, which introduces a disparity in the feature round, but not in the feature hi. This map does not include a chain shift pattern (it is idempotent), but the greater-similarity input, /e/, maps not to itself, [e], nor to the output for /a/, [i], but to something else entirely. This makes no difference where output drivenness is concerned. The fact that /a/ maps to [i] but /e/ does not map to [i] means that (2.7) is not an output-driven map, no matter what (other than [i]) /e/ maps to.

2.1.3 Unifying surface orientedness
As stated in (1.1), the essence of surface orientedness is that disparities between input and output should be introduced only to the extent necessary to satisfy
Output-driven maps

Output-drivenness translates “to the extent necessary” into entailment relations between candidates. This accomplishes two things simultaneously: it unifies the different map patterns that are not surface oriented into a single phenomenon and it characterizes surface orientedness without committing to what the particular output restrictions are.

Chain shifts and derived environment effects are two canonical map patterns that are not output-driven. Looked at from the point of view of output-driven maps, they differ only in the nature of the greater similarity input. In a chain shift, the greater-similarity input is identical to the output for the lesser-similarity input. In a derived environment effect, the greater-similarity input is not identical to the output for the lesser similarity input. Both involve situations where A maps to C, B has greater similarity to C than A has to C, and yet B does not map to C. In a chain shift, B = C, while in a derived environment effect, B ≠ C. That is the only difference. The theory of output-driven maps unifies map patterns that are not surface oriented, including the chain shift pattern and the derived environment effect pattern; they are variations of a single phenomenon.

Chain shift patterns are intuitively not surface oriented: if /a/ → [e] and /e/ → [i], as in (2.4), then to be surface oriented, [e] would have to both satisfy the output restrictions (because of the adequacy of [e] in /a/ → [e]) and not satisfy the output restrictions (because of the inadequacy of [e] in relation to /e/ → [i]), a clear contradiction. The source of the failure to be surface oriented depends on the status of [e] with respect to the output restrictions. If you believe that [e] does not satisfy the output restrictions, then the responsibility lies with /a/ → [e]: it failed to introduce enough disparities to satisfy the output restrictions (in particular, it failed to introduce a disparity between –hi in the input and +hi in the output). If you believe the [e] does satisfy the output restrictions, then the responsibility lies with /e/ → [i]: it introduces a disparity beyond the extent necessary to satisfy the output restrictions. The two different assignments of responsibility go with two different views about the output restrictions: does [e] satisfy them or not?

The definition of output-driven maps doesn’t require an answer to whether [e] satisfies the output restrictions; it doesn’t require a designation of which mapping is “problematic.” Output drivenness simply determines that an output-driven map couldn’t have both mappings. In this way, output drivenness succeeds in formalizing the intuitions of surface orientedness without committing to the content of the output restrictions. Either /a/ → [e] fails to introduce a necessary disparity, or /e/ → [i] introduces an unnecessary disparity. Either way, it cannot be the case that disparities are introduced only to the extent necessary to satisfy the output restrictions, whatever those output restrictions are.
2.1 The main idea

Similar observations can be made about derived environment effect patterns. If /a/ → [i] and /e/ → [e], as in (2.5), then to be surface oriented, [e] would have to both satisfy the output restrictions (because of the adequacy of [e] in /e/ → [e]) and not satisfy the output restrictions (because of the inadequacy of [e] in relation to /a/ → [i]). The latter is due to the fact that /a/ → [e] incurs only one disparity, +low to –low, while /a/ → [i] incurs that disparity plus an additional one, –hi to +hi: in order for the second disparity to be necessary, the first one alone must be insufficient to satisfy the output restrictions. Again, assignment of responsibility for the failure to be surface oriented depends on the choice of output restrictions.

As with the chain shift pattern, output drivenness is not sensitive to the choice of output restrictions for the derived environment effect pattern. Either /e/ → [e] fails to introduce a necessary disparity, or /a/ → [i] introduces an unnecessary disparity. Either way, it cannot be the case that disparities are introduced only to the extent necessary to satisfy the output restrictions, whatever those output restrictions are.

It is possible for a single map to contain instances of both chain shift and derived environment effect map patterns. The map in (2.8) contains a chain shift: /a/ → [i] and /i/ → [e]. It also contains a derived environment effect: /a/ → [i] and /e/ → [e] as in (2.5). Both instances have the same lesser similarity input, /a/.

The instances differ in the greater similarity input: the chain shift has greater-similarity input /i/, identical to the output for /a/, while the derived environment effect has greater-similarity input /e/, non-identical to the output for /a/.

(2.8) Non-output-driven: /a/ → [i] /e/ → [e] /i/ → [e]

What pattern you see depends upon which inputs you focus on. The mapping /a/ → [i] introduces two disparities, so there are two inputs with greater similarity to [i] than /a/ has to [i]. Each of the latter two inputs, /e/ and /i/, provides an independent opportunity to defy the dictates of output drivenness when paired with /a/ → [i].

The chain shift pattern in (2.8) contrasts with the chain shift pattern in (2.4) in its relation to surface orientedness. In (2.4), the first link of the chain shift, /a/ → [e], introduces a disparity in the feature low. The second link, /e/ → [i], introduces a disparity in the feature hi and does not “reverse” the value of the feature low introduced by the first link. As discussed above with respect to surface orientedness, either the first link fails to introduce a necessary disparity, or the second link introduces an unnecessary disparity. In (2.8), the first link of the chain shift, /a/ → [i], introduces two disparities, for the features low
and hi. The second link, /i/ → [e], introduces a single disparity in hi, but that introduced disparity “reverses” the disparity for the feature hi introduced by the first link. Thus, with respect to surface orientedness, we could say that either the first link introduces an unnecessary disparity (for the feature hi), or the second link introduces an unnecessary disparity, in particular one that reverses a (necessary) feature value introduced by the first link.

By translating the “to the extent necessary” intuition of surface orientedness into entailment relations between candidates, the theory of output-driven maps provides a unified understanding of different patterns that are intuitively not surface oriented, including chain shift and derived environment map patterns. By defining the concept in terms of entailment relations between candidates, the theory of output-driven maps identifies pairs of mappings that cannot coexist in an output-driven map, without needing to determine which of the two mappings is the “problematic” one, allowing output drivenness to succeed without needing to commit in advance to what the precise output restrictions are.

2.2 Relative similarity

Relative similarity is a central concept in the theory of output-driven maps. It concerns two kinds of comparison at once: comparison between the input and output of a candidate, and comparison between two candidates. These two kinds of comparison combine to give substance to the notion of one input having greater similarity than another input to an output. Each kind of comparison is based on a distinct correspondence relation. Correspondence is a central component of similarity between linguistic structures. The importance of various forms of correspondence will be demonstrated in several places in this book.

2.2.1 Relating the disparities of two candidates

The internal similarity of a candidate is an evaluation of the similarity between the input and the output of the candidate. It is expressed negatively, in terms of the disparities in a candidate, and is based on the input–output correspondence relation of the candidate.

The relative similarity of a pair of candidates is a comparison of the internal similarities of the two candidates. Such a comparison is based on a constructed correspondence relation between the disparities of the two candidates. Candidate B has greater internal similarity than candidate A if (i) the candidates have identical output forms; (ii) every disparity in B has an identical corresponding disparity in A. If two candidates have distinct outputs, then neither
2.2 Relative similarity

Figure 2.1. Relative similarity relation (upward is greater internal similarity).

has greater internal similarity than the other; they are not related with respect to relative similarity. If two candidates have identical outputs, but each candidate has a disparity that the other lacks, then neither has greater internal similarity than the other.

Formally, a relative similarity relation is a partially ordered set of candidates: the ordering relation is reflexive, antisymmetric, and transitive. Reflexivity means that each candidate is paired with itself in the relation: any given candidate has greater internal similarity than itself.\footnote{The slightly less ambiguous but much more cumbersome phrase “internal similarity which is at least as great” is being avoided here.} The partial order can be partitioned by output. Each part of the partition contains all and only those candidates sharing a particular output; by definition, two candidates with different outputs cannot be ordered with respect to each other. The part of the relative similarity relation involving the candidates sharing a particular output can be labeled the \textbf{relative similarity order} for that output. Figure 2.1 depicts the relative similarity relation for the vowel height candidates described in Section 2.1.2.

In each relative similarity order in Figure 2.1, the top candidate (the one with the greatest internal similarity for that order) has zero disparities. The top candidate has as its input a form identical to the output shared by all of the candidates of the order. Candidate /a/[a] has greater internal similarity than candidate /e/[a], which has one disparity involving the feature low, while /e/[a] has greater internal similarity than /i/[a], which shares the disparity involving low but also has an additional disparity involving the feature hi. Candidate /e/[e] has greater internal similarity than both /i/[e] and /a/[e]. However, /i/[e] and /a/[e] are not ordered with respect to each other, even though they have the same output: /i/[e] has a disparity in hi but not in low, while /a/[e] has a disparity in low but not in hi. Each has a disparity that the other lacks.
Recall the maps in (2.1), (2.2), (2.4), and (2.5), repeated here.

(2.1) Output-driven: \( /a/ \rightarrow [i] /e/ \rightarrow [i] /i/ \rightarrow [i] \)

(2.2) Output-driven: \( /a/ \rightarrow [a] /e/ \rightarrow [i] /i/ \rightarrow [i] \)

(2.4) Non-output-driven: \( /a/ \rightarrow [e] /e/ \rightarrow [i] /i/ \rightarrow [i] \)

(2.5) Non-output-driven: \( /a/ \rightarrow [i] /e/ \rightarrow [e] /i/ \rightarrow [i] \)

The candidate \( /a/\lbrack i\rbrack \) has two other candidates above it in the relative similarity relation: \( /e/\lbrack i\rbrack \) and \( /i/\lbrack i\rbrack \). For a map with grammatical \( /a/\lbrack i\rbrack \) to be output-driven, both of those candidates must also be grammatical. In (2.1), they are. In (2.5), candidate \( /e/\lbrack i\rbrack \) is not grammatical; \( /e/ \) maps to \( /e/\lbrack e\rbrack \) instead. Thus, (2.5) is not output-driven. The grammaticality entailments between candidates follow directly from the relative similarity relation: the grammaticality of a candidate entails the grammaticality of all candidates ordered above it.

The candidate \( /a/\lbrack a\rbrack \) has no other candidates above it in the relative similarity relation. With respect to output drivenness, \( /a/ \rightarrow [a] \) has no consequences for the output assigned to \( /e/ \) (or to \( /i/ \)). Candidate \( /e/\lbrack i\rbrack \) has one candidate above it, \( /i/\lbrack i\rbrack \), so in an output-driven map \( /e/ \rightarrow [i] \) entails \( /i/ \rightarrow [i] \). Like \( /a/\lbrack a\rbrack \), candidate \( /i/\lbrack i\rbrack \) has no other candidates above it, so \( /i/ \rightarrow [i] \) has no entailments (apart from itself). The map in (2.2) is output-driven because all entailments are satisfied.

The candidate \( /a/\lbrack e\rbrack \) has one candidate above it: \( /e/\lbrack e\rbrack \). Mapping \( /a/ \rightarrow [e] \) entails \( /e/ \rightarrow [e] \) in an output-driven map. The map in (2.4) does not satisfy this entailment, and therefore is not output-driven.

The relative similarity relation in Figure 2.1 is determined solely by the nature of the candidates themselves. It is not dependent on any particular map. Thus, that same relative similarity relation is in equal force when evaluating all of the different maps in (2.1)–(2.5). This complete independence from any particular map on the representations makes it possible for the relative similarity relation to function as a standard for evaluating different maps defined over the same representational space.

### 2.2.2 Individuating disparities

Relative similarity assumes an inventory of disparities: the representational configurations that constitute individual instances of disparity.\(^2\) The preceding

---

2 There is a similarity of spirit between the representational concept of disparity as used here, and McCarthy’s (2007b) notion of basic faithfulness constraint violations, which are the presupposed basis for localized unfaithful mappings (LUMs) in OT with candidate chains (OT-CC).
discussion has implicitly assumed that disparities are differing values of individual features, differing between corresponding input and output segments, as well as individual segments of the input or output that lack IO correspondents. The inventory of disparities implied by those assumptions is part of the representational theory developed in Section 2.4, and will be used in the analyses of Chapter 3 and Chapter 4. It is worth pointing out, however, that the choice of an inventory of disparities can have a non-trivial effect on whether a map is judged to be output-driven or not.

For one candidate to have greater internal similarity than another, the corresponding disparities must be identical. This avoids notions of similarity in which one might speak of a single disparity in one candidate being “a lesser disparity” than its corresponding disparity in another candidate. If two corresponding disparities are not identical, then their correspondence cannot sustain a relationship of greater internal similarity between their respective candidates. The motivation for requiring corresponding disparities to be identical is discussed in Section 2.4.6.

The importance of the inventory of disparities can be illustrated by revisiting the situations presented in examples (2.1)–(2.5) using a different representational theory. Suppose segments are characterized atomically, without any reference to features. In this representational theory, [a], [e], and [i] are three distinct segments with no internal structure to refer to. In such a theory, the possible disparities involving corresponding input and output segments are quite limited: two segments are either identical, or they are not. The disparities involving corresponding elements are defined solely with reference to segments. Thus, with this inventory of disparities, the candidate /a/[i] has only one disparity, the corresponding non-identical segments a:i. The candidate /e/[i] also has one disparity, but it is not identical to the (single) disparity of /a/[i]. Disparity a:i is not the same as disparity e:i; the two disparities have different values on the input side.

Under this representational theory, maps (2.1)–(2.3) are output-driven, like before. Map (2.4) is non-output-driven, like before: /a/ maps to [e], and /e/ has greater similarity to [e] than /a/ has, yet /e/ does not map to [e]. Chain shifts are unavoidably non-output-driven, so long as a phonological form (like [e]) has greater similarity to itself than any other form has. However, map (2.5), repeated below, is output-driven under the new representational theory.

(2.5)  
/a/ \rightarrow [i]  
/e/ \rightarrow [e]  
/i/ \rightarrow [i]  

/a/[i] is grammatical, but /e/ does not have greater similarity to [i] than /a/ has; both are simply non-identical to [i]. Neither of the candidates /a/[i] and /e/[i] has
greater internal similarity than the other. One could consider the disparities $a:i$ and $e:i$ to be corresponding, as they involve the same output segment, but they are not identical disparities. The disparity $e:i$ of /e/[i] does not have an identical corresponding disparity in /a/[i]. Therefore, under the new representational theory, /a/ $\rightarrow$ [i] does not entail anything about the output for /e/ with respect to output drivenness.

The two schemes for relative similarity just described both require identical corresponding disparities. They differ on the inventories of disparities. The first approach individuates disparities between corresponding elements on the level of features, while the second approach individuates disparities between corresponding elements on the level of segments. It seems natural to expect that the inventory of disparities would derive directly from the linguistic representational theory being used. If features are included in linguistic representations, in particular if features play a linguistically efficacious role in the evaluation of input–output correspondences, then one would expect disparities to be individuated with respect to feature values (at least in part). If a linguistic theory includes input–output correspondences directly between autosegments, then one would expect that the inventory of disparities would include individual autosegments (independent of any segmental affiliation) with no IO correspondents.

2.2.3 Relative similarity is a relational notion
A key point must be emphasized: similarity here is not based on the number of disparities between two representations. Similarity here is not a numeric distance metric. It is instead a relational notion: B has greater similarity to C than A has to C if every disparity between B and C has a matching disparity between A and C. To change A to something with greater similarity to C, you can only remove disparities with respect to C; you cannot merely remove more disparities than you add.

Numeric distances for comparing structures, such as Levenshtein distance (Levenshtein 1966), are very useful when the differences between structures are thought to have arisen from independent randomly occurring events. If such events are rare occurrences, then the most plausible relation between the two structures is the one assuming the fewest such events (for further discussion, see Kruskal 1983: 36–40). Because the events are random, there is no expectation that the same structure would be altered in the same way on different occasions. The situation with output drivenness is exactly the opposite: disparities are predictable, repeatable consequences of the grammar. A given grammar will repeatedly assign the same output to an input, with the very same disparities.
Output drivenness is concerned not with numbers of disparities, but with consistency in the introduction of disparities. The idea is not that output restrictions provide an inventory of well-formed outputs, and an input is mapped to whichever of those outputs can be reached with the fewest disparities. Grammars can differ not only in the surface inventories they generate, but also in the disparities that are used to enforce output restrictions. Relative similarity defines relationships between candidates that warrant using the same disparity in the same way in both. If a given grammatical candidate introduces a certain disparity to help satisfy output restrictions, relative similarity determines the other inputs for which the grammatical candidates must employ the same disparity (and map to the same output), such that the introduction of the disparities in the map is consistent.

Recall the map in (2.2), repeated below.

(2.2) Output-driven: /a/ → [a] /e/ → [i] /i/ → [i]

Now consider larger forms containing those vowels (presume for the moment that there is no consonant–vowel interaction or any other kind of contextual conditioning for the vowels). The mappings in (2.9) and (2.10) would be predicted.

(2.9) /tapeke/ → [tapiki]
(2.10) /tepiki/ → [tipiki]

Of interest here is the relation between the two mappings, and the fact that they do not contradict the definition of output-driven map. Specifically, we want to see why it is possible for /tepiki/ to map to an output form other than [tapiki], despite the fact that [tapiki] is the optimal output form in (2.9), and /tepiki/ has fewer disparities with [tapiki] than the input in (2.9), /tapeke/, does. In other words, we need to explain the non-grammaticality of (2.11) in light of the grammaticality of /tapeke/[tapiki], the candidate of (2.9).

(2.11) * /tepiki/[tapiki]

In (2.9), the input differs from the output in the fourth and sixth segments, on the feature hi on each vowel. The disparities of (2.9) are given in (2.12). The notation uses subscripts to indicate which segment of a form is being referred to: $i_4$ means the fourth segment (counting from the beginning) of the input form, while $o_6$ means the sixth segment of the output form. The disparity $i_4(-hi):o_4(+hi)$ for (2.9) indicates that the fourth segment of the input is in correspondence with the fourth segment of the output, and the correspondents
disagree on the value of the feature hi, with the input correspondent having the value –hi and the output correspondent having the value +hi.

\[(2.12)\] Disparities of (2.9): \(i_4(-hi):o_4(+hi)\) \(i_6(-hi):o_6(+hi)\)

Candidate (2.11) has the same output as (2.9), but a different input. The input in (2.11) matches the output in the fourth and sixth segments, but differs in the second segment on the feature low. The disparities of (2.11) are given in (2.13).

\[(2.13)\] Disparities of (2.11): \(i_2(-low):o_2(+low)\)

Candidate (2.11) has fewer feature value disparities than (2.9); there are fewer disparities between /tepiki/ and [tapiki] than there are between /tapeke/ and [tapiki]. Yet, the input of (2.10) and (2.11), /tepiki/, does not have “greater similarity” to the output of (2.9), [tapiki], in the sense relevant here, because (2.11) has a disparity that (2.9) lacks: the differing value of low in the second segment. Thus, the fact that the output of (2.10) is not the same as the output of (2.9) does not violate the requirements of an output-driven map, because (2.9) does not entail (2.11).

The relational nature of relative similarity can be further illustrated by examining an input whose output is entailed in an output-driven map by (2.9). The mapping in (2.14) is entailed by (2.9) in an output-driven map.

\[(2.14)\] /tapeki/ \(\rightarrow\) [tapiki]

The disparities of (2.14) are given in (2.15).

\[(2.15)\] Disparities of (2.14): \(i_4(-hi):o_4(+hi)\)

The single disparity of (2.14) has a corresponding disparity in (2.9). The disparities correspond because they involve the same segment of the (shared) output, \(o_4\), and they are the same type of disparity, a mismatch in the feature hi with –hi in the input segment and +hi in the output segment. The input in (2.14) allows a simple reduction in the number of disparities with output [tapiki] relative to the input in (2.9). In an output-driven map, (2.9) entails (2.14).

The relational approach to similarity is different from conceptions of similarity that minimize the number of differences between two forms, such as Levenshtein distance (discussed earlier). It is also in principle different from perceptually based conceptions of similarity, such as the p-map hypothesis (Steriade 2001), where the significance of a disparity for similarity could vary significantly across contexts, depending on the perceptibility of the associated
contrast. However, a clear premonition of relative similarity can be found in work by Reiss (1996: 305–306), which proposes a concept of closeness between segments that is based on shared feature values, where one input segment is closer to an output segment than another input segment if the set of shared feature values between the former input segment and output segment contains (is a superset of) the set of shared feature values between the latter input segment and the output segment. Reiss’ closeness differs in significant respects from relative similarity: it is defined for single corresponding segments, not entire inputs and outputs, and is based on direct overlap of feature values rather than on correspondence between disparities, which would make it difficult to generalize closeness to entire inputs and outputs. But the relational nature is common to the two proposals.

This section has given the motivation for defining relative similarity in terms of correspondence between the disparities of two candidates sharing the same output (further technical discussion of how to establish such a correspondence is given later, in Section 2.4). Another form of correspondence, that between input and output elements within a candidate, is also crucial for output drivenness. Input–output correspondence is important because of the role it plays in the determination of the disparities of a candidate. The importance of input–output correspondence is explained next, in Section 2.2.4.

2.2.4 The importance of input–output correspondence
The discussion in Section 2.1.2 is simple to follow, because the input and output forms consist of a single segment, and the input segment of a candidate always corresponds to the output segment of that candidate. The discussion in Section 2.2.3 assumes that the input–output (IO) correspondence relation is an order-preserving bijection. A bijection is a relation that is 1-to-1 and onto, meaning that each input segment has exactly one output correspondent and each output segment has exactly one input correspondent (no insertion or deletion of segments). An order-preserving relation between input and output is one such that the precedence relation between any two input segments matches the precedence relation between the output correspondents of the two segments (no segmental metathesis). In an order-preserving bijection, the first segment of the input corresponds to the first segment of the output, the second segment in the input corresponds to the second segment in the output, and so forth.

For a given input–output pair, there can never be more than one possible order-preserving bijective correspondence relation, so if an order-preserving
bijection is presumed, then the content of that relation can be left implicit in the discussion. In general, however, it is desirable to allow more freedom of correspondence between inputs and outputs in candidates than is allowed by strict order-preserving bijective relations. Candidates with insertion and/or deletion of segments will not have bijective IO correspondence relations. In general, then, the IO correspondence relation must be more explicitly dealt with when characterizing similarity.

The fundamental importance of correspondence to similarity between sequences (and structured representations generally) has long been recognized in the sequence comparison literature (Levenshtein 1966, Sankoff and Kruskal 1983). While in linguistics the terminology of correspondence is perhaps found most explicitly in the OT literature, the concept is equally important to any generative theory. There is a correspondence relation implicit in every SPE-style rule. Any application of a rule has an input representation (before the rule applies) and an output representation (after the rule applies), and a correspondence relation is assumed between them. When a rule devoices an obstruent word-finally, the word-final obstruent of the output representation corresponds to the word-final obstruent of the input representation, and the other (unaltered) segments of the output correspond to the positionally analogous segments of the input. For a deletion rule, the targeted segment of the input has no correspondent in the output. An implicit correspondence holds between the underlying representation at the start of a derivation and the surface representation at the end of the derivation, via the composition of the correspondence relations for each of the rules that apply during the derivation.

Input–output correspondence is indispensable to linguistic theorizing in general. A candidate cannot be properly characterized solely by its input and output; the correspondence relation must be specified as well. This is illustrated here with an example that draws upon Optimality Theory, but corresponding examples could be constructed for any plausible generative theory. Consider an underlying form /ɡɑ/ paired with surface form [ʔɑ]. This same input–output pair can give rise to rather different conclusions about a grammar, depending upon what correspondence relation is assumed.

An IO correspondence relation $R_{IO}$ is here represented as a set of pairs, each pair consisting of an input segment and its IO-corresponding output segment. In (2.16), the first segment of the input $i_1 = /ɡ/$ corresponds to the first segment of the output $o_1 = [ʔ]$, and the second segment of the input $i_2 = /ɑ/$ corresponds to the second segment of the output $o_2 = [ɑ]$. In (2.17), the second segment of the input $i_2 = /ɑ/$ corresponds to the second segment of the output
\( o_2 = [\alpha] \), but the first segment of the input \( i_1 = /g/ \) has no output correspondent, and the first segment of the output \( o_1 = [?] \) has no input correspondent. The subscripted indices for the segments in the IO correspondence relation come from the precedence relations on the input and the output in each candidate.

\[
\text{(2.16)} \quad /ga[/?\alpha] \quad \mathbf{R}_{\text{IO}} = \{(i_1, o_1), (i_2, o_2)\}
\]

\[
\text{(2.17)} \quad /ga[/?\alpha] \quad \mathbf{R}_{\text{IO}} = \{(i_2, o_2)\}
\]

The candidate in (2.16), with input \( /g/ \) corresponding to output [?], has feature value disparities, such as a disparity in voicing (denoted \( i_1(+\text{voi}):o_1(–\text{voi}) \)). The candidate in (2.17), on the other hand, does not have any feature value disparities between IO correspondents, but does have a deletion disparity (denoted \( i_1_:\) ) and an insertion disparity (denoted \( _o_1: \)). In (2.17), the input \( /g/ \) has been deleted, and the output [?] has been inserted. In the general case, reasoning over a space of candidates requires reasoning not over input–output pairs alone, but over candidates resulting from the possible IO correspondence relations for each input–output pair.

Note that, in an SPE-style rule system, (2.16) and (2.17) require very different derivations using very different rules, the former requiring rules for feature change and the latter requiring rules for segmental deletion and insertion. The derivations do not differ in the input or the output; they differ in their implicit input–output correspondence relations. Derivations and input–output correspondence relations are mutually dependent. Similarly, in a conventional OT analysis, the candidate in (2.16) violates constraints evaluating feature identity between corresponding segments, such as \text{Ident}[F] \ (McCarthy and Prince 1995), while the candidate in (2.17) violates constraints against segmental deletion and insertion, such as \text{Max} and \text{Dep} \ (McCarthy and Prince 1995). Optimality for each of these candidates requires different ranking relations among such constraints.

To see why correspondence is important for relative similarity, consider the candidates in (2.18), (2.19) and (2.20). All three candidates have the same output form.

\[
\text{(2.18)} \quad /paka[/baga] \quad \mathbf{R}_{\text{IO}} = \{(i_1, o_1), (i_2, o_2), (i_3, o_3), (i_4, o_4)\}
\]

\[
\text{(2.19)} \quad /paka[/baga] \quad \mathbf{R}_{\text{IO}} = \{(i_2, o_2), (i_3, o_3), (i_4, o_4)\}
\]

\[
\text{(2.20)} \quad /aka[/baga] \quad \mathbf{R}_{\text{IO}} = \{(i_1, o_2), (i_2, o_3), (i_3, o_4)\}
\]

The disparities of the three candidates are listed in (2.21), (2.22), and (2.23).
(2.21) Disparities of (2.18): \( i_1(-\text{voi}):o_1(+\text{voi}) \) \( i_3(-\text{voi}):o_3(+\text{voi}) \)

(2.22) Disparities of (2.19): \( i_1:_o_1 \) \( i_3(-\text{voi}):o_3(+\text{voi}) \)

(2.23) Disparities of (2.20): \( _o_1 \) \( i_2(-\text{voi}):o_3(+\text{voi}) \)

First compare the candidates (2.18) and (2.19). In (2.18), there are two disparities, the disagreement in voicing between the correspondents \((i_1, o_1)\) and the disagreement in voicing between the correspondents \((i_3, o_3)\). In (2.19), there are three disparities: the disagreement in voicing between the correspondents \((i_3, o_3)\), input segment \(i_1 = /p/\) with no output correspondent, and output segment \(o_1 = [b]\) with no input correspondent. The inputs of the two candidates are identical, as are the outputs. Yet they are different mappings, because they have different IO correspondence relations.

Next compare (2.18) and (2.20), which have different inputs but the same output. In (2.20), there are two disparities: the disagreement in voicing between the correspondents \((i_2, o_3)\), and \(o_1 = [b]\) with no input correspondent. Each of (2.18) and (2.20) has a disparity that the other lacks: (2.18) has a disagreement in voicing between \(o_1\) and its input correspondent, while in (2.20) the output segment \(o_1\) has no input correspondent. Looked at in this fashion, neither of the inputs has greater similarity than the other to the shared output; each differs in a different way.

Now compare (2.19) and (2.20), which also have different inputs but the same output. In both candidates, segment \(o_1 = [b]\) has no input correspondent; those disparities are identical. In both candidates, segment \(o_3\) has a disparity in voicing with its input correspondent, the output value \(+\text{voi}\) disagreeing with the input value \(-\text{voi}\); those disparities are identical. Candidate (2.19) shares both of the disparities of (2.20), but also has an additional disparity: the input segment \(i_1 = /p/\) with no output correspondent. Looked at in this fashion, the input of (2.20) has greater similarity than the input of (2.19) to the shared output.

Clearly, the differing IO correspondences for (2.18) and (2.19) have non-trivial implications for relative similarity. Both (2.18) and (2.19) have the same input and the same output. Does /aka/ have “greater similarity” to [bɑkɑ] than /pɑkɑ/ has? You cannot answer that question as stated, because the necessary correspondence relations aren’t provided. A different sort of question needs to be asked, one that accounts for IO correspondence. Does (2.20) have a greater internal similarity than (2.19)? The answer is yes, because every disparity in (2.20) has an identical corresponding disparity in (2.19). Does (2.20) have a greater internal similarity than (2.18)? The answer is no, because (2.20) has a disparity, \(_o_1\), with no corresponding disparity in (2.18).
2.3 Output drivenness is not process opacity

2.3.1 One map, multiple generalizations

The commonly cited characterization of process opacity is the one given by Kiparsky (1971, 1973):

\[(2.24)\] A process \( P \) of the form \( A \rightarrow B / C__D \) is opaque to the extent that there are surface representations of the form:

\( a. \) \( A \) in the environment \( C__D \), or

\( b. \) \( B \) derived by \( P \) in environments other than \( C__D \).

Note that in this characterization, opacity is a property of processes in the context of derivations: a process is opaque to the extent that the process relates to derivations of the language in certain ways. Process opacity is not a property of a process in isolation; the definition refers to surface representations (outputs), so a process can only be evaluated for opacity in the context of a particular language. Further, \((2.24)b\) refers to “\( B \) derived by \( P \),” which cannot be identified solely on the basis of an output in correspondence with an input; the derivation of the output (from a particular input) has to be examined.

Process opacity is not a property of a phonological map; it is a property of the relationship between a process and the derivations that give rise to a phonological map. A map associates each input with a grammatical candidate; it makes reference only to inputs, outputs, and IO correspondence relations. Output drivenness characterizes phonological maps with reference to disparities, but without reference to an analysis in terms of processes; output drivenness is a property of phonological maps themselves.

The distinction between opacity and output drivenness remains when one considers the more general characterization of opacity cited in Chapter 1: a generalization which contributes meaningfully to an analysis, but which does not always hold of the outputs. “Generalization” is analogous to “process,” and “analysis” is analogous to “derivation.” One cannot determine the extent to which a generalization is opaque in the analysis of a language without a prior commitment to both the generalizations and the analysis. Under the view that OT markedness constraints are the relevant generalizations, opacity still is not.

IO correspondence is essential to the determination of the disparities in a candidate. It is therefore essential to the determination of correspondence between the disparities of two candidates, and to the determination of relative similarity.
a property of a map itself; it is a property of the relation between a markedness constraint and a map.

Some maps that have been analyzed in terms of opaque processes are also non-output-driven. Chain shifts are obvious cases. Recall the case of Etxarri Basque vowel raising in hiatus, introduced in Section 1.1.2. The key vowel mappings are shown in (2.25).

\[(2.25) \quad /e/ \rightarrow [i] \quad /i/ \rightarrow [i^\prime]\]

Candidate /e/-[i] is grammatical. Candidate /i/[i] has greater internal similarity than /e/[i], by virtue of removal of the disparity in the value of the feature hi, yet /i/[i] is not grammatical, so the map containing these mappings is not output-driven. Kirchner describes this in terms of an opaque raising process: the application condition of the process raising high vowels to high and raised is met by the output [i] in /e/[i], yet the process has not applied. In terms of (2.24), the process i → i^\prime / __V (or whatever the precise raising process is) is opaque in forms where [i] surfaces in the environment __V.

The map itself is non-output-driven, and the relevant raising process is opaque with respect to mapping /e/ → [i]. This is not surprising. The chain shift map violates a basic intuition underlying output drivenness, that a grammatical output should map to itself. The output [i] is grammatical, yet /i/ → [i^\prime]. The same property makes it almost inevitable that a process-based analysis using the same map will involve an opaque process. A process will be needed to achieve the mapping /i/ → [i^\prime], so /i/ will have to meet the conditions for application of that process, and thus the process will be opaque with respect to mapping /e/ → [i].

Nevertheless, output drivenness is fundamentally different from opacity. This can be made apparent by examining a case where the same phonological map can be plausibly analyzed in two different ways, with each analysis using a different phonological generalization. The key is that one of the generalizations is opaque relative to the map, while the other is not. This makes particularly clear that opacity is not a property of maps alone. Output drivenness, on the other hand, renders a single judgment on the phonological map, without reference to phonological generalizations like processes.

Recall from Section 1.2.1 the pattern of syllables being required to have onsets except for word-initial syllables, enforced via consonant epenthesis when necessary. This pattern could be attributed to (at least) two different processes.

\[3\] In an analysis where the mapping /i/ → [i^\prime] is achieved by the serial activity of several processes, the first process of the derivation series, at least, would be opaque in this sense.
One possibility is a process which targets non-initial syllables that lack onsets. The other possibility is a process which targets all syllables that lack onsets. The first possibility would be a transparent process for the described pattern: every non-initial syllable would have an onset, and no initial syllable would have an onset that had been inserted by this process. The second possibility would be an opaque process for the described pattern: initial syllables lacking onsets would constitute instances where the process did not apply even though the target environment was met. Yet the map itself remains the same for both cases. A given map is either output-driven or not, regardless of what processes one uses to describe it.

2.3.2 Epenthesis and assimilation in Lithuanian

A more complex case of process ambiguity occurs in Lithuanian, with a set of prefix alternations involving epenthesis and voice assimilation. I follow the presentation and discussion given by Baković 2007 (see also Baković 2005). All of the Lithuanian data used by Baković come from Ambrazas 1997, Dambriunas et al. 1966, Kenstowicz 1972, Kenstowicz and Kisseberth 1971, Mathiassen 1996.

The data concern two distinct verbal prefixes /at/ and /ap/. The consonants of the prefixes assimilate to adjacent stem-initial consonants in voicing and palatalization; I will focus on the voicing assimilation.

(2.26) Voicing assimilation in Lithuanian
at-ko:p'/:i ː to rise, climb up  ap-kal'b'et'i ː to slander
ad-gaut'i ː to get back  ab-gaut'i ː to deceive
at'/:p'/:aut'i ː to cut off  ap'/:em'/:iːt'i ː to obscure
ad'/:b'/:ek'/:ti ː to run up  ab'/:g'/:iːd'iːt'i ː to cure (to some extent)

If the prefix-final consonant and the initial consonant of the stem are the same apart from possible differences in voicing and palatalization, then the prefixes surface as [at'/:i] and [ap'/:i], respectively.4

(2.27) Vowel epenthesis in Lithuanian
at'/:i-taik'/:iːt'i ː to make fit well  ap'/:i-pu'/:i ː to grow rotten
at'/:i-teis't'/i ː to adjudicate  ap'/:i-p'/:iːl't'i ː to spill something on
at'/:i-duo't'/i ː to give back, return  ap'/:i-bar'/:iːt'i ː to scold a little bit
at'/:i-de'et'i ː to delay, postpone  ap'/:i-b'/:er'/:t'i ː to strew all over

4 The consonants palatalize before front vowels, including the epenthetic front vowels here (Baković 2007).
Baković cites an analysis by Odden 2005 which analyzes this in terms of two processes. One is an epenthesis process that epenthesizes vowels between adjacent obstruent stops with identical place of articulation. The other is a regressive voicing process in which an obstruent is voiced when it appears immediately before a voiced obstruent. Significantly, in this analysis the epenthesis process applies first, so that assimilation does not apply to obstruent stops with identical place.

Baković notes that the epenthesis process applies precisely in those situations where assimilation would have otherwise created adjacent identical obstruent stops. In line with this, he proposes an alternative epenthesis process, one that epenthesizes vowels between adjacent identical obstruent stops. This is a “process” in the sense that it is a conditioned change. If turned into a rule as stated, it won’t work in an SPE-style analysis: for obstruent stops with identical place but non-identical voicing, the rule would need to apply not to adjacent identical consonants but to consonants that would have become identical via assimilation (this is one of the main points made by Baković). SPE-style rules typically do not look ahead in that fashion. But such a restriction needn’t hold of processes in other theories. If one applies Kiparsky’s definition of opaque process to the Baković process, the process qualifies as opaque: there are instances of [i], derived by the epenthesis process, that appear in environments other than the one that conditions the process (between identical obstruent stops), that is, the instances of [i] appear between non-identical obstruents.

The present goal is not to comment on which analysis of the Lithuanian data is preferable (see Baković 2007 for further discussion). Instead, note that the Odden epenthesis process is not an opaque process in the analysis of the Lithuanian data: vowels epenthesized by the process only appear on the surface between consonants that satisfy the conditions of application for the process (identical place of articulation). The Baković epenthesis process, however, is opaque: it epenthesizes vowels between consonants that are not, on the surface, identical to each other.

The indeterminacy of process analysis for maps can be further highlighted by examining the Optimality Theoretic analysis, discussed by Baković, of the Lithuanian data. He notes that a ban on adjacent identical obstruent stops can be reified into a markedness constraint, and, in the context of appropriate other constraints, ranked properly, causes vowels to be epenthesized between consonants that would have otherwise become identical via assimilation. If one were to describe this in terms of phonological processes, the Baković process seems like the correct one to describe the OT analysis, and that process is
opaque. But, as discussed earlier, the status of such “processes” in Optimality Theory is questionable at best. If one focuses on markedness constraints as the potentially opaque generalizations, then the key constraint would be the one violated by adjacent identical obstruent stops. That constraint is never violated on the surface; it is not opaque in the sense discussed by McCarthy and Idsardi. But the markedness constraint itself says nothing about epenthesis at all; it is inadequate on its own to capture any generalizations about where epenthesized vowels do and don’t surface. The two takes on opacity in Optimality Theory give opposite conclusions about the opacity of the same analysis (the OT analysis of the Lithuanian data), and furthermore neither one seems like an adequate characterization of the OT analysis.

Output drivenness does not make reference to processes or constraints. It does require a specification of the mappings for all relevant inputs. The relevant mappings, based on the description of Lithuanian, are shown in (2.28)–(2.31). The mappings for two attested forms, based on the analysis above of the underlying form for the prefix, are given in (2.28) and (2.30). For each attested mapping of the original analysis, the greater internal similarity mappings are shown: these are inferred from the description of the language. The greater internal similarity mappings for (2.28) are given in (2.29), and the greater internal similarity mappings for (2.30) are given in (2.31). The map shown in (2.28)–(2.31) is output-driven.

(2.28)  Attested: \(/at\-t\-\text{\textit{eis/ti}/} \rightarrow at\text{\textit{i-t/eis/ti}}\) (2 disparities)

(2.29)  Greater similarity: \(/at\^i-t\-\text{\textit{eis/ti}/} \rightarrow at\text{\textit{i-t/eis/ti}}\) (1 disparity)
          \(/ati-t\-\text{\textit{eis/ti}/} \rightarrow at\text{\textit{i-t/eis/ti}}\) (1 disparity)
          \(/at\text{\textit{i-t/eis/ti}/} \rightarrow at\text{\textit{i-t/eis/ti}}\) (0 disparities)

(2.30)  Attested: \(/at-d\text{\textit{et/i}/} \rightarrow at\text{\textit{i-d/et/i}}\) (2 disparities)

(2.31)  Greater similarity: \(/at\^i-d\text{\textit{et/i}/} \rightarrow at\text{\textit{i-d/et/i}}\) (1 disparity)
          \(/ati-d\text{\textit{et/i}/} \rightarrow at\text{\textit{i-d/et/i}}\) (1 disparity)
          \(/at\text{\textit{i-d/et/i}/} \rightarrow at\text{\textit{i-d/et/i}}\) (0 disparities)

Some comments are in order. Output drivenness is a property of phonological maps, not just data forms. Thus, it is only independent of a choice between analyses to the extent that the differing analyses predict the same phonological map. Analyses that differ in their assignment of underlying forms to surface forms, or that differ in the posited IO correspondence relations, or that make different predictions about inputs not assigned to attested data, differ in the actual phonological maps they predict.
The output drivenness of a phonological map is determined not by attempting to individuate processes and look for individual conditioning environments in an output, but by looking to see if the systematic reduction (via changes to the input) of disparities between the input and output always yields an input mapped to that same output.

2.3.3 Closeness with processes
Reiss (1996) proposes a generalization regarding assimilation that is similar to output drivenness in some respects. The generalization refers to individual segments that change to other segments, rather than entire inputs and outputs, and is defined in terms of phonological processes. Reiss’ statement of the generalization is given in (2.32).

(2.32) Target–Output Closeness (Reiss 1996: 306): Suppose that in a language \( L \) there is a phonological process (a rule or set of rules) \( P \), by which segment \( x \) becomes \( z \). If a segment \( y \) is closer to \( z \) than \( x \) is, \( y \) will also be a target of \( P \) in \( L \) and also become \( x \).

Reiss defines the closeness of two segments as the set of feature values that the two segments share, so that \( y \) can only be closer to \( z \) than \( x \) if \( y \) agrees with \( z \) on every feature that \( x \) agrees with \( z \) on (see Section 2.2.3). Later, Reiss offers a restatement of Target–Output Closeness (TOC), shown in (2.33).

(2.33) The TOC in OT terms (Reiss 1996: 312): Suppose that three segments \( x \), \( y \) and \( z \) stand in the following closeness relationship: \( y \cap z \supset x \cap z \) (\( y \) is closer to \( z \) than \( x \) is). If \( z \) is the optimal surface form of \( x \), then \( z \) is also the optimal surface form of \( y \) in a given context.

With respect to individual segments and identity disparities, the statement in (2.33) sounds like a “shared feature value” rendition of the “shared disparities” of output drivenness, with superset of shared feature values playing the role of subset of disparities. However, Reiss draws some conclusions about phenomena with respect to TOC that are radically different from conclusions we have already demonstrated for output drivenness.

Reiss discusses segmental chain shifts in assimilation, describing them as an instance of process counterfeeding. He describes a phenomenon of “stepwise”
assimilation in the Italian dialect of Servigliano (Camilli 1929), where underlying \( p \) surfaces as \( b \) after a nasal (assimilation in voicing), and underlying \( b \) surfaces as a nasal after a nasal (assimilation in nasality). He then notes that the conjoined constraints of the sort proposed by Kirchner (Reiss refers to them as “disjunctive constraints”) are capable of deriving such chain shifts in OT analyses (see Sections 1.1.2 and 2.3.1). Reiss then states that “Fortunately, such disjunctive constraints appear not lead [sic] to TOC violations” (Reiss 1996: 316). He is asserting that chain shifts (at least of this sort) do not violate the principle of Target–Output Closeness.

It isn’t clear how to arrive at the conclusion that chain shifts do not violate the TOC. If \( p \rightarrow b \), then the optimal surface form of \( p \) is \( b \), and underlying \( b \) clearly has greater overlap in feature values with surface \( b \) than underlying \( p \) does, so \( b \) surfacing as anything other than \( b \) is a violation of the TOC in OT terms: \( p \rightarrow b \rightarrow m \) violates the statement in (2.33). The only apparent way to arrive at a different conclusion is to define closeness so as to exclude identical segments: to stipulate that underlying \( b \) is not closer to surface \( b \) than anything else. In any event, if in fact chain shifts do not violate the TOC, then the TOC is clearly a very different property from anything resembling surface orientedness; chain shifts are canonical examples of phenomena that are not surface oriented.

The original statement of the TOC in (2.32) more explicitly defines things in terms of processes: if a process targets one segment, then the same process targets other segments that are closer to the same output. Defining things in terms of processes brings further problems. The question is raised: can a process be claimed to target a segment that it would not alter in any way? This would appear to be a matter of how a process is defined: a process that targets voiceless labials and voices them does not target labials that are already voiced, while a process that targets labials and voices them does target labials that are already voiced. Thus, the satisfaction/violation of the TOC can vary, depending on subtle details of how the processes are defined, even if the different processes themselves produce the same outputs for the same inputs. If the process yielding \( p \rightarrow b \) targets voiceless labials, then it violates the TOC because it does not target underlying \( b \) (no matter how underlying \( b \) surfaces). If the process yielding \( p \rightarrow b \) targets labials, then it satisfies the TOC by targeting \( b \), provided that \( b \rightarrow b \). As with process opacity, the same map could yield opposite conclusions for the TOC, depending on what processes are assumed.

This is another demonstration of the difficulty and equivocation that can arise when attempting to reason about surface-oriented properties using processes. The solution offered by output drivenness is to dispense with processes altogether and reason purely in terms of the maps themselves.
2.4 Formal analysis with segmental IO correspondence

This section lays out a more detailed formal development of the theory of output-driven maps. After giving a general formal definition in Section 2.4.1, the rest of Section 2.4 works through a case with more specific representational assumptions: IO correspondence is defined only over segments, a segment cannot have more than one IO correspondent, and IO correspondence is order-preserving. This case study provides a more concrete basis for presentation of some of the linguistic motivations for and consequences of output drivenness.

2.4.1 Maps from inputs to candidates

A candidate is an input, an output, and an input–output (IO) correspondence relation between the input and the output. A candidate with input form $\text{in}_a$, IO correspondence relation $\text{R}_k$, and output form $\text{out}_x$ will be denoted with the term $\text{ak}_x$. The candidate label includes a separate index ($k$) for the IO correspondence relation to explicitly recognize that the same input–output pair can have more than one IO correspondence relation defined on it (each relation defining a different candidate). When the correspondence relation is assumed to be an order-preserving bijection, candidates will sometimes be denoted with only the input and output: $\text{in}_a[\text{out}_x]$.

The notation $\text{ak}_x$ for a candidate is quite terse, expressing a complex object purely in terms of identifying subscripts for each of its three main parts. This will prove useful in keeping later discussion readable. A convention that will be observed in the rest of this book is that when a relation of greater internal similarity is being discussed, the label $\text{ak}_x$ will be used for the candidate with lesser internal similarity, and the label $\text{bm}_x$ will be used for the candidate with greater similarity. The two share the output form $\text{out}_x$, and the input of $\text{bm}_x$, $\text{in}_b$, has greater similarity to $\text{out}_x$ than $\text{in}_a$, the input of $\text{ak}_x$. Each candidate has its own IO correspondence relation. This convention makes the discussion in the rest of the book easier to follow.

For the purposes of evaluating maps, some reference set of candidates must be used; it defines the linguistic representations under consideration for the analysis (the representations that the theory in use permits). The reference set of candidates is here labeled the reference representation space (RRS). The input space consists of those input forms that appear in at least one candidate in the RRS, while the output space consists of those output forms that appear in at least one candidate in the RRS.
A map $M$ is a function from individual inputs to sets of candidates.\(^6\) A mapping is a pairing of an input with one of the candidates in the set that the input is mapped to. The domain of $M$ is the input space. The codomain of $M$ is the power set of the RRS (the possible subsets of the RRS). An important restriction on $M$ is that the set of candidates assigned to an input by $M$ may only contain candidates which include that input.\(^7\) $M(in_a)$ cannot contain a candidate with input $in_b$ if $in_a \neq in_b$. Any candidate that is in a set assigned by $M$ to some input is labeled a grammatical candidate. That is, $akx$ is grammatical if and only if $akx \in M(in_a)$.\(^8\)

Output drivenness involves a comparison between candidates with identical output forms. Formally, what is required is a relation on the RRS, relative similarity, containing pairs of candidates with identical output forms, where each pair is interpreted as meaning that the first candidate of the pair has greater internal similarity than the second candidate of the pair. The term internal similarity signifies that the similarity between the input and output of a candidate is being referred to (similarity is “internal” to a candidate). Relative similarity is the comparison of the internal similarity of one candidate to the internal similarity of another.

Given a reference representation space RRS and a relative similarity relation RSIM defined over that RRS, an output-driven map is defined as follows. $M$ is an output-driven map if, for all candidates, the grammaticality of $akx$ entails the grammaticality of every candidate $bmx$ in the RRS with greater internal similarity than $akx$. Letting $M(in_a)$ represent the set of candidates that are assigned by map $M$ to input $in_a$, the condition of output drivenness for a map $M$ can be expressed more formally as shown in (2.34). The expression $akx \in M(in_a)$ indicates that $akx$ is grammatical, while the expression $(bmx,akx) \in RSIM$ indicates that $bmx$ has greater internal similarity than $akx$.

---

\(^6\) While it might seem natural to formalize the map as a function from inputs to candidates, for the sake of generality, each input is mapped to a set of candidates, thus allowing for the possibility that an input has more than one output (or none at all). Maps for which each input has exactly one output are a special case of primary interest, where each input is mapped to a set containing exactly one candidate.

\(^7\) Here, a candidate “includes” an input in the sense that the input is a part of the full candidate, not in the sense that elements of the input are directly contained in the output. This is reflected in the analysis of Optimality Theory pursued in Chapter 3, which assumes a correspondence theory of faithfulness (McCarthy and Prince 1995), not a containment-based theory of faithfulness (Prince and Smolensky 1993/2004).

\(^8\) The term “grammatical” is here used for convenience, given that maps defined by grammars will be of central interest. The term “well-formed” might seem more neutral, but the use of “grammatical” in this sense is so widespread that it will be continued here.
Output-driven maps

(2.34) \( \forall akx \in \text{RRS}, \forall bmx \in \text{RRS} [(akx \in \text{M}(in_a) \& (bmx, akx) \in \text{RSIM}) \rightarrow (bmx \in \text{M}(in_b))] \)

Given the formulation in (2.34), with the map formally yielding sets of candidates, a map like (2.1) might be written as shown in (2.35).

(2.35)  

To simplify the presentation (and match standard notation), mappings will commonly be denoted with an arrow from an input to the output of the single grammatical candidate for that input, as is done in (2.1).

The definition of output-driven maps given in (2.34) can apply with a variety of reference representation spaces and relative similarity relations. It requires only that the relative similarity relation be a partial ordering of the RRS such that only pairs of candidates with the same output can be related. The grammaticality entailment relations between candidates related by relative similarity is the abstract essence of output drivenness.

However, more contact can be made with linguistic theory if more specific commitments are made about the representations and the relative similarity relation on them. In the rest of this section, the RRS is taken to consist of candidates such that inputs and outputs are based on strings of segments, and segments in turn are characterized by valued features. IO correspondence is taken to be exclusively a relation between segments (that is, there are no independent IO correspondence relations for other levels of representation, such as features or prosodic categories). Relative similarity is based on a disparity inventory of segment deletions, segment insertions, and feature value mismatches between IO-corresponding segments. These representational assumptions are basic, but familiar, and lend themselves well to analysis. They will be retained throughout the rest of this book. However, there is clearly room for application of the general definition of output-driven maps to theories with differing representational assumptions, and this section gives an indication of the issues that must be addressed.

2.4.2 The internal structure of candidates
Recall that the reference representation space (RRS) contains the candidates that will be referred to in the evaluation of maps. Input–output correspondence is here assumed to be segment-based: the input–output correspondence relation for a candidate relates input segments to output segments, following McCarthy and Prince (1995); for an alternative view, see Lombardi (2001). Nothing here conflicts with prosodic structure in the output, so long as (non-segmental)
prosodic elements do not stand in input–output correspondence. The assumed conditions on the RRS are summarized in (2.36).

(2.36) Conditions on the reference representation space
- Inputs and outputs each contain a string of segments.
- Segments are characterized by features.
- Candidates have only a single correspondence relation between the input and the output, one that relates only segments (input and output segments can stand in correspondence, but no other representational elements can).
- Any type of segment (characterized in terms of its feature values) in the input can correspond to any type of segment in the output.
- Segments (both input and output) can have at most one correspondent.
- For any pair of output segments with input correspondents, the precedence (temporal order) of the segments in the output must match the precedence of their input correspondents.

The conditions in (2.36) are not intended to be the final word on representational theory; they are intended to be a good starting point for investigating output-driven maps. Some of the additional complexities involved in relaxing these conditions are discussed in Section 2.5.2.

The conditions in (2.36) limit the classes of disparities that can occur in candidates. The classes of disparities are given in (2.37).

(2.37) The classes of disparities
- Deletion: an input segment with no output correspondent.
- Insertion: an output segment with no input correspondent.
- Identity: a difference in the values of a feature for corresponding input and output segments.

The inventory of disparities consists of the possible disparities of any of these classes. There will be one deletion disparity in the inventory for each possible segment. A segment /a/ without a correspondent is a distinct (non-identical) disparity from a segment /e/ without a correspondent, although both are of

9 There is no conflict with prosodic structure in the input, either, provided the same restriction on correspondence holds; however, some complications could arise. If the grammar does not make reference to prosodic structure in the input, then such structure is completely inert and has no impact on the grammar. If the grammar does make reference to such input structure, then it could distinguish candidates with identical corresponding disparities (the candidates would be distinguished by the input structure that does not stand in IO correspondence and therefore cannot constitute disparities). This is not necessarily a problem, but it would mean that the relative similarity relation would no longer be a partial order, because it would not be antisymmetric: two non-identical candidates could each have greater internal similarity than the other. This isn’t an issue for prosodic structure in the output, because candidates with distinct outputs cannot be related in terms of relative similarity.
the same type with respect to (2.37). Analogously, there will be one insertion disparity in the inventory for each possible segment. There will be one identity disparity in the inventory for each possible combination of non-identical input and output values for a feature. The input/output distinction matters: the identity disparity \(-hi:+hi\) (input correspondent is \(-hi\), while the output correspondent is \(+hi\)) is not identical to the identity disparity \(+hi:-hi\); they are distinct members of the inventory of disparities. The complete inventory of disparities for the three vowel hi/low system is given in (2.38).

\[
(2.38) \quad \text{The inventory of disparities for the three vowel example}
\]

<table>
<thead>
<tr>
<th>Deletion</th>
<th>Insertion</th>
<th>Identity</th>
</tr>
</thead>
<tbody>
<tr>
<td>a: _</td>
<td>_ :a</td>
<td>(-hi:+hi)</td>
</tr>
<tr>
<td>e: _</td>
<td>_ :e</td>
<td>(+hi:-hi)</td>
</tr>
<tr>
<td>i: _</td>
<td>_ :i</td>
<td>(-low:+low)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(+low:-low)</td>
</tr>
</tbody>
</table>

2.4.3 Relating candidates to each other

Suppose we are given a pair of candidates \(bmx\) and \(akx\), with identical output form \(out_x\), and input forms \(in_b\) and \(in_a\) respectively. An analysis of the relative similarity between \(akx\) and \(bmx\) requires that a correspondence be established between the disparities of the two candidates. In order to relate the disparities of the candidates, we have to relate the segments of the candidates. The anchor of the relationship between the two candidates is the shared output form. Since the two candidates have identical output forms, it is easy to relate the output segments of the candidates: each segment of the output in one candidate relates to “itself” in the other candidate.\(^{10}\)

Relating the disparities of the two candidates involves comparing the fates of corresponding segments in the two candidates. How an output segment is handled in one candidate will be directly related to how that same output segment is handled in the other candidate. If an output segment is inserted in one candidate, say \(bmx\), that constitutes an insertion disparity in \(bmx\). That insertion disparity will have a corresponding disparity in \(akx\) if and only if the same output segment is also inserted in \(akx\). The two insertion disparities can correspond only if they affect the same output segment.

Deletion disparities concern input segments, and identity disparities concern both input and output segments. Comparing these kinds of disparities between

\(^{10}\) Technically, an order-preserving bijective correspondence is assumed between the identical outputs of the two candidates, matching the first segments to each other, and so forth. In other words, there are two identical outputs, one for each candidate, and the obvious correspondence between them is assumed. When comparing candidates with identical outputs, I will write as if there were a single output, to cut down on the wordiness of the discussion.
candidates involves creating a correspondence relation between the inputs of the candidates. The **input–input correspondence relation**, which will be denoted $R_{II}$, is not a part of a grammar, nor is it a part of any candidate; it is a structure constructed as part of an analysis of output drivenness. In general, the input forms of the candidates will not be identical, so defining the correspondence between the input forms is not as trivial as defining the one between the (identical) output forms. Relative similarity is intrinsically concerned with how two different candidates relate to the same output form, so the input–input correspondence is based upon the input–output relations between each of the input forms and the output form.

Segments of the respective inputs can only be input–input correspondents if they relate to the output form in the same way. A pair of input segments can be input–input correspondents only if they either (a) both IO-correspond to the same output segment, or (b) both lack output correspondents. The input–input correspondence is important to the establishment of a correspondence between disparities: a pair of disparities from the two candidates can correspond only if the disparities involve segments that correspond between the two candidates. Deletion disparities can only correspond if the respective deleted input segments are input–input correspondents. Identity disparities can only correspond if they involve corresponding input segments (input–input correspondents) and corresponding output segments (an identical segment of the shared output form).

The input–input correspondence $R_{II}$ between $in_b$ and $in_a$ must satisfy the conditions listed in (2.39). For instance, given segment $s_b$ of $in_b$ with output correspondent $s_x$ of $out_x$ in $bmx$ (that is, $s_b R_{m} s_x$), $s_b$ has II correspondent $s_a$ in $in_a$ if and only if $s_x$ has input correspondent $s_a$ in $akx$ (that is, $s_a R_{k} s_x$). If $s_x$ has no input correspondent in $akx$, then $s_b$ has no II correspondent in $in_a$.

(2.39) **Conditions on the input–input (II) correspondence relation $R_{II}$**

a. As with the IO correspondence, II correspondence must be order-preserving: for any pair of segments in an input with II correspondents, the precedence of those segments in one input must match the precedence of their II correspondents in the other input.

b. As with IO correspondence, segments can have at most one II correspondent.

c. An input segment with an output correspondent in one candidate has an II correspondent if and only if that output segment has an input correspondent in the other candidate (the latter input segment then is the II correspondent):

i. if $s_b R_{m} s_x$, then ($s_a R_{II} s_b$ if and only if $s_a R_{k} s_x$).

ii. if $s_a R_{k} s_x$, then ($s_a R_{II} s_b$ if and only if $s_b R_{m} s_x$).
Given an input–input correspondence satisfying the conditions in (2.39), a correspondence between the disparities of the two candidates can be constructed as described in (2.40).

(2.40) Constructing a correspondence between the disparities of \( bmx \) and \( akx \), given input–input correspondence \( R_{II} \)
- Let \( s_b :_1 \) be a deletion disparity in \( bmx \). This disparity has a corresponding disparity \( s_a :_1 \) in \( akx \) if and only if \( s_b \) has input–input correspondent \( s_a \) in \( akx \) (and thus \( s_a \) necessarily has no output correspondent in \( akx \), by the conditions on input–input correspondence).
- Let \( :_s \) be an insertion disparity in \( bmx \). This disparity has a corresponding disparity \( :_s \) in \( akx \) if and only if \( s \) has no input correspondent in \( akx \).
- Let \( s_x \) be an output segment of \( bmx \) with an input correspondent \( s_b \) such that \( s_b \) and \( s_x \) differ on the value of feature F. This disparity in \( bmx \) has a corresponding disparity in \( akx \) if and only if \( s_x \) has an input correspondent \( s_a \) in \( akx \) such that \( s_a \) and \( s_x \) differ on the value of feature F (\( s_b \) and \( s_a \) are then necessarily input–input correspondents by the conditions on input–input correspondence).

These constructions are illustrated in the following example. Again, I will only be concerned with the segmental features hi and low. In order to adequately describe the example, some further notational conventions need to be introduced. For a string of segments denoted by form index \( x \), denote the first segment as \( x_1 \), the second segment as \( x_2 \), etc. Thus, if output form \( out_x = [tibi] \), denote the segments of \( out_x \) as \( x_1 = t \), \( x_2 = i \), \( x_3 = b \), \( x_4 = i \). This indexing of the segments is useful for describing the various correspondence relations that hold between different forms.

Let \( akx \) be a grammatical candidate with the components listed in (2.41). Two of the input segments, \( a_2 \) and \( a_5 \), have no output correspondents (constituting deletion disparities). One of the output segments, \( x_1 \), has no input correspondent (constituting an insertion disparity).

(2.41) \( in_a = /agbik/ \quad out_x = [tibi] \quad R_k = \{(a_1,x_2), (a_3,x_3), (a_4,x_4)\} \)

Let \( bmx \) be the candidate with the components listed in (2.42). One of the input segments, \( b_5 \), has no output correspondent. All of the output segments have input correspondents.

(2.42) \( in_b = /tebik/ \quad out_x = [tibi] \quad R_m = \{(b_1,x_1), (b_2,x_2), (b_3,x_3), (b_4,x_4)\} \)

Figure 2.2 shows the candidates, with their IO correspondences indicated by lines. At the bottom, the input–input correspondence between \( akx \) and \( bmx \), described next, is also shown.

The input–input correspondence \( R_{II} \) between \( in_a \) and \( in_b \) is shown in (2.43).
2.4 Formal analysis with segmental IO correspondence

\[ R_{II} = \{(a_1, b_2), (a_3, b_3), (a_4, b_4), (a_5, b_5)\} \]

The first three pairs of \( R_{II} \) are based on common output correspondents: \( a_1 \) and \( b_2 \) both have output correspondent \( x_2 \), \( a_3 \) and \( b_3 \) both have output correspondent \( x_3 \), and \( a_4 \) and \( b_4 \) both have output correspondent \( x_4 \). The last pair of \( R_{II} \), \( (a_5, b_5) \), is possible because neither segment has an output correspondent and their correspondence doesn’t violate the order preservation requirement of (2.39)a. \( a_4 \) precedes \( a_5 \) in \( in_a \), and the respective II correspondents are similarly ordered, with \( b_4 \) preceding \( b_5 \) in \( in_b \). The same precedence requirement blocks the possibility of \( a_2 \) replacing \( a_5 \) as the II correspondent of \( b_5 \), even though neither has an output correspondent: \( a_3 \) (necessarily) has II correspondent \( b_3 \), and \( a_2 \) precedes \( a_3 \) while \( b_5 \) follows \( b_3 \).

Let \( a_1: \_ \) denote the disparity in which input segment \( a_1 \) has no output correspondent. Let \( \_ : x_1 \) denote the disparity in which output segment \( x_1 \) has no input correspondent. Let \( a_1(-hi):x_2(+hi) \) denote the disparity in which input segment \( a_1 \) has the feature value \(-hi\) and its output correspondent \( x_2 \) has the feature value \(+hi\).

Candidate \( akx \), as described in (2.41), has the disparities listed in (2.44).

\[ \text{(2.44)} \quad \_ : x_1 \quad a_1(-hi):x_2(+hi) \quad a_1(+low):x_2(-low) \quad a_2:\_ \quad a_5:\_ \]

Candidate \( bmx \), as described in (2.42), has the disparities listed in (2.45).

\[ \text{(2.45)} \quad b_2(-hi):x_2(+hi) \quad b_5:\_ \]

Candidate \( akx \) has five disparities, while \( bmx \) has two. Following (2.40), a correspondence between the disparities of the two candidates can now be constructed, given in (2.46).

\[ \text{(2.46)} \quad \text{A correspondence between the disparities of } bmx \text{ and } akx \]

- \( b_2(-hi):x_2(+hi) \) of \( bmx \) corresponds to \( a_1(-hi):x_2(+hi) \) of \( akx \)
- \( b_5:\_ \) of \( bmx \) corresponds to \( a_5:\_ \) of \( akx \)
Disparity \(a_1(-hi)\cdot x_2(+hi)\) of \(akx\) corresponds to \(b_2(-hi)\cdot x_2(+hi)\) of \(bmx\), because \(a_1\) and \(b_2\) are input–input correspondents \((a_1RIIb_2)\), and the disparities are identity disparities involving the feature \(-hi\). Disparity \(a_5\cdot _\_\) of \(akx\) corresponds to \(b_5\cdot _\_\) of \(bmx\), because \(a_5\) and \(b_5\) are input–input correspondents \((a_5RIIb_5)\), and both disparities are deletion disparities.

To support a claim of greater internal similarity, corresponding disparities must be “identical”: the disparities must be instances of the same disparity in the inventory of disparities. Consider two identity disparities involving input segments that disagree in the feature \(F\) with their (common) output correspondent. If feature \(F\) has more than two possible values, it is not sufficient that the pair of disparities each be such that the feature value on the input side is different from the value on the output side; the two disparities must have the same value for \(F\) on the input side in order to be identical (the disparities have no choice but to have the same value for \(F\) in the output segment, because by definition it is the same output segment for both disparities). Similarly, two disparities consisting of input segments with no output correspondents are only identical if the input segments have the same values for all features, that is, they are identical segments (they are the same segment type).

The conditions on input–input correspondence in (2.39) are motivated by the role they play in defining correspondence between disparities. (2.39)c reflects the fact that the primary basis for correspondence between the two candidates is their shared output form. If a segment in \(in_a\) and a segment in \(in_b\) both have the same output segment as their output correspondents, they should be input–input correspondents; they are playing corresponding roles in the two candidates by virtue of the fact that they correspond to the output in the same way. An identity disparity involving feature \(F\) for one of the input segments should correspond to an identity disparity involving feature \(F\) for the other input segment (if such a disparity exists in the other candidate). Making the input segments input–input correspondents accords with the correspondence between the disparities sharing the same output segment. It also follows from that condition that input segments with no output correspondents may only have input–input correspondents also lacking output correspondents: the segments then have the same relation to their respective outputs (lack of output correspondents) and constitute the same sort of disparity. Input–input correspondence between segments lacking output correspondents supports a correspondence between deletion disparities in the two candidates.

(2.39)a and (2.39)b simply keep input–input correspondence in accord with the conditions assumed on input–output correspondence; the correspondences should be order-preserving, and segments should have at most one
2.4 Formal analysis with segmental IO correspondence

2.4.4 The non-uniqueness of input–input correspondence

It is worth noting that the conditions on input–input correspondence do not always uniquely determine an input–input correspondence for a pair of candidates sharing the same output. The indeterminacy involves input–input correspondence between segments lacking output correspondents. While the conditions limit the input–input correspondence of segments lacking output correspondents to other segments lacking output correspondents, it does not oblige such segments to have input–input correspondents, even if a possible input–input correspondent is available. Of course, if each of a pair of candidates has an input segment lacking an output correspondent and lacking an input–input correspondent, then each candidate has a disparity with no corresponding disparity in the other candidate, and neither candidate can have greater internal similarity than the other with respect to the selected input–input correspondence. In order to sustain a “greater internal similarity” relation between two candidates, the selected input–input correspondence must be such that all input segments of the greater internal similarity candidate lacking output correspondents must have input–input correspondents, so that all of the deletion disparities of the greater internal similarity candidate have corresponding disparities in the lesser internal similarity candidate.

Under certain circumstances, more than one input–input correspondence can sustain a “greater internal similarity” relation for the same pair of candidates. This can be illustrated by slightly altering the illustration of (2.41)–(2.45). Let \(ak\) and \(bm\) be as shown in (2.47) and (2.48).

\[
(2.47) \quad ini = /agbikk/ \quad outi = [tibi] \quad R_k = \{(a_1, x_2), (a_2, x_3), (a_4, x_4)\}
\]

\[
(2.48) \quad inb = /tebik/ \quad outb = [tibi] \quad R_m = \{(b_1, x_1), (b_2, x_2), (b_3, x_3), (b_4, x_4)\}
\]

Candidate \(ak\), as described in (2.47), has the disparities listed in (2.49).

\[
(2.49) \quad \_\_ x_1 \quad a_1(-hi):x_2(+hi) \quad a_3(+low):x_3(-low) \quad a_2: \_ \quad a_5: \_ \quad a_6: \_ \]

Candidate \(bm\), as described in (2.48), has the disparities listed in (2.50).

\[
(2.50) \quad b_2(-hi):x_2(+hi) \quad b_5: \_
\]

The only difference is the additional segment /k/ at the end of \(ini\), denoted \(a_6\). Importantly, \(a_6\) has no output correspondent and is identical to the preceding
segment, which also has no output correspondent. This creates the opportunity for two equivalent but non-identical input–input correspondences. The final segment of \( \mathit{in}_b, \mathit{b}_5 \), can have either \( \mathit{a}_5 \) or \( \mathit{a}_6 \) as an input–input correspondent. Either way, the disparity \( \mathit{b}_5:\_ \) in \( \mathit{bm}_x \) (a /k/ with no output correspondent) has an identical corresponding disparity in \( \mathit{ak}_x \): it corresponds to either \( \mathit{a}_5:\_ \) or \( \mathit{a}_6:\_ \). The corresponding disparity is determined by the II correspondence: if \( \mathit{b}_5\mathit{R}_{II}\mathit{a}_5 \) then disparity \( \mathit{b}_5:\_ \) corresponds to disparity \( \mathit{a}_5:\_ \). If \( \mathit{b}_5\mathit{R}_{II}\mathit{a}_6 \) then \( \mathit{b}_5:\_ \) corresponds to \( \mathit{a}_6:\_ \).

Because of the preceding observations, the definition of greater internal similarity in (2.51) is expressed in terms of the existence of an appropriate correspondence between disparities, rather than in terms of a uniquely defined correspondence between disparities.

(2.51)  Candidate \( \mathit{bm}_x \) has greater internal similarity than \( \mathit{ak}_x \) if there exists a correspondence between the disparities of the two candidates, satisfying the conditions in (2.40), such that every disparity in \( \mathit{bm}_x \) has an identical corresponding disparity in \( \mathit{ak}_x \).

For the candidates \( \mathit{ak}_x \) and \( \mathit{bm}_x \) given in (2.41) and (2.42), the input–input correspondence in (2.43) yields the correspondence in (2.46) between the disparities of the candidates. Candidate \( \mathit{bm}_x \) has greater internal similarity than \( \mathit{ak}_x \) because both of the disparities for \( \mathit{bm}_x \) have corresponding disparities for \( \mathit{ak}_x \), as described in (2.46), and in both instances the corresponding disparities are identical.

2.4.5  Removing disparities by changing the input

Because the pairs of candidates being compared in the definition of output-driven maps have identical outputs, they can differ only in their inputs and in their input–output correspondence relations. Intuitively, \( \mathit{bm}_x \) has greater internal similarity than \( \mathit{ak}_x \) if it is possible to get the input form for \( \mathit{bm}_x \) by taking the input form for \( \mathit{ak}_x \) and changing it in ways that give it greater similarity to the output form.\(^\text{11}\) Because similarity is characterized in terms of corresponding disparities, changing the input must have the effect of eliminating

\(^{11}\) In principle, it is possible for two candidates with the same input form to stand in a greater internal similarity relationship. This requires that the greater-similarity candidate have a pair of identical IO corresponding segments that both lack IO correspondents in the lesser-similarity candidate: the greater-similarity candidate lacks both an insertion disparity and a deletion disparity relative to the lesser-similarity candidate. While technically all that is different between the two candidates is the IO correspondence relation, the “input-changing” metaphor could be sustained in terms of deleting the input segment lacking an output correspondent, and then inserting an identical input segment in IO correspondence with the output segment.
disparities without introducing any new ones. Thus, we can itemize the kinds of “changes” that can result in greater similarity by identification with the disparities that those changes eliminate. Removing an input segment that has no output correspondent eliminates a deletion disparity. Inserting an input segment such that it corresponds to an output segment previously lacking an input correspondent eliminates an insertion disparity, and if the added input segment is identical to its output correspondent then no new identity disparities are introduced. If the value for a feature of an input segment is changed to match the value of its output correspondent, that eliminates an identity disparity.

2.4.6 The identical disparity requirement and surface orientedness
The definition of greater internal similarity given in (2.51) has some consequences that may not be immediately obvious, but are important. They are given in (2.52), (2.53), and (2.54).

(2.52) If an output segment \( s_x \) has an input correspondent \( s_b \) in \( bmx \) but not in \( akx \), then \( bmx \) can have greater internal similarity than \( akx \) only if \( s_b \) and \( s_x \) are identical.

If \( s_b \) and \( s_x \) aren’t identical, then the feature value mismatches distinguishing the segments constitute identity disparities that have no corresponding disparities in \( akx \).

(2.53) If \( bmx \) has greater internal similarity than \( akx \), then every segment in \( in_a \) with an output correspondent has an input–input correspondent in \( in_b \).

If segment \( s_a \) of \( in_a \) had an output correspondent \( s_x \) in \( out_x \), but no input–input correspondent, then \( s_x \) would have no input correspondent in \( bmx \), which would constitute a disparity in \( bmx \) with no correspondent in \( akx \), contradicting the premise that \( bmx \) has greater internal similarity than \( akx \). Contrapositively speaking, every segment in \( in_a \) without an input–input correspondent in \( in_b \) has no output correspondent.

(2.54) If \( bmx \) has greater internal similarity than \( akx \), then for every segment \( s_a \) in \( in_a \) without an output correspondent, either \( s_a \) has an input–input correspondent \( s_b \) that also has no output correspondent and is identical to \( s_a \), or \( s_a \) has no input–input correspondent.

If \( s_a \) had a non-identical input–input correspondent \( s_b \), then the corresponding deletion disparities would be non-identical, contradicting the premise that \( bmx \) has greater internal similarity than \( akx \).
The definition in (2.51) requires that corresponding disparities be identical in order to support a relation of greater internal similarity. This requirement figures quite directly in (2.52) and (2.54). Identity disparities are only identical if both disparities have the same feature value on the input side and the same feature value on the output side. Deletion disparities are only identical if the deleted input segments are identical (have all of the same feature values). Requiring corresponding disparities to be identical is important to capturing the notion of surface orientedness.

The significance of requiring identity of deletion disparities can be illustrated with coda deletion phenomena. Consider a language which allows homorganic nasal-stop clusters and geminates, but no other coda consonants, and in which obstruents are deleted to avoid codas, but nasals assimilate in place when followed by a stop. Lombardi (2001) gives just this description for Diola (Sapir 1965), Akan (Schachter and Fromkin 1968), and Axininca person prefixes (Payne 1981). In this pattern, the candidate /ketbu/[kebu], with the third input segment /t/ deleted and the others in the expected IO correspondence, is grammatical. Under the definition of greater internal similarity, the candidate /kenbu/[kebu], with input segment /n/ deleted, does not have greater internal similarity than /ketbu/[kebu]. The deletion disparity in /kenbu/[kebu], deleting /n/, is not identical to the deletion disparity in /ketbu/[kebu], deleting /t/. A different candidate for the input /kenbu/, /kenbu/[kembu] with the order-preserving bijective IO correspondence, is grammatical in this pattern, where the third input segment, /n/, has an output correspondent, [m], which has assimilated in place to the following stop.

If /kenbu/[kebu] were considered to have greater internal similarity than /ketbu/[kebu], then in an output-driven map /kenbu/[kebu] would have to be grammatical whenever /ketbu/[kebu] was. But the grammaticality of /kenbu/[kembu] along with /ketbu/[kebu] can be accounted for by output restrictions. A ban on non-geminate obstruents in codas could account for the deletion of /t/ while still allowing the grammar to choose /kenbu/[kembu] rather than /kenbu/[kebu]. While the segments being deleted in the candidates /ketbu/[kebu] and /kenbu/[kebu] do not appear in the outputs of the candidates, their feature values have relevant relationships to possible output restrictions. Output restrictions can be responsible for deletion disparities in a way that is sensitive to the feature values of the deleted segments.

The lack of a greater internal similarity relation between /ketbu/[kebu] and /kenbu/[kebu] contrasts with a case like /tik/[ti] having greater internal similarity than /tek/[ti]. A surface-oriented map should not be able to distinguish the fate of /k/ in one input from the other when the rest of the input surfaces
identically. The only difference between the two inputs, /tek/ and /tik/, is the height of the vowel. The vowel height for /tek/ is neutralized to +hi in the output of the grammatical candidate /tek/[ti]. The only way for a map to treat the final /k/ differently for input /tik/ is to make reference to something other than the /k/ itself and the output: the quality of the vowel in the input. The intuition of surface orientedness here is that the fate of a potential output segment, like [k], can be directly dependent only upon its input correspondent, and the rest of the output. A map admitting /tek/[ti] but not /tik/[ti] does not conform to this intuition about surface orientedness and does not satisfy the definition of output-driven map.

2.4.7 Individuating disparities (again)

The types of disparities listed in (2.37) and exemplified in the inventory of disparities in (2.38) are non-uniform with respect to representational level. The identity disparities are described in terms of feature values, while the insertion and deletion disparities are described in terms of entire segments. When segments are distinguished entirely by their feature values and all features are obligatorily present, it might seem more uniform to view each deleted segment as a set of deletion disparities, one for each feature of the deleted segment, and similarly view each inserted segment. This would allow all disparities to be characterized at the level of features. The danger of that view is the creation of the mistaken impression that the features stand in input–output correspondence on their own. That is definitely not the case for the representational theory developed here in Section 2.4; only segments stand in input–output correspondence. All disparities are defined in terms of segmental IO correspondence. The featural evaluations in disparities are all parasitic on the IO correspondence relations of the segments bearing the features. IO correspondence is defined in terms of segments, while identity of segments is defined in terms of feature values.

For input–output identity disparities, how you individuate the disparities can make a big difference. This was illustrated in Section 2.2. Individuating identity disparities only in terms of entire segments produces a different relative similarity relation than individuating in terms of individual feature values does.

For insertion and deletion disparities, individuating separate feature disparities yields the same relative similarity relation as individuating whole segment disparities. Given two deleted segments, standing in input–input correspondence, each would have one feature deletion disparity for each feature of the deleted segment. Unless the two segments are identical, there will be at least one feature for which the deleted segments have different values. The disparity
for the deletion of that feature’s value in one candidate will be missing from the other, and vice-versa; the deletion disparities will not be identical, because they will involve different values of the same feature. Thus they will not be able to sustain a conclusion that the candidate containing one has greater internal similarity than the candidate containing the other, the same result as when the disparities are individuated by entire segment.

To illustrate, suppose we have two candidates, and they have corresponding deletion disparities a:_ and e:_. Stated as segmental disparities, the two are clearly not identical: /a/ is not identical to /e/. If we translate these into feature-level disparities, the candidate with a:_ has the disparities in (2.55), while the candidate with e:_ has the disparities in (2.56).

\[(2.55) \quad +\text{low:}_- -\text{hi:}_-\]

\[(2.56) \quad -\text{low:}_- -\text{hi:}_-\]

The disparities involving the feature hi are identical, but the disparities involving the feature low are not, so it is still the case that each candidate has a disparity without an identical corresponding disparity in the other candidate. Because segmental identity is defined purely in terms of feature values, two input–input corresponding segments will only have identical corresponding feature deletion disparities if they are in fact identical segments.

This highlights the role of IO correspondence in capturing the spirit of surface orientedness. The representational elements that can stand in IO correspondence restrict the scope of what can be an individual disparity. Any individual disparity can make reference to at most a single pair of IO correspondents. In a system where only segments may have IO correspondents, the disparities must be individuated so that they refer to at most one segment and its IO correspondent. Any interaction between input segments must be mediated by the output: each of the input segments can directly affect its output correspondent, and the two output correspondents can interact via output restrictions.

The theory of IO correspondence helps define the units in which internal similarity is measured. Deletion and insertion disparities are individuated in terms of the units that stand in IO correspondence; here, segments. Identity disparities are individuated in terms of the representational elements that distinguish the units standing in IO correspondence; here, features of segments. Together, these two aspects of representational theory help determine the nature of internal similarity, which in turn constitutes a bridge between the specifics of a particular representational theory and the general theory of output-driven maps.
2.5 Expanding to other representational theories

2.5.1 Non-identical corresponding representational elements

The initial intuition behind output drivenness is that if an input is unfaithfully mapped to an output, then you continue to get the same output as you change the input to be more like the output. This rests on the further intuition that input and outputs are fundamentally built out of the same kinds of “things.” For example, inputs and outputs both consist of segments, specifically the same kinds of segments, ones with the same features. More precisely, what is assumed is that the elements of inputs and outputs that stand in IO correspondence are the same kinds of things. The definition of output-driven maps has no trouble with grammars in which output forms have prosodic structures like syllables that input lack, so long as such prosodic elements do not stand in IO correspondence relations.\(^\text{12}\)

It is possible to relax the assumption that elements standing in IO correspondence are exactly the same types of objects.\(^\text{13}\) What is required is a specification of which IO correspondence relationships qualify as “faithful” ones. An example of this would be an analysis in which input segments have the feature accent, whereas output segments have instead the features stress, tone, and pitch_accent. In such an analysis, an input segment with the feature value +accent would not have a feature value disparity if its output correspondent had any of the values +stress, +tone, or +pitch_accent. Note that the term “feature” could be construed rather generically here as denoting any representational element: an output segment having the feature value +stress might actually be realized in the output representation as the output segment having a certain kind of projection on a prominence grid (Liberman and Prince 1977, Prince 1983). What matters is that the analysis specify which pairs of respective properties of the IO corresponding elements count as faithful, that is, which pairs do not constitute disparities.

Another example in the same domain would again have the feature accent be solely a property of input segments, and an input segment with +accent would not have a feature value disparity if its output correspondent segment was in a prosodic head position. The relation between prosodic heads and stress, pitch accent, etc. would then be purely matters of output structure. In this case, although it is strictly segments that are in IO correspondence, properties of

\(^\text{12}\) Moreton (2004) calls such elements “nonhomogeneous elements.”

\(^\text{13}\) This is a very different issue from the matter of requiring corresponding disparities to be identical, the topic of Section 2.4.6.
prosodic structure (prosodic headship) are being allowed into the identity of the disparities.

2.5.2 Non-unique correspondence
The conditions in (2.36) include the condition that correspondence is unique, in the sense that any element has at most one IO correspondent. Expanding the analysis of output-driven maps to include non-unique correspondence (coalescence and breaking) will be non-trivial. Adding violations of uniqueness to the inventory of disparities requires specifying precisely how to individuate the instances. The individuation is necessary to establish a correspondence between the disparities of two candidates having the same output form, to determine if one candidate has greater internal similarity than the other.

Individual segmental correspondences alone cannot constitute individuated disparities: a correspondence between an input segment and an output segment isn’t a non-uniqueness disparity on its own, only when combined with another segmental correspondence involving either the same input segment or the same output segment. A segment having more than one IO correspondent could constitute an individuated non-uniqueness disparity; the disparities would be localized to single segments. This might prove inadequate, because segments with non-identical sets of correspondents would simply constitute non-identical disparities. The potential inadequacy lies in the intuition that if one disparity involves a segment with three IO correspondents, and another disparity involves the same segment with two of the three IO correspondents, the latter disparity should somehow count as “greater similarity,” rather than simply non-identity; one of the offending IO correspondences has been removed, and that ought to result in greater internal similarity (other things being equal). If preserving this intuition ultimately proves desirable, it might help to shift from a relative similarity in which corresponding disparities must be identical to one in which distinct disparities can stand in an ordered similarity relation, so that one non-uniqueness disparity can count as “greater similarity” than another if the first disparity involves IO correspondence with a set of segments that is a strict subset (short of the empty set) of the set of corresponding segments in the second disparity.

2.5.3 Autosegmental representation
The implications of autosegmental representations (Goldsmith 1976) for output-driven maps depend upon precisely how such representations are used in an analysis. Output drivenness concerns IO correspondence. If output representations are realized with root nodes linked to autosegments as an expression of
feature values, and the only IO correspondents are input segments and output root nodes (segments, effectively), then the segment-based analysis of output drivenness given in Section 2.4 could apply without alteration.

On the other hand, the analysis of Section 2.4 would need to be extended to handle representational theories in which elements other than segments can be IO correspondents. This would be the case if feature values were realized as autosegments in both the input and output, and autosegments could correspond independent of segmental affiliation. Within Optimality Theory, the work of Lombardi 2001, among others, presumes the existence of multiple correspondence relations of this sort.

Zoll proposes a more flexible view of a single IO correspondence relation in her work on floating features, proposing that both inputs and outputs can contain floating features as well as full segments, and that floating features in the input can correspond to either full output segments or to output floating segments (Zoll 1998: 40). Analyzing output drivenness with those representational assumptions would require addressing a number of issues, such as the implications for internal similarity when input floating features have full segment output correspondents (and vice-versa).

2.6 The map

The definition of output-driven map formally realizes the concept of disparities being motivated by output restrictions without needing to commit in advance to the content of the output restrictions. That allows output-driven maps to capture surface orientedness in a very general way, one that is equally compatible with ordered rule theories and constraint optimization theories.

Output drivenness translates the intuitive notion of introducing disparities “to the extent necessary” into entailment relations between candidates. This will prove very useful in Chapter 3, where entailment relations between candidates will be further translated into entailment relations between ranking conditions in Optimality Theory. That chapter will give a strong characterization of what it is about an Optimality Theoretic grammar that determines if it defines an output-driven or non-output-driven map.

The generality of output drivenness makes it less subject to analysis-specific ambiguity than approaches that are process based. This was demonstrated in Section 2.3. The map in the Lithuanian case is output-driven, a conclusion that is compatible with the intuition that the phenomenon is driven by an output restriction (a ban on adjacent identical obstruent stops), even though that restriction cannot be recognizably expressed in the form of a traditional
Output-driven maps

transparent process. Output drivenness clearly distinguishes cases like non-initial syllable onset insertion and Lithuanian voice assimilation from cases that are truly not surface oriented (like chain shifts).

Some cases in which the presumed map is not output-driven will be examined in Chapter 4. The discussion there will involve both the maps themselves and Optimality Theoretic analyses of the cases, connecting with the material of Chapter 3 to expose constraint properties responsible for producing the non-output-driven effects.
This chapter examines the relationship between output-driven maps and Optimality Theory and gives sufficient conditions ensuring that an Optimality Theoretic system defines only output-driven maps. For concreteness, all discussion will assume the representational conditions specified in Section 2.4.

Section 3.1 reviews some crucial background theory from the existing literature that will be used in the rest of the chapter (and later in the book as well). Section 3.2 develops the key connection between output-driven maps and Optimality Theory and presents the format that will be used for arguments concerning output drivenness in Optimality Theory. Section 3.3 states a set of conditions on Optimality Theoretic systems that are sufficient to ensure that all grammars definable in such a system generate output-driven maps and gives a proof of the sufficiency of those conditions. Of particular interest are the conditions on constraints: any constraint which meets these conditions is said to be output-driven preserving, and cannot cause a grammar to generate a map that is non-output-driven.

Section 3.4 summarizes the status of some of the most basic types of constraints in Optimality Theory: they are output-driven preserving. It is easy to prove that all markedness constraints are output-driven preserving, as is shown in Section 3.4.2. Proving that some of the most basic faithfulness constraints are output-driven preserving takes much more effort. While Section 3.4 briefly sketches the arguments relevant to the constraints Max-IO, Dep-IO, and Ident-IO, the rigorous formal demonstration that those constraints are output-driven preserving is given in Sections 3.5 and 3.6 (readers less interested in the intricate details might want to skip Sections 3.5 and 3.6 on the first reading). Section 3.5 works through a classification of all possible relationships between disparities that are relevant to output drivenness. Section 3.6, drawing heavily on the results of Section 3.5, gives the actual proofs that Max-IO, Dep-IO, and two generalized versions of Ident-IO are output-driven preserving, and thus incapable of causing maps that are non-output-driven.

63
As just explained, the second part of this chapter focuses on constraints which are output-driven preserving. The next chapter, Chapter 4, focuses on what kind of constraint behavior is required for a constraint to fail to be output-driven preserving, providing a rigorous understanding of the kinds of constraint behavior capable of causing non-output-driven maps in OT systems. The property output-driven preserving will then be shown to provide a unified understanding of a variety of constraints that have been proposed in the literature for the analysis of phenomena that aren’t surface oriented.

3.1 Background: ERC entailment in Optimality Theory

The results in this chapter are based upon a connection that can be made between mapping entailment, which lies at the heart of output drivenness, and elementary ranking condition entailment, a key concept in Optimality Theory. This section provides background information on elementary ranking conditions and their entailment relations.

3.1.1 Elementary ranking conditions

In Optimality Theory, candidates are evaluated by constraints. Each constraint assesses some number (a non-negative integer) of violations to a candidate. Violations are penalizing: a candidate fares better on a constraint if it has fewer violations, and the best a candidate can do on a constraint is zero violations. Because the theory is inherently comparative, what matters is not so much the absolute number of violations assessed a candidate, but the number of violations relative to other (competing) candidates. The magnitude of the difference in the number of violations doesn’t matter. What matters when comparing two candidates with respect to a constraint is whether they have a differing number of violations of that constraint and, if so, which candidate has fewer violations.

A constraint renders one of three possible evaluations of a comparison between two candidates. Following Prince 2002, a comparison between two candidates \( w \) and \( l \) is denoted \( w \sim l \). The first of the two candidates (here, \( w \)) is assigned the role labeled “winner,” while the second candidate (here, \( l \)) is assigned the role labeled “loser.” Such a pair is routinely called a winner–loser pair. The three possible evaluations are the possible preferences between the candidates of the pair: the constraint can prefer the winner, it can prefer the loser, or it can have no preference. The possible evaluations of winner–loser pairs are denoted as shown in (3.1).
Notation for constraint evaluations of winner–loser pairs (“C” here is a constraint)

- \( C[w \sim l] = W \) indicates that \( C \) prefers \( w \) to \( l \).
- \( C[w \sim l] = e \) indicates that \( C \) has no preference between \( w \) and \( l \).
- \( C[w \sim l] = L \) indicates that \( C \) prefers \( l \) to \( w \).

In Optimality Theory, **harmony** is defined by a set of constraints ranked in a strict dominance hierarchy. The relative harmony of two candidates is decided by the highest constraint in the hierarchy (the highest-ranked constraint) that has a preference between the two candidates. For the comparison \( w \sim l \), if the highest-ranked constraint with a preference assigns the value \( W \), then \( w \) is more harmonic than \( l: w \gg l \). If the highest-ranked constraint with a preference assigns the value \( L \), then \( l \) is more harmonic than \( w: l \gg w \). Two candidates can have the same harmony only if none of the constraints has a preference between them. The expression \( w \gg l \) denotes the proposition that \( w \) is at least as harmonic as \( l \), and the expression \( w \gg l \) denotes the proposition that \( w \) is strictly more harmonic than \( l \).

An elementary ranking condition, or **ERC**, is a collection of the evaluations of a candidate comparison by all of the constraints of the system. The ERC for the comparison between candidates \( w \) and \( l \) (with \( w \) as the winner) is denoted \([w \sim l]\). An ERC expresses a condition on the ranking of the constraints: the winner of the ERC is more harmonic than the loser of the ERC for those rankings that satisfy the condition. Whenever the ERC \([w \sim l]\) is satisfied, it is necessarily the case that \( w \gg l \).

Consider the winner–loser pair in (3.2). The bottom row, labeled \([w \sim l]\), gives the ERC associated with this winner–loser pair. The constraint ML is violated zero times by the winner and is violated once by the loser, so the constraint ML is assigned the value \( W \). The constraint ID[L] is violated twice by the winner and only once by the loser, so it is assigned the value \( L \). The constraint ID[S] is violated once by each of the constraints, so it is assigned the value \( e \).

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ID[S]</th>
<th>ML</th>
<th>MR</th>
<th>ID[L]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w )</td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( l )</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>([w \sim l])</td>
<td>e</td>
<td>e</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td>W</td>
</tr>
</tbody>
</table>

The general content of an ERC is given in (3.3). The precise content of the ERC in (3.2) is given in (3.4).
66 Output-driven maps in Optimality Theory

(3.3) At least one W-assigning constraint must dominate all L-assigning constraints.

(3.4) \((\text{ML or NoLong}) \gg (\text{MR and ID}[L])\)

For any ranking satisfying the ERC \([w \sim l]\), it will be the case that the highest constraint with a preference between the candidates prefers the winner, \(w\), because at least one such constraint must dominate all constraints that prefer the loser, \(l\).

A candidate is grammatical (with respect to a given hierarchy) if it is optimal, that is, if it is at least as harmonic as all of its competing candidates. A constraint hierarchy determines grammaticality for a set of competitors. Reasoning the other way, in order for a candidate to be grammatical, the constraint ranking in question must satisfy all of the ERCs formed by comparing that candidate (as the winner) to each of its competing candidates. The set of such ERCs collectively define the conditions that must hold of a ranking in order for the specified winner to be grammatical.

While every winner–loser pair defines a unique ERC, in general an ERC does not have a unique winner–loser pair. Two different winner–loser pairs could impose identical requirements on the ranking, and it is entirely possible that some ERCs do not express the conditions for any winner–loser pair.

3.1.2 Single ERC entailment: L-retraction and W-extension

Prince (2002, 2003) lays out the theory of ERC entailment in Optimality Theory based on entailment relations among the three possible evaluations, L, e, and W. The entailment relations are reflected in a particular ordering of the three evaluations, shown in (3.5). The interpretation of this is that L entails each of \{L,e,W\}, e entails each of \{e,W\}, and W entails W. One ERC entails another in the linguistic system if, for each constraint, the evaluation of the first ERC entails the evaluation of the second.

(3.5) Evaluation entailment order: \(L \rightarrow e \rightarrow W\)

The validity of (3.5) follows from the logical structure of ERCs. To illustrate, recall the ERC in (3.2), which has the logical content given in (3.4). That logical content can be rewritten as a logical formula of strictly pairwise ranking relations, as shown in (3.6).

(3.6) \([(\text{ML} \gg \text{MR}) \land (\text{ML} \gg \text{ID}[L])] \lor [(\text{NoLong} \gg \text{MR}) \land (\text{NoLong} \gg \text{ID}[L])]\)

Consider the logical consequences of changing an L in the ERC to an e. Suppose we change the ERC’s evaluation for MR from L to e. This changes the ERC
from [e e W L L W] to [e e W e L W]. Such a maneuver is labeled \textbf{L-retraction}. The effect on the logical content of the ERC is that MR no longer needs to be dominated by anything (as far as this ERC is concerned). The resulting logical content of the resulting ERC is shown in (3.7).

\begin{equation}
(3.7) \quad (\text{ML} \gg \text{ID}[L]) \text{ or } (\text{NoLong} \gg \text{ID}[L])
\end{equation}

The formula in (3.7) is entailed by the one in (3.6). This follows from the general fact of classical logic that \([\text{A and B}] \text{ entails A}\). The first disjunct of (3.7), \((\text{ML} \gg \text{ID}[L])\), is entailed by the first disjunct of (3.6), \([(\text{ML} \gg \text{MR}) \text{ and } (\text{ML} \gg \text{ID}[L])]\). The second disjuncts have the same relationship. Thus, L-retraction results in an ERC that is entailed by the original.

Now consider the logical consequences of changing an e to a W. Suppose we take the ERC that resulted from L-retraction, [e e W e L W], and change the evaluation for constraint WSP from e to W, resulting in the ERC [W e W e L W]. Such a maneuver is labeled \textbf{W-extension}. The effect on the logical content is that the resulting ERC can be satisfied fully if WSP dominates ID[L]. The logical content of the resulting ERC is shown in (3.8).

\begin{equation}
(3.8) \quad (\text{ML} \gg \text{ID}[L]) \text{ or } (\text{NoLong} \gg \text{ID}[L]) \text{ or } (\text{WSP} \gg \text{ID}[L])
\end{equation}

The formula in (3.8) is entailed by the one in (3.7). This follows from the general fact of classical logic that \(\text{A entails } [\text{A or B}]\). The two formulas have the first two disjuncts in common; (3.8) adds an additional disjunct. Thus, W-extension results in an ERC that is entailed by the original.

\subsection{3.1.3 Joint ERC entailment: fusion}

Information can combine across ERCs to entail conditions that are not entailed by any of the ERCs in isolation. Consider the pair of ERCs in (3.9).

\begin{equation}
(3.9) \quad \text{A pair of ERCs with transitive entailment}
\end{equation}

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ID[S]</th>
<th>ML</th>
<th>MR</th>
<th>ID[L]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>erc1</td>
<td>W</td>
<td>e</td>
<td>W</td>
<td>e</td>
<td>L</td>
<td>e</td>
</tr>
<tr>
<td>erc2</td>
<td>e</td>
<td>e</td>
<td>L</td>
<td>e</td>
<td>W</td>
<td>e</td>
</tr>
</tbody>
</table>

Neither ERC, on its own, entails that WSP \(\gg\) ML. The first ERC, erc1, mentions ML, but doesn’t require it to be dominated by anything, while erc2 doesn’t require anything of WSP at all. But, taken together, the pair of ERCs do entail that WSP \(\gg\) ML. Erc1 requires that either WSP or ML dominate ID[L], but erc2 denies that ML can dominate ID[L], because ID[L] must dominate ML.
That leaves only WSP to dominate ID[L] in satisfaction of erc1. Since WSP $\gg$ ID[L], and ID[L] $\gg$ ML, it follows by transitivity that WSP $\gg$ ML.

The **fusion** operation (Prince 2002) is a way of explicitly extracting ranking information that is jointly entailed by a set of ERCs. The fusion of a set of ERCs is itself an ERC, one that is collectively entailed by the original set: if the set of ERCs is true, then the fusion of the set is true. The fusion operation applies separately for each constraint: the evaluation value (L, e, or W) for a constraint in the fusion is a function of the evaluation values for that constraint in each of the ERCs in the set. The fusion operator is denoted “$\circ$,” both for sets of individual evaluation values (single constraint) and full ERCs (multiple constraints).

(3.10) Fusion (X is a variable that can take any of the three evaluation values)

\[
X \circ X = X \\
X \circ L = L \\
X \circ e = X
\]

A demonstration of the fusion operator is given in (3.11): across the six constraints, all six possible combinations of two evaluation values are shown.

(3.11) An illustration of fusion (all possible pairs of evaluation values are shown)

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ID[S]</th>
<th>ML</th>
<th>MR</th>
<th>ID[L]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>erc3</td>
<td>e</td>
<td>L</td>
<td>W</td>
<td>e</td>
<td>L</td>
<td>W</td>
</tr>
<tr>
<td>erc4</td>
<td>W</td>
<td>e</td>
<td>L</td>
<td>e</td>
<td>L</td>
<td>W</td>
</tr>
<tr>
<td>erc3 $\circ$ erc4</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td>e</td>
<td>L</td>
<td>W</td>
</tr>
</tbody>
</table>

Jointly entailed ranking information for the pair of ERCs in (3.9) can be obtained by taking the fusion of the two ERCs, as shown in (3.12). The fusion of erc1 and erc2 requires that WSP dominate both ML and ID[L]. Note that the fusion ERC, [W e L e L e], entails both [W e L e e e] (WSP $\gg$ ML) and [W e e e L e] (WSP $\gg$ ID[L]); this follows from L-retraction in both cases.

(3.12) The fusion reveals jointly entailed ranking information

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ID[S]</th>
<th>ML</th>
<th>MR</th>
<th>ID[L]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>erc1</td>
<td>W</td>
<td>e</td>
<td>W</td>
<td>e</td>
<td>L</td>
<td>e</td>
</tr>
<tr>
<td>erc2</td>
<td>e</td>
<td>e</td>
<td>L</td>
<td>e</td>
<td>W</td>
<td>e</td>
</tr>
<tr>
<td>erc1 $\circ$ erc2</td>
<td>W</td>
<td>e</td>
<td>L</td>
<td>e</td>
<td>L</td>
<td>e</td>
</tr>
</tbody>
</table>

In general, it is not possible to squeeze all of the information contained in a set of ERCs into a single ERC. The fusion operation extracts some information,
but in the general case frequently loses other information. There are techniques, making use of the fusion operation, that can convert one set of ERCs into another set of ERCs, containing exactly the same ranking information, such that ranking relations between constraints that are only entailed via transitivity in the former set are directly represented in the latter set. Theory and methods behind various canonical forms for expressing the ranking information contained in a set of ERCs are given in (Brasoveanu and Prince 2011).

For our purposes, the fusion operation will be most useful in the analysis of learning; fusion will make multiple appearances in the second half of this book.

3.2 Relating output-driven maps to Optimality Theory

This section presents a way of relating the structure of output-driven maps to the structure of Optimality Theory, via the connection between the mapping entailment of ODM and the ERC entailment of OT. The final result of this section, in (3.37), will be used in Section 3.3 to derive conditions sufficient to ensure that an Optimality Theoretic system only generates output-driven maps.

The results in Sections 3.2.1 through 3.2.4 are based upon the assumption of a general optimization-based style of grammar; they could conceivably apply to certain other optimization-based grammars, such as harmonic grammar. The arguments in Section 3.2.5, however, draw crucially on the theory of ERC entailment presented in Section 3.1 and rely more on the strict domination of constraints in Optimality Theory.

3.2.1 Output-driven maps and optimization

In Optimality Theory, a candidate is optimal if it is at least as harmonic as all of its competitors (the other candidates sharing the same input form). The function GEN identifies sets of competitors; GEN(ina) denotes the set of candidates that have ina as their input form. The condition that akx is grammatical can be expressed as in (3.13), where aqz is here a variable representing any member of GEN(ina).

\[ \forall aqz(akx \gg aqz) \]  

(3.13)

The condition that bmx is grammatical can be similarly expressed as in (3.14), where bpy is here a variable representing any member of GEN(inb).

\[ \forall bpy(bmx \gg bpy) \]  

(3.14)

A map defined by an Optimality Theoretic system is output-driven if, for any pair akx and bmx such that bmx has greater internal similarity than akx, the
Output-driven maps in Optimality Theory

condition in (3.15) is true. This condition states, using the notation of harmony optimization, that the optimality of \( a_kx \) entails the optimality of \( b_{mx} \).

(3.15) \[ \forall aqz(a_kx \gg aqz) \rightarrow [\forall b_{py}(b_{mx} \gg b_{py})] \]

The same condition can be expressed by taking the contrapositive: the non-optimality of \( b_{mx} \) entails the non-optimality of \( a_kx \).

(3.16) \[ \neg [\forall b_{py}(b_{mx} \gg b_{py})] \rightarrow \neg [\forall aqz(a_kx \gg aqz)] \]

Applying each negation operator to the expression within its scope yields the result in (3.17).

(3.17) \[ [\exists b_{py}(b_{py} > b_{mx})] \rightarrow [\exists aqz(aqz > a_kx)] \]

In words, in an output-driven map, if \( b_{mx} \) has greater internal similarity than \( a_kx \), then the existence of a candidate (in \( Gen(in_b) \)) more harmonic than \( b_{mx} \) entails the existence of a candidate (in \( Gen(in_a) \)) more harmonic than \( a_kx \).

3.2.2 A designated competitor: \( aoy \)

The condition in (3.17) states that, for any candidate \( b_{mx} \) with greater internal similarity than \( a_kx \), the existence of a candidate \( b_{py} \) such that \( b_{py} > b_{mx} \) entails the existence of some candidate \( aqz \) such that \( aqz > a_kx \). To make the analysis more tractable, I will now adopt a stronger requirement, leading to sufficient conditions for output-driven maps. This stronger condition, given in (3.18), requires the existence of a particular candidate, \( aoy \), defined relative to \( b_{py} \), and requires that for any \( b_{py} \) such that \( b_{py} \) is more harmonic than \( b_{mx} \), the corresponding \( aoy \) is more harmonic than \( a_kx \). This allows the analysis to focus on only one competitor to \( a_kx \) relative to \( b_{py} \).

(3.18) \[ \forall b_{py} ((b_{py} > b_{mx}) \rightarrow (aoy > a_kx)) \]

The definition of \( aoy \), given below in (3.19), is not arbitrary; I will argue that under ordinary expectations about candidate spaces, this candidate will always exist, and that for most if not all output-driven maps of interest this candidate will always satisfy the condition in (3.18). In other words, if (3.17) is true, then it should be the case that (3.18) is true: if \( b_{py} > b_{mx} \), and there exists some \( aqz \) such that \( aqz > a_kx \), then the particular candidate \( aoy \) exists and is such that \( aoy > a_kx \).

A significant benefit of this approach is that the sufficient conditions for output-driven maps can be broken in to separate conditions on \( Gen \) (the candidate space) and on \( Con \) (the constraints). The conditions on \( Gen \) ensure that the candidate \( aoy \) exists as a candidate. The conditions on \( Con \) ensure
that \( aoy \succ akx \) whenever \( bpy \succ bmx \). Further, the conditions on \( \text{CON} \) apply to each constraint in isolation: a given constraint does or does not satisfy the conditions regardless of what other constraints also reside in the grammar, and \( \text{CON} \) satisfies the conditions if each individual constraint of \( \text{CON} \) satisfies the conditions.

The candidate \( aoy \) has input \( in_a \), identical to the input of \( akx \), and output \( out_y \), identical to the output of \( bpy \). What remains to be defined is its IO correspondence relation \( R_o \). \( R_o \) is defined relative to \( akx \), \( bmx \), and \( bpy \), where \( bmx \) has greater internal similarity than \( akx \). The premise that \( bmx \) has greater internal similarity than \( akx \) requires that an appropriate correspondence exist between the disparities of the candidates, which in turn requires that an appropriate input–input correspondence \( R_{II} \) exists between the inputs \( in_a \) and \( in_b \). Here, “appropriate” means that \( R_{II} \) satisfies the conditions given in (2.39) and supports the appropriate correspondence between the disparities of the candidates. \( R_o \) is defined in (87), in terms of \( R_{II} \) and \( bpy \).

\[
(3.19) \quad \text{The correspondence relation } R_o \text{ for candidate } aoy
\]

\[
\bullet \quad \text{For each segment } s_a \text{ in } in_a \text{ that has an input–input correspondent } s_b \text{ in } in_b,
\]

\[
\quad s_a \text{ has the same output correspondent in } aoy \text{ that } s_b \text{ has in } bpy; \text{ if } s_b \text{ has no output correspondent in } bpy, \text{ then } s_a \text{ has no output correspondent in } aoy.
\]

\[
\bullet \quad \text{Each segment } s_a \text{ in } in_a \text{ that does not have an input–input correspondent in } in_b \text{ has no output correspondent in } aoy.
\]

This fully determines the correspondence relation \( R_o \), which is summarized in (3.20).

\[
(3.20) \quad s_a R_o s_y \iff \exists s_b [s_u R_{II} s_b \text{ and } s_b R_p s_y]
\]

\( R_o \) is defined so that \( aoy \) relates to \( bpy \) in a way that is analogous to the way \( akx \) relates to \( bmx \). Recall from Section 2.4.3 that \( R_{II} \) relates segments of inputs \( in_a \) and \( in_b \) that have matching IO correspondence roles in their respective candidates \( akx \) and \( bmx \). A pair of input–input correspondents either have the same output correspondent, or else they both lack output correspondents. The definition of \( aoy \) preserves the “matching roles” property by assigning, to each segment of \( in_a \) with an input–input correspondent, the same IO correspondence role in \( aoy \) that its input–input correspondent has in \( bpy \). Given input–input correspondents \( s_a \) in \( in_a \) and \( s_b \) in \( in_b \), if \( s_b \) has an output correspondent in \( bpy \), then \( s_a \) has the same output correspondent in \( aoy \); if \( s_b \) has no output correspondent in \( bpy \), then \( s_a \) has no output correspondent in \( aoy \).

If \( s_a \) does not have an input–input correspondent in \( in_b \), then \( s_a \) necessarily has no output correspondent in \( akx \) (see (2.53)). In that case, \( s_a \) lacks an output
correspondent in $aoy$, analogous to the lack of an output correspondent for $s_a$
in $akx$.

The various relationships involving $aoy$ are here illustrated by building on
the example from Section 2.4.3. Repeated below are candidates $akx$ and $bmx$
from (2.41) and (2.42), as well as the derived input-input correspondence $R_{II}$
(2.43).

\begin{align}
(2.41) \quad in_a &= /agbik/ \quad out_x = [tibi] \quad R_k = \{(a_1, x_2), (a_2, x_3), (a_4, x_4)\} \\
(2.42) \quad in_b &= /tebik/ \quad out_x = [tibi] \quad R_m = \{(b_1, x_1), (b_2, x_2), (b_3, x_3), (b_4, x_4)\} \\
(2.43) \quad R_{II} &= \{(a_1, b_2), (a_3, b_3), (a_4, b_4), (a_5, b_5)\}
\end{align}

The disparities of $akx$ and $bmx$ are repeated below in (2.44) and (2.45), respec-
tively, along with the correspondence between the disparities (2.46).

\begin{align}
(2.44) \quad &: x_1 \quad a_1(-hi):x_2(+hi) \quad a_1(+low):x_2(-low) \quad a_2:\quad a_5:\quad \\
(2.45) \quad &: x_2(+hi) \quad b_2(-hi) \quad b_5:\quad \\
(2.46) \quad \text{A correspondence between the disparities of } bmx \text{ and } akx \\
\quad &\bullet \quad b_2(-hi):x_2(+hi) \text{ of } bmx \text{ corresponds to } a_1(-hi):x_2(+hi) \text{ of } akx \\
\quad &\bullet \quad b_5:\quad \text{of } bmx \text{ corresponds to } a_5:\quad \text{of } akx
\end{align}

Now consider a new output form, $out_y = [tebi]$. Consider also a candidate
with input $in_b$ (the same input as for $bmx$), output $out_y$, and IO correspondence
relation $R_p$. This candidate, $bpy$, is summarized in (3.21), and its disparities
are listed in (3.22).

\begin{align}
(3.21) \quad in_b &= /tebik/ \quad out_y = [tebi] \quad R_p = \{(b_1, y_1), (b_2, y_2), (b_3, y_3), (b_4, y_4)\} \\
(3.22) \quad b_5:\quad
\end{align}

Relative to $bpy$, the definition in (3.19) yields the candidate $aoy$ summarized in
(3.23), which has the disparities listed in (3.24).

\begin{align}
(3.23) \quad in_a &= /agbik/ \quad out_y = [tebi] \quad R_o = \{(a_1, y_2), (a_3, y_3), (a_4, y_4)\} \\
(3.24) \quad &: y_1 \quad a_1(+low):y_2(-low) \quad a_2:\quad a_5:\quad
\end{align}

Figure 3.1 repeats the candidates $akx$ and $bmx$ from Figure 2.2 and additionally
shows the candidates $aoy$ and $bpy$, along with the input–output correspondence
relation $R_{II}$. $R_{II}$ is (by definition of $aoy$) identical for both pairs of candidates.
The candidates $aoy$ and $bpy$ differ from $akx$ and $bmx$ in that they involve output
form $out_y = [tebi]$. 
3.2 Relating output-driven maps to Optimality Theory

The use of the input–input correspondence $R_{II}$ in the definition of the correspondence relation $R_o$ patterns the relationship between the IO correspondences of $aoy$ and $bpy$ along the same lines as the relationship between the IO correspondences of $akx$ and $bmx$. This can be seen for each of the five segments of $i_{n_2}:

- Input segment $a_1$ has input–input correspondent $b_2$. Because candidate $bpy$ relates input segment $b_2$ to its output correspondent $y_2$, $a_1$ has output correspondent $y_2$ in $aoy$. This mirrors the fact that $b_2$ and $a_1$ have the same output correspondent, $x_2$, in $bmx$ and $akx$, respectively.
- Input segment $a_2$ has no input–input correspondent. Therefore, $a_2$ in $aoy$ has no output correspondent. This is analogous to the fact that $a_2$ in $akx$ has no output correspondent.
- Input segment $a_3$ has input–input correspondent $b_3$. Because candidate $bpy$ relates input segment $b_3$ to its output correspondent $y_3$, $a_3$ has output correspondent $y_3$ in $aoy$. This mirrors the fact that $b_3$ and $a_3$ have the same output correspondent, $x_3$, in $bmx$ and $akx$, respectively.
- Input segment $a_4$ has input–input correspondent $b_4$. Because candidate $bpy$ relates input segment $b_4$ to its output correspondent $y_4$, $a_4$ has output correspondent $y_4$ in $aoy$. This mirrors the fact that $b_4$ and $a_4$ have the same output correspondent, $x_4$, in $bmx$ and $akx$, respectively.
- Input segment $a_5$ has input–input correspondent $b_5$. Because input segment $b_5$ has no output correspondent in $bpy$, $a_5$ has no output correspondent in $aoy$. This mirrors the fact that $b_5$ and $a_5$ both lack output correspondents in $bmx$ and $akx$, respectively.
Input–input correspondents should have the same IO correspondence “fate” in both \textit{aoy} and \textit{bpy}, but it will not always be the case that their shared IO correspondence fate in \textit{aoy} and \textit{bpy} will be the same as it is in \textit{akx} and \textit{bmx}. The possibility of the fates being different is illustrated in another example, in Section 3.2.4.

The point of defining \textit{aoy} as in (3.19) is to achieve an analogical alignment between the disparities of the candidates. The formal details of this analogical alignment of disparities and its implications are presented in Section 3.2.3, but the motivating result can be previewed here. Any reduction of disparities that \textit{bpy} has relative to \textit{bmx} will have an analogous reduction of disparities for \textit{aoy} relative to \textit{akx}. Such a relationship supports (3.18) above: if those differences in disparities are sufficient to make \textit{bpy} more harmonic than \textit{bmx}, then they should also make \textit{aoy} more harmonic than \textit{akx}. The analogous reduction of disparities is the key, because any harmonic difference due to a reduction in markedness follows automatically: if a markedness constraint prefers \textit{out}_y to \textit{out}_x when comparing \textit{bpy} and \textit{bmx}, it must necessarily also prefer \textit{out}_y to \textit{out}_x when comparing \textit{aoy} and \textit{akx}. Thus, whatever the reason (markedness or faithfulness), if \textit{bpy} is more harmonic than \textit{bmx}, then \textit{aoy} will be more harmonic than \textit{akx}. The analogous reduction of disparities is the key, because any harmonic difference due to a reduction in markedness follows automatically: if a markedness constraint prefers \textit{out}_y to \textit{out}_x when comparing \textit{bpy} and \textit{bmx}, it must necessarily also prefer \textit{out}_y to \textit{out}_x when comparing \textit{aoy} and \textit{akx}. Thus, whatever the reason (markedness or faithfulness), if \textit{bpy} is more harmonic than \textit{bmx}, then \textit{aoy} will be more harmonic than \textit{akx} for the same reason.

3.2.3 Relationships among the disparities

Fully describing the relationships among the disparities of the four candidates requires associating some disparities in \textit{aoy} with analogous disparities in \textit{akx}. These relationships between disparities are a kind of correspondence, but they are different in some respects from the disparity correspondences that are the basis of relative similarity. For one thing, \textit{aoy} and \textit{akx} do not (in general) have identical outputs, but instead have identical inputs. Further, the associations between disparities in \textit{aoy} and \textit{akx} are determined with respect to not just \textit{aoy} and \textit{akx} themselves, but with respect to the relationships between \textit{aoy} and \textit{bpy}, and between \textit{akx} and \textit{bmx}.

For the rest of this book, disparities that correspond between candidates with identical outputs (in support of relative similarity) will be referred to as corresponding disparities, while disparities that correspond between candidates with identical inputs (in support of analogy) will be referred to as analogous disparities. The difference in terminology helps the exposition by making it easier to distinguish which kinds of relationships between disparities are being referred to at any given point.

The relationships among the disparities of the four candidates depicted in Figure 3.1 are shown in Figure 3.2. Each candidate is represented by a square...
3.2 Relating output-driven maps to Optimality Theory

box, containing the list of the disparities of that candidate; the lists of disparities come directly from (2.44), (2.45), (3.22), and (3.24).

The double-sided arrows point to corresponding disparities. The two double-sided arrows between the boxes for \( akx \) and \( bmx \) indicate the two pairs of corresponding disparities, as listed in (2.46). A correspondence can also be constructed between the disparities of \( aoy \) and \( bpy \), based on \( R_{II} \), just as was described in (2.40) for \( akx \) and \( bmx \); this correspondence is listed in (3.25), and depicted by the double-sided arrow between the boxes for \( aoy \) and \( bpy \). The disparities are both deletion disparities, and correspond because the deleted segments, \( a_{5} \) and \( b_{5} \), are input–input correspondents.

(3.25) Correspondence between the disparities of \( aoy \) and \( bpy \)
- \( a_{5} : _{\text{\textendash}} \) of \( aoy \) corresponds to \( b_{5} : _{\text{\textendash}} \) of \( bpy \)

While \( bmx \) has greater internal similarity than \( akx \), it is not necessarily the case that \( bpy \) has greater internal similarity than \( aoy \). It is possible for \( aoy \) and \( bpy \) to have corresponding disparities that are non-identical, and it is possible for \( bpy \) to have a disparity with no corresponding disparity in \( aoy \). Demonstrations of these possibilities can be found in Section 3.5.

The definition of \( aoy \) is designed to achieve a particular relationship between \( aoy \) and \( akx \), one in which certain disparities in \( aoy \) have analogous disparities

Figure 3.2. Relationships among the disparities. Corresponding disparities (same output) are connected with double-sided arrows; analogous disparities (same input) are connected by the square-angled lines.
Output-driven maps in Optimality Theory

in \(\text{akx}\). The definition of analogous disparities between \(\text{aoy}\) and \(\text{akx}\), given in (3.26), is based on the shared input form between the two candidates, and the disparity correspondences between \(\text{akx}\) and \(\text{bmx}\) and between \(\text{aoy}\) and \(\text{bpy}\), which in turn are based on the input–input correspondence \(\text{R}_{\text{II}}\).

\[
\text{(3.26) General definition of analogous disparities}
\]

- Let \(s_{\alpha}[:_\alpha}\) be a deletion disparity of \(\text{aoy}\). This disparity in \(\text{aoy}\) has an analogous deletion disparity \(s_{\alpha}[:_\alpha}\) in \(\text{akx}\) if and only if \(s_{\alpha}\) has no output correspondent in \(\text{akx}\).
- Let \(:_\beta s_{\gamma}\) be an insertion disparity in \(\text{aoy}\). This disparity has an analogous insertion disparity \(:_\beta s_{\gamma}\) in \(\text{akx}\) if and only if \(s_{\gamma}\) has an input correspondent \(s_{\beta}\) in \(\text{bpy}\), and \(s_{\beta}\) has output correspondent \(s_{\gamma}\) in \(\text{bmx}\).
- Let \(s_{\alpha}(\alpha):s_{\gamma}(\beta)\) be an identity disparity of \(\text{aoy}\) for feature \(F\) (\(\alpha \neq \beta\)). This disparity in \(\text{aoy}\) has an analogous identity disparity in \(\text{akx}\) if and only if \(s_{\alpha}\) has an output correspondent \(s_{\gamma}\) in \(\text{akx}\) such that \(s_{\alpha}\) and \(s_{\gamma}\) differ on the value of feature \(F\).

Turning back to the specific example in Figure 3.2, candidate \(\text{aoy}\) has four disparities, and all four have analogous disparities in \(\text{akx}\). The analogous disparities between \(\text{aoy}\) and \(\text{akx}\) for the example are listed in (3.27). Analogous disparities are depicted in Figure 3.2 with the square-angle lines.

\[
\text{(3.27) Analogous disparities of \(\text{aoy}\) and \(\text{akx}\).}
\]

- \(a_2[:_]\) of \(\text{aoy}\) is analogous to \(a_2[:_]\) of \(\text{akx}\).
- \(a_1(+\text{low}):y_2(–\text{low})\) of \(\text{aoy}\) is analogous to \(a_1(+\text{low}):x_2(–\text{low})\) of \(\text{akx}\).
- \(:_y y_1\) of \(\text{aoy}\) is analogous to \(:_x x_1\) of \(\text{akx}\).
- \(a_5[:_]\) of \(\text{aoy}\) is analogous to \(a_5[:_]\) of \(\text{akx}\).

Two of the analogs in (3.27) are between deletion disparities. These are based on the fact that the analogous disparities have the same input segment. Candidates \(\text{aoy}\) and \(\text{akx}\) share input \(\text{in}_a\), so \(a_2[:_]\) in one is analogous to \(a_2[:_]\) in the other. One of the analogous is between insertion disparities, and the output segments are necessarily from different output forms. Disparity \(:_y y_1\) of \(\text{aoy}\) involves segment \(y_1\) of \(\text{out}_y\). Segment \(y_1\) has input correspondent \(b_1\) in \(\text{bpy}\), \(b_1\) has output correspondent \(x_1\) in \(\text{bmx}\), and \(x_1\) in turn is the inserted output segment of the analogous disparity \(:_x x_1\) of \(\text{akx}\). One of the analogous is between identity disparities, based on the fact that the analogous disparities have the same input segment and involve an identity disparity on the same feature (low). These analogous disparities serve to maintain the analogy among the four candidates: \(\text{aoy}\) is to \(\text{akx}\) as \(\text{bpy}\) is to \(\text{bmx}\).

Regarding the first pair of disparities in (3.27), \(a_2[:_]\) of \(\text{akx}\) has no corresponding disparity in \(\text{bmx}\), reflected by the fact that \(a_2\) has no input–input correspondent. The same input segment is thus also deleted in \(\text{aoy}\), constituting a disparity with no corresponding disparity in \(\text{bpy}\). Disparity \(a_2[:_]\) of \(\text{aoy}\) is
3.2 Relating output-driven maps to Optimality Theory

analogous to \( a_2 : \_ \) of \( akx \). Disparities corresponding to \( a_2 : \_ \) do not appear in \( bpy \) or \( bmx \) and provide no distinctions between those two competitors, while the same disparity involving \( a_2 \) appears in both \( aoy \) and \( akx \) and provides no distinctions between those two competitors.

Regarding the second pair of disparities in (3.27), \( a_1 (+\text{low}): x_2 (–\text{low}) \) of \( akx \) has no corresponding disparity in \( bmx \): the input–input correspondent of \( a_1 \), \( b_2 \), has feature value –low, the same as \( x_2 \), its output correspondent in \( bmx \). In this example, \( b_2 \) also has no identity disparity for feature low in \( bpy \): its output correspondent \( y_2 \) is also –low. Input–input correspondents \( a_1 \) and \( b_2 \) have the same output correspondent, \( x_2 \), in \( akx \) and \( bmx \), respectively, so the overall analogy requires that \( a_1 \) in \( aoy \) have the same output correspondent that \( b_2 \) has in \( bpy \), namely \( y_2 \). Thus, \( aoy \) has an identity disparity for feature low involving \( a_1 \) that is analogous to the identity disparity for feature low involving \( a_1 \) in \( akx \). The lack of identity disparities for low involving \( b_2 \) provide no distinctions between \( bpy \) and \( bmx \), and the analogous identity disparities for low involving \( a_1 \) provide no distinctions between \( aoy \) and \( akx \).

Regarding the third pair of disparities in (3.27), the disparities \( _\_ : y_1 \) and \( _\_ : x_1 \) for \( aoy \) and \( akx \), respectively, are analogous because both output segments have the same input correspondent (\( b_1 \)) in the candidates with input \( in_a \), \( bpy \) and \( bmx \). Because \( bmx \) has greater internal similarity than \( akx \), it follows that \( x_1 \) is identical to \( y_1 \) (here, both are [1]). In this example \( y_1 \) is also identical to \( x_1 \),\(^1\) so the lack of disparities involving \( x_1 \) in \( bmx \) has an analogous lack of disparities involving \( y_1 \) in \( bpy \), providing no distinctions between those two competitors. The identity of \( y_1 \) and \( x_1 \) also entails the identity of disparities \( _\_ : y_1 \) and \( _\_ : x_1 \), providing no distinctions between \( aoy \) and \( akx \).

Regarding the fourth pair of disparities in (3.27), \( a_5 : \_ \) of \( aoy \) has corresponding disparity \( b_5 : \_ \) in \( bpy \), and analogous disparity \( a_5 : \_ \) in \( akx \). The analogous disparity \( a_5 : \_ \) in \( akx \) also has a corresponding disparity \( b_5 : \_ \) in \( bmx \). The final \( k \) of the inputs, \( a_5 \) for \( in_a \) and \( b_5 \) for \( in_b \), is deleted in all four candidates, and the four deletion disparities are collective counterparts, via disparity correspondence and disparity analogy. The disparities provide no distinctions between \( bpy \) and \( bmx \) and provide no distinctions between \( aoy \) and \( akx \). This illustrates the possibility of a disparity in \( aoy \) having both a corresponding disparity in \( bpy \) and an analogous disparity in \( akx \).

The four analogous disparities between \( aoy \) and \( akx \) help to demonstrate that the disparity relationship between \( aoy \) and \( akx \) is patterned after the disparity

\(^1\) While in this example \( y_1 \) is also identical to \( x_1 \), that will not be true in general: in other cases, the same input segment in \( in_a \) (\( b_1 \)) could have an output correspondent in \( bpy \) (\( y_1 \)) that is non-identical to its output correspondent in \( bmx \) (\( x_1 \)). Such a case does not undermine the relationship between \( bpy \) > \( bmx \) and \( aoy \) > \( akx \), however; see Section 3.5.
relationship between \(bpy\) and \(bmx\). Each pair of analogous disparities either has no corresponding disparities in \(bpy\) and \(bmx\), or has corresponding disparities in both \(bpy\) and \(bmx\). Another crucial part of the overall relationship comes from disparities that \(akx\) has but \(aoy\) does not. In the example, \(akx\) has a disparity, \(a_1(-hi):x_2(+hi)\), with no analogous disparity in \(aoy\). It does, however, have a corresponding disparity in \(bmx\), \(b_2(-hi):x_2(+hi)\), which in turn has no analogous disparity in \(bpy\). Looked at the other way, for each disparity that \(bpy\) lacks relative to \(bmx\), \(aoy\) lacks a related disparity relative to \(akx\). Any harmonic advantage \(bpy\) might have relative to \(bmx\) (by lacking a disparity that \(bmx\) has), \(aoy\) will have the same harmonic advantage relative to \(akx\) (it will lack a corresponding disparity of \(akx\)).

A general property of the definition of \(aoy\) is that every disparity of \(aoy\) will have either a corresponding disparity in \(bpy\) or an analogous disparity in \(akx\). A proof of this property may be found in the analysis of the relationships between disparities given in Section 3.5. In the example in Figure 3.2, every disparity of \(aoy\) has an analog in \(akx\), but that will not be true in general. Any disparity of \(aoy\) which does not have an analogous disparity in \(akx\) will have a corresponding disparity in \(bpy\), one lacking an analogous counterpart in \(bmx\).

To summarize the significant relationships among disparities involving \(aoy\), for any disparity that \(bpy\) lacks relative to \(bmx\), \(aoy\) lacks a corresponding disparity relative to \(akx\). For any disparity that \(aoy\) has without an analog in \(akx\), \(bpy\) has a corresponding disparity without an analog in \(bmx\).

The general condition being approximated here is the one given in (3.17): if \(bpy\) is more harmonic than \(bmx\), then some candidate is more harmonic than \(akx\). Given that \(bpy > bmx\), and that in an output-driven map some candidate must be more harmonic than \(akx\), we would expect specifically \(aoy > akx\). If \(bpy > bmx\) because of an output restriction preferring \(out_y\) to \(out_x\), clearly that condition will also prefer \(aoy\) to \(akx\). If \(bpy > bmx\) because \(bpy\) lacks a disparity that \(bmx\) possesses, then \(aoy\) will have a similar advantage over \(akx\) because it will lack the same disparity relative to \(akx\). Note that it is not claimed that if \(bpy\) is optimal, then \(aoy\) is optimal. It is not necessary to tie the optimality of \(aoy\) to the optimality of \(bpy\). What is necessary is to tie the non-optimality of \(akx\) to the non-optimality of \(bmx\). To show that \(akx\) is not optimal, it is sufficient to show that \(aoy\) is more harmonic than \(akx\), whether or not \(aoy\) is optimal.

### 3.2.4 As goes bpy, so goes aoy

As explained in Section 3.2.2, input–input correspondents necessarily have the same IO correspondence fate in \(akx\) and \(bmx\): they both have the same output correspondent, or they both lack output correspondents. The definition
of candidate $aoy$ is deliberately constructed so that input–input correspondents also have the same IO correspondence fate in $aoy$ and $bpy$. However, the IO correspondence fate in $aoy$ and $bpy$ need not be the same fate as it is in $akx$ and $bmx$. This possibility is shown in the following example, where a pair of input–input correspondents both lack output correspondents in $akx$ and $bmx$, but both have the same output correspondent in $aoy$ and $bpy$. As will be explained, this is the desired outcome for the example, as it properly maintains the analogy between the two pairs of candidates.

This example is quite similar to the one used in Sections 3.2.2 and 3.2.3. The candidates $akx$ and $bmx$ are identical to that previous example, and are repeated below.

\[
\text{(2.41)} \quad in_a = /agbik/ \quad out_x = [\text{tibi}] \quad R_a = \{(a_1, x_2), (a_3, x_3), (a_4, x_4)\}
\]

\[
\text{(2.42)} \quad in_b = /tebik/ \quad out_x = [\text{tibi}] \quad R_m = \{(b_1, x_1), (b_2, x_2), (b_3, x_3), (b_4, x_4)\}
\]

\[
\text{(2.43)} \quad R_{II} = \{(a_1, b_2), (a_3, b_3), (a_4, b_4), (a_5, b_5)\}
\]

The disparities of $akx$ and $bmx$ are repeated below in (2.44) and (2.45), respectively, along with the correspondence between the disparities (2.46).

\[
\text{(2.44)} \quad _{x_1} a_1(-\text{hi}) : x_2(+\text{hi}) \quad a_1(+\text{low}) : x_2(-\text{low}) \quad a_2:_- \quad a_5:_-
\]

\[
\text{(2.45)} \quad b_2(-\text{hi}) : x_2(+\text{hi}) \quad b_5:_-
\]

\[
\text{(2.46)} \quad \text{A correspondence between the disparities of } bmx \text{ and } akx
\]

- $b_2(-\text{hi}) : x_2(+\text{hi})$ of $bmx$ corresponds to $a_1(-\text{hi}) : x_2(+\text{hi})$ of $akx$

- $b_5:_-$ of $bmx$ corresponds to $a_5:_-$ of $akx$

For this example, a different alternative output is considered, $out_y = [\text{tebig}]$. It differs from the alternative output of the previous section’s example in that it has an additional segment, $[g]$, at the end. The competitor to $bmx$, $bpy$, is summarized in (3.28). The disparity of $bpy$ is given in (3.29); it is a disparity in the value of the voicing feature.

\[
\text{(3.28)} \quad in_b = /tebik/ \quad out_y = [\text{tebig}] \quad R_p = \{(b_1, y_1), (b_2, y_2), (b_3, y_3), (b_4, y_4), (b_5, y_5)\}
\]

\[
\text{(3.29)} \quad b_5(-\text{voi}) : y_3(+\text{voi})
\]

Candidate $aoy$, relative to $bpy$, is summarized in (3.30), and its disparities are listed in (3.31).

\[
\text{(3.30)} \quad in_o = /agbik/ \quad out_y = [\text{tebig}] \quad R_o = \{(a_1, y_2), (a_3, y_3), (a_4, y_4), (a_5, y_5)\}
\]

\[
\text{(3.31)} \quad _{y_1} a_1(+\text{low}) : y_2(-\text{low}) \quad a_2:_- \quad a_5(-\text{voi}) : y_3(+\text{voi})
\]
The correspondence between the disparities of \( aoy \) and \( bpy \) are given in (3.32).

\[(3.32) \quad \text{Correspondence between the disparities of } aoy \text{ and } bpy \]

- \( a_5(-\text{voi})y_5(+\text{voi}) \) of \( aoy \) corresponds to \( b_5(-\text{voi})y_5(+\text{voi}) \) of \( bpy \)

The IO and II correspondence relations are depicted in Figure 3.3. The upper part of the figure is identical to that in Figure 3.1; the candidates \( akx \) and \( bmx \) are the same as in the previous section’s example. In the lower part of the figure, \( bpy \) differs in one respect: input segment \( b_5 \) now has an output correspondent, \( y_5 \). Because \( b_5 \) and \( a_5 \) are input–input correspondents, \( a_5 \) has output correspondent \( y_5 \) in \( aoy \).

Because segments \( a_5 \) and \( b_5 \) are input–input correspondents, the IO correspondence fate of \( a_5 \) in \( aoy \) will be the same as that of \( b_5 \) in \( bpy \). In this example, both have output correspondent \( y_5 \). The matching role property of \( a_5 \) and \( b_5 \) is still preserved from \( akx/bmx \) to \( aoy/bpy \), but the actual IO correspondence fate is different in the two cases: both segments lack output correspondents in \( akx/bmx \), while both segments have output correspondent \( y_5 \) in \( aoy/bpy \).

The difference in the IO correspondence fates between the two pairs has implications for the analogous relationships between the disparities. The relationships between the disparities of the four candidates are depicted in Figure 3.4. Note that, while \( akx \) and \( bmx \) have deletion disparities for \( a_5 \) and \( b_5 \), respectively, candidates \( aoy \) and \( bpy \) instead have identity disparities in voicing for \( a_5 \) and \( b_5 \), respectively. Because the disparities are not of the same sort (one is a deletion disparity while the other is an identity disparity), the disparities involving \( a_5 \) in \( akx \) and \( aoy \) are not analogous disparities. Similarly, the disparities involving \( b_5 \) in \( bmx \) and \( bpy \) are not analogous.
### 3.2 Relating output-driven maps to Optimality Theory

#### 3.2.5 Output-driven maps and constraints

Recall the general condition on harmonic relations for output-driven maps given in (3.18), repeated here. This condition applies to every pair of candidates \( akx \) and \( bmx \) such that \( bmx \) has greater internal similarity than \( akx \) (no assumptions are made about the grammaticality of \( akx \) and \( bmx \)). Every candidate \( bpy \) is a member of \( GEN(in_{b}) \) (as is \( bmx \)), and \( aoy \) is defined relative to each \( bpy \) as specified in (3.19).

\[
(3.18) \quad \forall bpy ((bpy \succ bmx) \rightarrow (aoy \succ akx))
\]
The entailment in (3.18) can be understood as an entailment relation between elementary ranking conditions, and (3.18) can be cast in terms of ERCs, as shown in (3.33). For any competitor \( bpy \), if a constraint ranking satisfies the ERC \([bpy \sim bmx]\) then it satisfies the ERC \([aoy \sim akx]\).

(3.33) \( \forall bpy ([bpy \sim bmx] \rightarrow [aoy \sim akx]) \)

Recall that one ERC entails another if, for each constraint, that constraint’s evaluation of the first ERC entails the constraint’s evaluation of the second. Thus, to prove that (3.33) holds for a given \( akx \) and \( bmx \), it is sufficient to show that the condition in (3.34) holds.

(3.34) \( \forall bpy \forall C \in \text{CON} \ (C[bpy \sim bmx] \rightarrow C[aoy \sim akx]) \)

Note that the condition in (3.34) applies separately to each individual constraint: each constraint can be independently evaluated for the entailment relation between its evaluations of the two ERCs. This independence is a property of ERC entailment.

The entailment condition imposed on each constraint in (3.34) can be interpreted in light of the entailment relation among the three possible evaluations that a constraint can make of a candidate comparison, \( L \rightarrow e \rightarrow W \). If \( C[bpy \sim bmx] = L \), then the entailment condition will be satisfied no matter what evaluation \( C \) assigns to \([aoy \sim akx]\). If \( C[bpy \sim bmx] = e \), then \( C[aoy \sim akx] \) must be either \( e \) or \( W \). If \( C[bpy \sim bmx] = W \), then \( C[aoy \sim akx] \) must be \( W \). The latter two cases impose non-trivial restrictions on \( C[aoy \sim akx] \) and are summarized in (3.35).

(3.35) Output-driven conditions on constraints (ERC version). Given constraint \( C \), candidates \( bmx \) with greater internal similarity than \( akx \), \( bpy \) in \( \text{GEN}(in_b) \), and \( aoy \) as defined in (3.19):
- \( C[bpy \sim bmx] = W \rightarrow C[aoy \sim akx] = W \)
- \( C[bpy \sim bmx] = e \rightarrow [C[aoy \sim akx] = e \text{ OR } C[aoy \sim akx] = W] \)

These conditions can in turn be interpreted in terms of constraint preference on candidate comparisons, using the definitions in (3.1), yielding the version of the output-driven conditions given in (3.36).

(3.36) Output-driven conditions on constraints (preference version). Given constraint \( C \), candidates \( bmx \) with greater internal similarity than \( akx \), \( bpy \) in \( \text{GEN}(in_b) \), and \( aoy \) as defined in (3.19):
- (\( C \) prefers \( bpy \) to \( bmx \)) entails (\( C \) prefers \( aoy \) to \( akx \))
- (\( C \) has no preference between \( bpy \) and \( bmx \)) entails (\( C \) either prefers \( aoy \) to \( akx \) or has no preference between them)
Constraint preferences between candidates arise as a consequence of the number of violations assessed by a constraint to each of the candidates: a constraint prefers one candidate to a second if it assesses strictly fewer violations to the first than to the second, and a constraint has no preference between two candidates if it assesses an equal number of violations to both. The constraint preference conditions in (3.36) can be translated into violation count conditions, as given in (3.37). \( C(cand) \) denotes the number of violations assessed to candidate \( cand \) by constraint \( C \).

\[
\begin{align*}
\text{(3.37) Output-driven conditions on constraints (violation counts version). Given constraint } C, \text{ candidates } bmx \text{ with greater internal similarity than } akx, bpy \text{ in } G_{\text{EN}}(in_b), \text{ and } aoy \text{ as defined in (3.19):} \\
&\bullet \ C(bpy) < C(bmx) \text{ entails } C(aoy) < C(akx) \\
&\bullet \ C(bpy) = C(bmx) \text{ entails } C(aoy) \leq C(akx)
\end{align*}
\]

I will adopt (3.34), and the equivalent form in (3.37), as a sufficient condition on constraints in the analysis below (Section 3.3.2). A constraint which satisfies the conditions in (3.37) will be labeled an \textbf{output-driven preserving} constraint.

One minor point worth mentioning: while the condition in (3.34) is sufficient to ensure the condition in (3.33), two exceptional cases prevent it from being strictly necessary. The exceptions stem from the two types of “trivial” ERCs: logically valid ERCs, and logically invalid ERCs. Logically valid ERCs contain no \( L \)s, and are satisfied under every constraint ranking. Logically invalid ERCs contain no \( W \)s and at least one \( L \), and are not satisfied under any constraint ranking. If \([aoy \sim akx]\) is logically valid, then it is entailed by anything, including \([bpy \sim bmx]\) even if there exists a constraint that does not satisfy (3.34). For example, \([L \ W \ e]\) entails \([W \ e \ W]\), even though on the second constraint \( W \) does not entail \( e \), because \([W \ e \ W]\) is logically valid. Similarly, if \([bpy \sim bmx]\) is logically invalid, then it entails everything, including \([aoy \sim akx]\) even if there exists a constraint that does not satisfy (3.34). For example, \([L \ L \ e]\) entails \([W \ e \ L]\), even though on the third constraint \( e \) does not entail \( L \), because \([L \ L \ e]\) is logically invalid.

\section{3.3 Sufficient conditions for output-driven maps}

This section pulls together conditions on Optimality Theoretic systems that are sufficient to ensure that all grammars defined in such OT systems generate only output-driven maps. Section 3.3.1 defines and discusses correspondence uniformity, the key condition on \( G_{\text{EN}} \). Section 3.3.2 briefly restates the
definition of output-driven preserving constraints, the key condition on \textit{Con}.
\textbf{Section 3.3.3} pulls the conditions together in a formal proof: all OT systems where \textit{Gen} is correspondence uniform and the constraints are output-driven preserving generate only output-driven maps.

3.3.1 \textit{Properties of Gen: correspondence uniformity}

For an OT system to define output-driven maps, \textit{Gen} must generate the relevant candidates. For instance, if \textit{Gen} generates a candidate \(akx\) with input \(\text{in}_a\) and output \(\text{out}_x\), but does not generate some candidate \(bmx\) that has greater internal similarity than \(akx\) (where \(bmx\) is contained in the reference representation space), then the map could be non-output-driven as a consequence of \textit{Gen}; the candidates required to be grammatical by the definition of an output-driven map wouldn’t be made available by \textit{Gen} (let alone be optimal).

I distinguish the RRS, the reference representation space defined in \textbf{Section 2.4.1}, from \textit{Gen} in this analysis so that the RRS can establish the analyst’s expectations about the possible behaviors of maps. The RRS can then be used to evaluate different linguistic theories, including different OT systems (which might have different definitions of \textit{Gen}). An example of an intentional distinction between the RRS and \textit{Gen} would be the harmonic serialism variant of Optimality Theory (Prince and Smolensky 1993/2004); see also McCarthy 2006 for relevant discussion. In harmonic serialism, the outputs of the set of candidates assigned to an input may only differ from the input in at most one way,\(^2\) and a derivation consists of a series of such optimizations, each using as its input the output of the previous optimization, until a form is reached which maps to itself. The “differ in at most one way” restriction applies to each step, but that restriction does not apply to the mapping between the initial input (before any steps) and the final output (after all steps are completed).\(^3\) The RRS includes representations resulting from entire derivations (arrived at by composing the mappings of the individual steps), where the output can differ from the input in many ways at once.

A \textit{Gen} function is said to be \textbf{correspondence uniform}, relative to a reference representation space RRS, if it satisfies the conditions in (3.38).

\(^2\) Such a condition might well be formalized, in the terms of the present work, as a restriction that \textit{Gen} can only generate candidates containing at most one disparity.

\(^3\) Such restrictions on \textit{Gen}, in combination with serial derivation, can result in maps (defined by the beginning and ending representations of derivations) that are not output-driven.
(3.38) Conditions for \( \text{Gen} \) to be correspondence uniform

- For each candidate \( akx \) of the RRS generated by \( \text{Gen} \), every candidate \( bmx \) in the RRS that has greater internal similarity than \( akx \) must also be generated by \( \text{Gen} \).
- For each candidate \( akx \) of the RRS generated by \( \text{Gen} \), for each \( bmx \) with greater internal similarity than \( akx \) generated by \( \text{Gen} \), for each competitor \( bpy \) generated by \( \text{Gen} \), the corresponding candidate \( aoy \) defined in (3.19) must also be generated by \( \text{Gen} \).

These requirements of \( \text{Gen} \) are fully consistent with the standard “freedom of analysis” view of \( \text{Gen} \), in which any representation in the reference representation space is generated by \( \text{Gen} \). Far from requiring anything unusual of \( \text{Gen} \), the conditions in (3.38) merely guard against unusual, possibly pathological \( \text{Gen} \) functions. The conditions focus on those conventional properties of \( \text{Gen} \) that are essential for guaranteeing output-driven maps.

A \( \text{Gen} \) function meeting the conditions in (3.38) is said to be correspondence uniform because it must possess a certain uniformity of possible correspondence. The uniformity runs along the lines of the possible types of disparities. Given that \( \text{Gen} \) generates a candidate \( akx \), \( \text{Gen} \) must also generate all candidates in the RRS with the same output and a strict subset of disparities. With correspondence uniformity, the existence in \( \text{Gen} \) of a single candidate with numerous disparities can automatically compel the existence in \( \text{Gen} \) of a whole subspace of candidates with the same output and different inputs, each input strictly eliminating some of the disparities of the original candidate. Within the limits of the RRS, a candidate with some set of disparities and a given output compels the existence of candidates with any subset of the disparities. This imposes a uniformity on possible input–output correspondences; a correspondence uniform \( \text{Gen} \) cannot arbitrarily fail to generate candidates with certain combinations of disparities.

The wide scope of correspondence uniformity becomes more apparent when you consider that the existence of competitors for one candidate requires the existence of analogous competitors for every candidate with lesser internal similarity (this is the second condition of (3.38)). If \( \text{Gen} \) generates a candidate \( bmx \) with greater internal similarity than \( akx \), the existence of competitors to \( bmx \) automatically entails the existence of analogous competitors for \( akx \). The generation of a competitor (to \( bmx \)) \( bpy \) entails the generation of the analogous competitor (to \( akx \)) \( aoy \), analogous in the sense of the definition of \( aoy \). This use of the definition of \( aoy \) imposes a uniformity to the way related inputs and outputs can correspond.
Suppose that Gen generates an identity candidate \( xq x \), a candidate with no disparities, and competitors for the input \( in_x \). Every other input form \( in_c \) that has a generated candidate with output \( out_x \) and at least one disparity must, under correspondence uniformity, also have competitors analogous to every competitor of \( xq x \). If Gen is such that every input has at least one candidate with output \( out_x \), then every competitor of \( xq x \) must have an analogous candidate for every other input: any input that can correspond to output \( x \) must also be able to correspond to any other output that input \( x \) can correspond to. Under correspondence uniformity, it can be possible for the competitors for a single identity candidate to automatically entail most (if not all) of the competitors for all other inputs.

Correspondence uniformity fits naturally with a conception of Gen in which any input can stand in correspondence with any output. However, it is also consistent with a “partitioned” Gen, with separate sets of inputs each forming candidates with separate sets of outputs. For example, if one banned insertion and deletion in Gen (as well as coalescence and breaking), then an input could only stand in correspondence with outputs containing the same number of segments as that input. Thus, all inputs of length two could stand in correspondence with all outputs of length two but none of the outputs of length three, while all of the inputs of length three could stand in correspondence with all of the outputs of length three but none of the outputs of length two, four, etc. Such a “partitioned” Gen function could still be correspondence uniform: the uniformity would exist separately within each part (one part for each input length), and any pair of inputs would either have no outputs in common (if they were of different lengths) or all of their outputs in common (if they were of the same length).

3.3.2 Properties of constraints: output-driven preserving

Let \( akx \) be a candidate with input \( in_a \) and output \( out_x \), and let \( bm x \) be a candidate with input \( in_b \) and output \( out_x \) such that \( bm x \) has greater internal similarity than \( akx \), based on correspondence \( R_{II} \) between \( in_a \) and \( in_b \). For each candidate \( bpy \), with input \( in_b \) and output \( out_y \), let \( aoy \) denote the candidate with input \( in_a \), output \( out_y \), and correspondence \( R_o \) as defined in (3.19). A constraint C is output-driven preserving (ODP) if it has the properties previously listed in (3.37), for every such pair \( akx \) and \( bm x \), and every competitor \( bpy \) of each \( bm x \). The properties are separately labeled here as (3.39) and (3.40).

\[(3.39) \quad C(bpy) < C(bmx) \text{ entails } C(aoy) < C(akx)\]

\[(3.40) \quad C(bpy) = C(bmx) \text{ entails } C(aoy) \leq C(akx)\]
3.3 Sufficient conditions for output-driven maps

3.3.3 Proof of sufficient conditions for output-driven maps

Suppose an Optimality Theoretic system has a Gen function that is correspondence uniform, and all of the constraints are output-driven preserving. Then all grammars realizable in that system define output-driven maps.

Proof

Let M be the map defined by an arbitrary grammar in the OT system. Without loss of generality, let akx (with input ina and output outx) be any candidate that is optimal for M, and let bmx be any candidate in the RRS that has greater internal similarity than akx. Because Gen is correspondence uniform, it must generate bmx. To prove that M is output-driven, it is sufficient to prove that bmx is at least as harmonic as any candidate for input inb: for all candidates bpy, bmx ≥ bpy.

The proof is by contradiction. Suppose, to the contrary, that there exists a candidate bpy such that bpy > bmx, that is, such that bpy is strictly more harmonic than bmx. It will be shown that this unavoidably leads to a contradiction. Note that no particular commitment is made here about the input–output correspondence relation in bpy.

Let aoy be as defined in (3.19). Because Gen is correspondence uniform, it must generate aoy. Candidate akx is optimal, so it must be the case that akx ≥ aoy. Let Cp be the highest-ranked constraint with a preference in at least one of the two candidate comparisons, akx ≥ aoy and bpy > bmx. Note that the comparison bpy > bmx requires that at least one constraint distinguish them, so Cp must exist. All constraints of the system are ODP, including Cp, meaning that Cp has the properties given in (3.39) and (3.40).

With respect to the comparison of Cp(bpy) and Cp(bmx), there are three cases to consider that exhaust all possibilities. Because Cp is the highest-ranked constraint with a preference in at least one of the two comparisons, if Cp has a preference between bpy and bmx (cases 1 and 2), then it is the highest constraint with a preference between bpy and bmx, and thus determines the relative harmony of bpy and bmx.

**Case 1:** Cp(bmx) < Cp(bpy)

This entails that bmx > bpy, directly contradicting the hypothesis that bpy ≥ bmx. **Contradiction.**

4 The proof technique used here, of reasoning from the highest-ranked constraint with a preference in at least one of two related candidate comparisons, is adapted from the proof technique used by Tesar (2006b) to reason about contrast.
Case 2: \( C_p(bpy) < C_p(bmx) \)

By (3.39), this entails that \( C_p(aoy) < C_p(akx) \). That in turn entails that \( aoy > akx \), directly contradicting the premise that \( akx \) is optimal. **Contradiction.**

Case 3: \( C_p(bpy) = C_p(bmx) \)

By (3.40), this entails that \( C_p(aoy) \leq C_p(akx) \). By definition, \( C_p \) has a preference between the candidates of at least one of the pairs \( (bpy, bmx) \) and \( (aoy, akx) \). Because \( C_p \) does not have a preference in the comparison between \( bpy \) and \( bmx \) in this case, it must have a preference between \( aoy \) and \( akx \), and therefore it must be the case that \( C_p(aoy) < C_p(akx) \). This in turn entails that \( aoy > akx \), directly contradicting the premise that \( akx \) is optimal. **Contradiction.**

All possibilities have resulted in contradiction, so the hypothesis, that there exists a candidate \( bpy \) such that \( bpy > bmx \), must be false. It follows that \( bmx \) must be an optimal candidate.

**End of Proof**

### 3.4 Basic constraints: overview of the results

This section gives a quick overview of basic OT constraints, indicating which ones are output-driven preserving (ODP), and an intuitive sense of why. The proofs in Sections 3.5 and 3.6 provide much more rigorous arguments that \( \text{Max} \), \( \text{Dep} \), and \( \text{Ident} \) are ODP.

#### 3.4.1 Terminology: faithfulness and input-referring constraints

The correspondence between input and output in candidates is normally evaluated by constraints commonly referred to as faithfulness constraints (McCarthy and Prince 1999, Prince and Smolensky 1993/2004). The conventional view is that faithfulness constraints are violated by various failures of the output of a candidate to be faithful to the input of the candidate. Instances of “failure to be faithful” can be viewed as disparities, and many common faithfulness constraints are violated by particular classes of disparities. However, there is another use of the term “faithfulness constraint,” one that refers to any constraint which makes reference to the input. More generally, the term “faithfulness constraint” is used to refer to any constraint which evaluates a correspondence between two forms (usually by penalizing disparities between the forms), be it input–output (McCarthy and Prince 1999), base-reduplicant (McCarthy and Prince 1999), output–output (Benua 1997), or some other correspondence. Nomenclature reflecting distinctions among dimensions of correspondence
often involve suffixes on constraint names, such as $\text{Max-IO}$ (input–output), $\text{Max-BR}$ (base–reduplicant), and $\text{Max-OO}$ (output–output).

To avoid confusion, in this chapter the term **input-referring constraints** will frequently be used, meaning constraints that make reference of any kind to the input or to the input–output correspondence relation. Input-referring constraints are of particular interest to the investigation of output-driven maps, whether they penalize specific classes of disparities or other sorts of correspondence configurations. However, the term “faithfulness” is ubiquitous in the OT literature, and the vast majority of proposed input-referring constraints penalize disparities, so “faithfulness” will be the predominant term in subsequent chapters.

### 3.4.2 Markedness constraints

A markedness constraint is one that evaluates a candidate solely on the basis of the output form of the candidate, ignoring the input and the IO correspondence. All markedness constraints are trivially ODP. If $C_m$ is a markedness constraint and $C_m(bpy) < C_m(bmx)$, then any candidate with output $out_y$ will have fewer violations of $C_m$ than any candidate with output $out_x$, and thus $C_m(aoy) < C_m(akx)$. The analogous observation holds whenever $C_m(bpy) = C_m(bmx)$.

It is constraints other than markedness that must be further scrutinized to determine if they are ODP.

### 3.4.3 Value-independent input-referring constraints

The basic set of input-referring constraints for correspondence faithfulness proposed by McCarthy and Prince (1995) are ODP. They are $\text{Max}$, $\text{Dep}$, and $\text{Ident}[F]$ (for each segmental feature $F$). Proofs that these constraints are ODP can be found in Section 3.6. These constraints evaluate the basic types of disparity adopted in (2.37). $\text{Max}$ is violated by any deletion disparity: an input segment with no output correspondent. $\text{Dep}$ is violated by any insertion disparity: an output segment with no input correspondent. $\text{Ident}[F]$ is violated by any identity disparity for feature $F$: input–output corresponding segments with non-identical values for the feature $F$.

Intuitively, one can understand the ODP status of these constraints to follow from the relationships of disparities between ($bpy$ and $bmx$) and ($aoy$ and $akx$). If $\text{Max}$ (for instance) prefers $bpy$ to $bmx$, it must be because $bpy$ has fewer deletion disparities. Thus, there must be deletion disparities that $bpy$ lacks relative to $bmx$. The definition of $aoy$ is constructed so that $aoy$ should also lack, relative to $akx$, disparities corresponding to the ones that $bpy$ lacks relative...
to \textit{bmx}. Thus, if \textit{bpy} ends up with fewer deletion disparities than \textit{bmx}, \textit{aoy} will end up with fewer deletion disparities than \textit{akx}, and therefore \textsc{Max} will also prefer \textit{aoy} to \textit{akx}.

Elaborating slightly on the deletion disparity case, recall that every disparity in \textit{bmx} has a corresponding disparity in \textit{akx}, by the definition of greater internal similarity. Therefore, every deleted input segment in \textit{bmx} has an input–input correspondent that is deleted in \textit{akx}. Since \textit{bpy} has fewer violations of \textsc{Max}, it must be the case that at least one of the segments that is deleted in \textit{bmx} has an output correspondent in \textit{bpy}. By the definition of \textsc{aoy}, the input–input correspondent of each such segment is assigned the same output correspondent segment in \textit{aoy}. Because those segments (in input \textit{in\textsubscript{a}}) are input–input correspondents of segments (in input \textit{in\textsubscript{b}}) that are deleted in \textit{bmx}, those segments (in input \textit{in\textsubscript{b}}) are deleted in \textit{akx}, and thus constitute deletion disparities (and \textsc{Max} violations) that \textit{akx} has but \textit{aoy} does not.

### 3.4.4 Value-restricted input-referring constraints

Input-referring constraints include what will here be called \textbf{value-restricted} constraints, where the constraint is violated by a type of disparity only if distinguished segments of the disparity have one of a constraint-specified set of values for a particular feature. A value-restricted \textsc{Ident} constraint only evaluates IO correspondents for agreement on the value of a feature if one of the corresponding segments has one of a specific set of values for the feature. Pater (1999) first proposed constraints that were like \textsc{Ident}, but restricted to correspondences in which the input segment had a particular value. Pater used the notation \textsc{Ident} \textsubscript{I} \textarrow{O}[\alpha\textsc{F}] for such constraints. An example is \textsc{Ident} \textsubscript{I} \textarrow{O}[	extsc{Nas}] (where 	extsc{Nas} is equivalent to +nasal), which is violated if an input segment is +nasal and its output correspondent is not. Pater also discussed analogous constraints, conditioned on the value of the output correspondent for feature \textsc{F}, which he labeled \textsc{Ident} \textsubscript{O} \textarrow{I}[\alpha\textsc{F}].

De Lacy (2002) proposed \textsc{Ident} constraints restricted to correspondences in which the input segment had a value belonging to a subset of the possible values for a feature, in the context of markedness scales and scale category conflation. Such constraints are here labeled \textsc{Ident}\textsubscript{in\textarrow{V}}: \textsc{Ident}\textsubscript{in\textarrow{V}} is violated by any pair of IO correspondents such that the input correspondent’s value of feature \textsc{F} is a member of the set of values \textit{V}, and the output correspondent’s value for \textsc{F} is different from the input correspondent’s value for \textsc{F} (regardless of whether the output correspondent’s value for \textsc{F} is a member of \textit{V}). Analogous constraints restricting evaluation to pairs where the output correspondent’s value of \textsc{F} is a member of \textit{V} can also be defined, and are here labeled \textsc{Ident}\textsubscript{out\textarrow{V}}.
The class of value-restricted \textsc{Ident} constraints can be understood as a generalization of the idea of \textsc{Ident} constraints, one which includes equivalent forms of \textsc{Ident}[F], \textsc{Ident}I → O[αF] and \textsc{Ident}O → I[αF]. \textsc{Ident}[F] can be understood as \textsc{Ident}[F_{in} ∈ V], where V is equal to the set of all possible values for F. \textsc{Ident}I → O[αF] can be understood as \textsc{Ident}[F_{in} ∈ V], where V contains only the single specified value for F.

Value-restricted \textsc{Ident} constraints are ODP; they cannot induce non-output-driven maps. A proof of this is given for \textsc{Ident}[F_{in} ∈ V] in Section 3.6.4, and for \textsc{Ident}[F_{out} ∈ V] in Section 3.6.5. Because \textsc{Ident}[F_{in} ∈ V] is a generalization of \textsc{Ident}I → O[αF], \textsc{Ident}[F_{out} ∈ V] is a generalization of \textsc{Ident}O → I[αF], and both are generalizations of \textsc{Ident}[F], these two proofs cover all of those constraints as well.

Value-restricted \textsc{Max} and \textsc{Dep} constraints, on the other hand, are in general not ODP. This is shown in detail in the next chapter, in Sections 4.4.4 and 4.5.1. Note that these classes include constraints actually proposed in the literature, such as \textsc{Max}[C] and \textsc{Max}[V].

### 3.5 Analysis of relationships between disparities

This section provides a complete analysis of the possible disparities of candidates \(akx\), \(bmx\), \(aoy\), and \(bpy\), and their relationships to each other. Various results of this analysis will be used in the constraint-specific proofs that follow in Section 3.6.

#### 3.5.1 The set-up

Let \(akx\) be a candidate with input \(ina\) and output \(out_{x}\), let \(bmx\) be a candidate with input \(in_{b}\) and output \(out_{x}\), and let \(R_{II}\) be a correspondence relation between \(ina\) and \(in_{b}\) such that \(bmx\) has greater internal similarity than \(akx\). Let \(bpy\) be a candidate with input \(in_{b}\) and output \(out_{y}\). For a constraint to be ODP, it must be the case that for any such \(akx\), \(bmx\), \(R_{II}\), and \(bpy\), the corresponding candidate \(aoy\) satisfies the conditions (3.39) and (3.40) with respect to the constraint. The candidate \(aoy\) was defined in Section 3.2.2 with the IO correspondence relation \(R_{o}\) given in (3.20), repeated here.

\begin{equation}
\text{(3.20) } s_{a}R_{o}s_{y} \iff \exists s_{b} \left[ s_{a}R_{II}s_{b} \text{ and } s_{b}R_{p}s_{y} \right]
\end{equation}

The conditions (3.39) and (3.40) that \(aoy\) must satisfy in order for a constraint to be ODP are repeated here.

\begin{align*}
(3.39) & \quad C(bpy) < C(bmx) \text{ entails } C(aoy) < C(akx) \\
(3.40) & \quad C(bpy) = C(bmx) \text{ entails } C(aoy) \leq C(akx)
\end{align*}
The correspondence between the disparities of $akx$ and $bmx$ defined in (2.40), repeated below, is based upon the input–input correspondence $RII$ supporting the claim that $bmx$ has greater internal similarity than $akx$, and the fact that the two candidates share the same output. The same definition can be used to construct a correspondence between the disparities of $aoy$ and $bpy$, using the same input–input correspondence $RII$.

(2.40) Constructing a correspondence between the disparities of $akx$ and $bmx$, given input–input correspondence $RII$.

- Let $sb:_-$ be a deletion disparity in $bmx$. This disparity has a corresponding disparity $sa:_-$ in $akx$ if and only if $sb$ has input–input correspondent $sa$ in $akx$ (and thus $sa$ necessarily has no output correspondent in $akx$, by the conditions on input–input correspondence).
- Let $:_-sx$ be an insertion disparity in $bmx$. This disparity has a corresponding disparity $:_-sx$ in $akx$ if and only if $sx$ has no input correspondent in $akx$.
- Let $sx$ be an output segment of $bmx$ with an input correspondent $sb$ such that $sb$ and $sx$ differ on the value of feature F. This disparity in $bmx$ has a corresponding disparity in $akx$ if and only if $sx$ has an input correspondent $sa$ in $akx$ such that $sa$ and $sx$ differ on the value of feature F ($sb$ and $sa$ are then necessarily input–input correspondents by the conditions on input–input correspondence).

The definition of analogous disparities between $akx$ and $aoy$, given in (3.26), is repeated here. The candidates being related in this case share the same input form ($ina$). The shared input is the basis for the relation between deletion disparities in $akx$ and $aoy$, and between identity disparities in $akx$ and $aoy$. The relation between insertion disparities in $akx$ and $aoy$ is based on the treatment of the corresponding output segments in $bmx$ and $bpy$.

(3.26) General definition of analogous disparities

- Let $sa:_-$ be a deletion disparity of $aoy$. This disparity in $aoy$ has an analogous deletion disparity $sa:_-$ in $akx$ if and only if $sa$ has no output correspondent in $akx$.
- Let $:_-sy$ be an insertion disparity in $aoy$. This disparity has an analogous insertion disparity $:_-sy$ in $akx$ if and only if $sy$ has an input correspondent $sb$ in $bpy$, and $sb$ has output correspondent $sx$ in $bmx$.
- Let $sa(\alpha):sy(\beta)$ be an identity disparity of $aoy$ for feature F ($\alpha \neq \beta$). This disparity in $aoy$ has an analogous identity disparity in $akx$ if and only if $sa$ has an output correspondent $sx$ in $akx$ such that $sa$ and $sx$ differ on the value of feature F.

The motivation for the definition of $aoy$ lies in the relations between the disparities of the four candidates $akx$, $bmx$, $aoy$, and $bpy$. It is shown that every
disparity of $a_{oy}$ either has a corresponding disparity in $b_{py}$ or has an analogous disparity in $a_{kx}$. It is further shown that every disparity of $a_{oy}$ that lacks a corresponding disparity in $b_{py}$ has an analogous disparity in $a_{kx}$ that has no corresponding disparity in $b_{mx}$.

3.5.2 Deletion disparities

Input segments with no input–input correspondents
Segments of $i_{na}$ with no input–input correspondent must have no output correspondent in $a_{kx}$. By the definition of $R_o$, segments of $i_{na}$ with no input–input correspondent must have no output correspondent in $a_{oy}$. By the definition of correspondence between disparities (2.40), a deletion disparity involving a segment with no input–input correspondent cannot have a corresponding disparity.

(3.41) Each deletion disparity $s_{a_{i-}}$ in $a_{kx}$, where $s_a$ has no input-input correspondent, has an identical analogous disparity $s_{a_{i-}}$ in $a_{oy}$, and no corresponding disparity in $b_{mx}$.

(3.42) Each deletion disparity $s_{a_{i-}}$ in $a_{oy}$, where $s_a$ has no input-input correspondent, has an identical analogous disparity $s_{a_{i-}}$ in $a_{kx}$, and no corresponding disparity in $b_{py}$.

Combining (3.41) and (3.42) yields the result in (3.43).

(3.43) Each deletion disparity $s_{a_{i-}}$ in $a_{oy}$, where $s_a$ has no input-input correspondent, has an identical analogous disparity $s_{a_{i-}}$ in $a_{kx}$ with no corresponding disparity in $b_{mx}$.

Each segment of $i_{nb}$ with no input–input correspondent must have an identical output correspondent in $b_{mx}$, and therefore cannot be part of a deletion disparity in $b_{mx}$.

(3.44) There are no deletion disparities $s_{b_{i-}}$ in $b_{mx}$ such that $s_b$ has no input–input correspondent.

Each deletion disparity of $b_{py}$ for a segment $s_b$ with no input–input correspondent has no corresponding disparity in $a_{oy}$, because there is no input–input correspondent for $s_b$.

(3.45) Each deletion disparity $s_{b_{i-}}$ in $b_{py}$, where $s_b$ has no input–input correspondent, has no corresponding disparity in $a_{oy}$.

Note that each such deletion disparity has no counterpart in $b_{mx}$, because segment $s_b$ cannot be part of a deletion disparity in $b_{mx}$.
Input segments with input–input correspondents
A segment $s_a$ with input–input correspondent $s_b$ has no output correspondent in $akx$ if and only if ($s_a$ has no output correspondent in $bmx$ and $s_a = s_b$), by the definition of greater internal similarity. Every such deletion disparity $s_a:_$ in $akx$ has a corresponding disparity $s_b:_$ in $bmx$, and vice-versa. Because $s_a = s_b$, each corresponding pair of disparities is identical.

(3.46) Each deletion disparity $s_a:_$ of $akx$, where $s_a$ has input–input correspondent $s_b$, has an identical corresponding disparity $s_b:_$ in $bmx$.

(3.47) Each deletion disparity $s_b:_$ of $bmx$, where $s_b$ has input–input correspondent $s_a$, has an identical corresponding disparity $s_a:_$ in $akx$.

A segment $s_a$ with input–input correspondent $s_b$ has no output correspondent in $aoy$ if and only if $s_b$ has no output correspondent in $bpy$, by the definition of $R_o$. Every such deletion disparity $s_a:_$ in $aoy$ has a corresponding disparity $s_b:_$ in $bpy$, and vice-versa. However, the corresponding disparities will be non-identical when $s_a \neq s_b$. Note that when $s_a$ and $s_b$ are input–input correspondents, $s_a \neq s_b$ only when both $s_a$ and $s_b$ have an output correspondent $s_x$ in $akx$ and $bmx$, respectively.

(3.48) Each deletion disparity $s_a:_$ of $aoy$, where $s_a$ has input–input correspondent $s_b$, and $s_a = s_b$, has an identical corresponding disparity $s_b:_$ in $bpy$.

(3.49) Each deletion disparity $s_a:_$ of $aoy$, where $s_a$ has input–input correspondent $s_b$, and $s_a \neq s_b$, has a non-identical corresponding disparity $s_b:_$ in $bpy$.

(3.50) Each deletion disparity $s_b:_$ of $bpy$, where $s_b$ has input–input correspondent $s_a$, and $s_a = s_b$, has an identical corresponding disparity $s_a:_$ in $aoy$.

(3.51) Each deletion disparity $s_b:_$ of $bpy$, where $s_b$ has input–input correspondent $s_a$, and $s_a \neq s_b$, has a non-identical corresponding disparity $s_a:_$ in $aoy$.

Each deletion disparity of $aoy$ either has a corresponding disparity in $bpy$ or an analogous disparity in $akx$. There is the possibility of overlap. Given an input segment $s_a$ with input–input correspondent $s_b$ such that $s_a:_$ is a disparity of both $akx$ and $aoy$, $s_b:_$ must be a disparity of both $bmx$ and $bpy$. In that case, $s_a:_$ in $aoy$ has both a corresponding disparity in $bpy$ and an analogous disparity in $akx$.

3.5.3 Insertion disparities

Insertion disparities in candidates $bmx$ and $bpy$
By the definition of greater internal similarity, every disparity of $bmx$ has an identical corresponding disparity in $akx$. Therefore, each segment $s_x$ of $out_x$
that lacks an input correspondent in $bmx$ has a corresponding disparity in $akx$. The corresponding disparities are identical, each involving the same output segment $s_x$.

(3.52) Each insertion disparity $:_s_x$ of $bmx$ has an identical corresponding disparity in $akx$.

By the definition of $aoy$, any segment $s_y$ of $out_y$ that lacks an input correspondent in $bpy$ must also lack an input correspondent in $aoy$. Therefore, each insertion disparity in $bpy$ has a corresponding disparity in $aoy$. The corresponding disparities are identical, each involving the same output segment $s_y$.

(3.53) Each insertion disparity $:_s_y$ of $bpy$ has an identical corresponding disparity in $aoy$.

**Insertion disparities in candidates $akx$ and $aoy$**

Because every insertion disparity in $bmx$ has a corresponding disparity in $akx$, it remains to account for any insertion disparities in $akx$ with no corresponding disparity in $bmx$. For each disparity $:_s_x$ of $akx$ with no corresponding disparity in $bmx$, it must be the case that $s_x$ has an identical input correspondent $s_b$ in $bmx$, such that $s_b$ has no input–input correspondent. If $s_b$ has no output correspondent in $bpy$, then the disparity $:_s_x$ in $akx$ has no analog in $aoy$. If $s_b$ has an output correspondent $s_y$ in $bpy$, then $s_y$ has no input correspondent in $aoy$, because $s_b$ has no input–input correspondent. Therefore, the disparity $:_s_y$ in $aoy$ will be analogous to the disparity $:_s_x$ in $akx$.

Because every insertion disparity in $bpy$ has a corresponding disparity in $aoy$, it remains to account for any insertion disparities in $aoy$ with no corresponding disparity in $bpy$. For each disparity $:_s_y$ of $aoy$ with no corresponding disparity in $bpy$, it must be the case that $s_y$ has an input correspondent $s_b$ in $bpy$. By the definition of $R$, $s_b$ must not have an input–input correspondent. Therefore, $s_b$ must have an identical output correspondent $s_x$ in $bmx$, and $s_x$ must have no input correspondent in $akx$. The disparity $:_s_y$ in $aoy$ is therefore analogous to the disparity $:_s_x$ in $akx$. Because $s_x$ has input correspondent $s_b$ in $bmx$, $:_s_x$ in $akx$ has no corresponding disparity in $bmx$.

The analogous disparities $:_s_x$ in $akx$ and $:_s_y$ in $aoy$ are possibly not identical, as there is nothing to require that $s_x = s_y$.

(3.54) Each insertion disparity $:_s_x$ of $akx$, where $:_s_x$ has no corresponding disparity in $bmx$, $s_x$ has input correspondent $s_b$ in $bmx$, and $s_b$ has no output correspondent in $bpy$, has no analogous disparity in $aoy$. 
Each insertion disparity $\_:\_s_y$ of $aoy$, where $\_:\_s_y$ has no corresponding disparity in $bpy$, $s_y$ has input correspondent $s_b$ in $bpy$, $s_b$ has output correspondent $s_x$ in $bmx$, and $s_x = s_y$, has an **identical** analogous disparity $\_:\_s_x$ in $akx$, where $\_:\_s_x$ in $akx$ lacks a corresponding disparity in $bmx$.

Each insertion disparity of $aoy$ either has a corresponding disparity in $bpy$ or an analogous disparity in $akx$.

### 3.5.4 Identity disparities

In order for a pair of identity disparities to correspond, the output segments of the disparities must be the same segment, and thus the input segments of the disparities must be input–input correspondents. In order for a pair of identity disparities to be analogous, the input segments of the disparities must be the same segment (we don’t concern ourselves with output–output correspondence for $akx \sim aoy$).

**Input segments without input–input correspondents**

If a segment $s_a$ of $in_a$ has no input–input correspondent, then it has no output correspondent in $akx$, and thus cannot participate in an identity disparity in $akx$. By the definition of $aoy$, if $s_a$ has no input–input correspondent then it has no output correspondent in $aoy$, and thus cannot participate in an identity disparity in $aoy$.

(3.57) No identity disparities of $akx$ involve an input segment $s_a$ without an input–input correspondent.

(3.58) No identity disparities of $aoy$ involve an input segment $s_a$ without an input–input correspondent.

If a segment $s_b$ of $in_b$ has no input–input correspondent, then it must have an identical output correspondent in $bmx$, and thus cannot participate in an identity disparity in $bmx$. If $s_b$ has a non-identical output correspondent in $bpy$, the resulting identity disparities have no correspondents in $aoy$ ($s_b$ has no input–input correspondent).

(3.59) No identity disparities of $bmx$ involve an input segment $s_b$ without an input–input correspondent.

(3.60) Each identity disparity $s_b(\alpha):s_y(\beta)$ in $bpy$, where $s_b$ has no input–input correspondent, has no corresponding disparity in $aoy$. 

Note that each such identity disparity in \( b_p y \) has no correspondent in \( b_m x \) (\( s_b \) is identical to its output correspondent in \( b_m x \)).

**Input segments with input–input correspondents**

If \( s_a \) and \( s_b \) are input–input correspondents, they must have the same output correspondent \( s_x \) in \( a_k x \) and \( b_m x \), respectively, in order to possibly have any identity disparities in either of those candidates. Similarly, they must have the same output correspondent \( s_y \) in \( a_o y \) and \( b_p y \), respectively, in order to possibly have any identity disparities in either of those candidates. There are three kinds of situations of interest:

- \( s_b \) has an output correspondent in \( b_m x \), but not in \( b_p y \).
- \( s_b \) has an output correspondent in \( b_p y \), but not in \( b_m x \).
- \( s_b \) has an output correspondent in both \( b_m x \) and \( b_p y \).

Each of the three will be considered in turn.

**\( s_b \) has an output correspondent in \( b_m x \), but not in \( b_p y \)**

In this case, the possible values for feature F that could be assigned to corresponding segments \( s_a, s_b, \) and \( s_x \) are considered. What matters for purposes of distinguishing disparities is whether or not two values for feature F are the same. In the tables that follow, distinct Greek letters signify necessarily distinct values for feature F.

\[(3.61)\]

<table>
<thead>
<tr>
<th>(F(s_a))</th>
<th>(F(s_b))</th>
<th>(F(s_x))</th>
<th>(a_k x)</th>
<th>(b_m x)</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\alpha)</td>
<td>(\alpha)</td>
<td>(\alpha)</td>
<td>(\alpha \to \alpha)</td>
<td>(\alpha \to \alpha)</td>
<td>no disparities</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>(\alpha)</td>
<td>(\gamma)</td>
<td>(\alpha \to \gamma)</td>
<td>(\alpha \to \gamma)</td>
<td>corresponding disparities (a_k x:b_m x)</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>(\beta)</td>
<td>(\alpha)</td>
<td>(\alpha \to \alpha)</td>
<td>(\beta \to \alpha)</td>
<td>NOT POSSIBLE, by def. of greater internal similarity</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>(\beta)</td>
<td>(\beta)</td>
<td>(\alpha \to \beta)</td>
<td>(\beta \to \beta)</td>
<td>(a_k x) disparity has no correspondent in (b_m x)</td>
</tr>
<tr>
<td>(\alpha)</td>
<td>(\beta)</td>
<td>(\gamma)</td>
<td>(\alpha \to \gamma)</td>
<td>(\beta \to \gamma)</td>
<td>NOT POSSIBLE, by def. of greater internal similarity</td>
</tr>
</tbody>
</table>

The logical possibilities are listed in (3.61). Recall that, in the present case, \( s_a \) and \( s_b \) are input–input correspondents, and they have output correspondent \( s_x \) in \( a_k x \) and \( b_m x \), respectively. Two of the rows are shaded; these rows represent cases that do not satisfy the requirement that \( b_m x \) have greater internal similarity than \( a_k x \). In the first shaded row, \( b_m x \) has disparity \( s_b(\beta):s_x(\alpha) \) on feature F,
while \( \text{akx} \) has no corresponding disparity. In the second shaded row, both \( \text{akx} \) and \( \text{bmx} \) have disparities on feature F for the corresponding segments, but the disparities are not identical, and therefore cannot be corresponding disparities, due to different values for feature F in \( s_a \) and \( s_b \): \( \text{akx} \) has disparity \( s_a(\alpha):s_x(\gamma) \), while \( \text{bmx} \) has disparity \( s_b(\beta):s_x(\gamma) \).

In the first row of (3.61), there are no disparities: both \( s_a \) and \( s_b \) have the same value for feature F as their output correspondent \( s_x \). In the second row, \( s_a \) and \( s_b \) have the same value for feature F, and \( \text{akx} \) and \( \text{bmx} \) have (identical) corresponding identity disparities.

In the fourth row, \( \text{akx} \) has a disparity, \( s_a(\alpha):s_x(\beta) \), with no correspondent in \( \text{bmx} \). In the present case, \( s_b \) has an output correspondent in \( \text{bmx} \), but not in \( \text{bpy} \). Because \( s_b \) lacks an output correspondent in \( \text{bpy} \), by the definition of \( aoy \), \( s_a \) has no output correspondent in \( aoy \). Therefore, the disparity \( s_a(\alpha):s_x(\beta) \) in \( \text{akx} \) has no analogous identity disparity in \( aoy \).

(3.62) Each identity disparity \( s_a(\alpha):s_x(\beta) \) in \( \text{akx} \) without a corresponding disparity in \( \text{bmx} \), where \( s_x \) has input–input correspondent \( s_b \) and \( s_b \) has no output correspondent in \( \text{bpy} \), has no analogous disparity in \( aoy \).

**\( s_b \) has an output correspondent in \( \text{bpy} \), but not in \( \text{bmx} \)**

In this case, the possible values for feature F that could be assigned to corresponding segments \( s_a, s_b, \) and \( s_y \) are considered. Since \( s_a \) and \( s_b \) are input–input correspondents, and \( s_b \) has no output correspondent in \( \text{bmx} \), it must be the case (by the definition of greater internal similarity) that \( s_a \) has no output correspondent in \( \text{akx} \), and further that \( s_a = s_b \). Thus, none of the combinations in which \( s_a \) and \( s_b \) have differing values for F are possible; these are the three shaded rows at the bottom of (3.63).

(3.63) Identity disparities when \( s_b \) has an output correspondent in \( \text{bpy} \), but not in \( \text{bmx} \).

<table>
<thead>
<tr>
<th>F(( s_a ))</th>
<th>F(( s_b ))</th>
<th>F(( s_y ))</th>
<th>( aoy )</th>
<th>( bpy )</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha )</td>
<td>( \alpha )</td>
<td>( \alpha )</td>
<td>( \alpha \rightarrow \alpha )</td>
<td>( \alpha \rightarrow \alpha )</td>
<td>no disparities</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>( \alpha )</td>
<td>( \delta )</td>
<td>( \alpha \rightarrow \delta )</td>
<td>( \alpha \rightarrow \delta )</td>
<td>identical corresponding disparities ( aoy:bpy )</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>( \beta )</td>
<td>( \alpha )</td>
<td>( \alpha \rightarrow \alpha )</td>
<td>( \beta \rightarrow \alpha )</td>
<td>NOT POSSIBLE, by def. of greater internal similarity</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>( \beta )</td>
<td>( \beta )</td>
<td>( \alpha \rightarrow \beta )</td>
<td>( \beta \rightarrow \beta )</td>
<td>NOT POSSIBLE, by def. of greater internal similarity</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>( \beta )</td>
<td>( \delta )</td>
<td>( \alpha \rightarrow \delta )</td>
<td>( \beta \rightarrow \delta )</td>
<td>NOT POSSIBLE, by def. of greater internal similarity</td>
</tr>
</tbody>
</table>
In the first row of (3.63), there are no disparities: both \( s_a \) and \( s_b \) have the same value for feature F as their output correspondent \( s_y \). In the second row, \( s_a \) and \( s_b \) have the same value for feature F, and \( aoy \) and \( bpy \) have identical corresponding identity disparities. Because \( s_a \) has no output correspondent in \( akx \), the disparity \( s_a(\alpha):s_y(\delta) \) in \( aoy \) has no analogous disparity in \( akx \).

\[
\begin{align*}
(3.64) & \quad \text{Each identity disparity } s_a(\alpha):s_y(\delta) \text{ in } aoy, \text{ where } s_a \text{ has input–input correspondent } s_b \text{ and } s_b \text{ has no output correspondent in } bmx, \text{ has an identical corresponding disparity in } bpy, \text{ and has no analogous disparity in } akx.
\end{align*}
\]

**\( s_b \) has an output correspondent in both \( bmx \) and \( bpy \)**

In this case, the possible values for feature F that could be assigned to segments \( s_a, s_b, sx, \) and \( sy \) are considered, where \( s_a \) and \( s_b \) are input–input correspondents with output correspondents \( sx \) in both \( akx \) and \( bmx \), and \( s_b \) has output correspondent \( s_y \) in \( bpy \). It follows from the definition of \( aoy \) that \( s_a \) has output correspondent \( s_y \) in \( aoy \).

\[
(3.65) \quad \text{Identity disparities when } s_b \text{ has an output correspondent in both } bmx \text{ and } bpy.
\]

<table>
<thead>
<tr>
<th>( F(s_a) )</th>
<th>( F(s_b) )</th>
<th>( F(s_x) )</th>
<th>( akx )</th>
<th>( bmx )</th>
<th>( bpy )</th>
<th>( aoy )</th>
<th>Observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \alpha )</td>
<td>( \alpha )</td>
<td>( \alpha )</td>
<td>( \alpha \rightarrow \alpha )</td>
<td>( \alpha \rightarrow \alpha )</td>
<td>( \alpha \rightarrow \alpha )</td>
<td>( \alpha \rightarrow \alpha )</td>
<td>no disparities</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>( \alpha )</td>
<td>( \delta )</td>
<td>( \alpha \rightarrow \alpha )</td>
<td>( \alpha \rightarrow \delta )</td>
<td>( \alpha \rightarrow \delta )</td>
<td>identical corresp. disparities ( aoy:bpy )</td>
<td></td>
</tr>
<tr>
<td>( \alpha )</td>
<td>( \alpha )</td>
<td>( \gamma )</td>
<td>( \alpha \rightarrow \gamma )</td>
<td>( \alpha \rightarrow \gamma )</td>
<td>( \alpha \rightarrow \alpha )</td>
<td>( \alpha \rightarrow \gamma )</td>
<td>identical corresp. disparities ( akx:bmx )</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>( \gamma )</td>
<td>( \gamma )</td>
<td>( \alpha \rightarrow \gamma )</td>
<td>( \alpha \rightarrow \gamma )</td>
<td>( \alpha \rightarrow \gamma )</td>
<td>( \alpha \rightarrow \gamma )</td>
<td>identical corresp. disparities ( akx:bmx, aoy:bpy )</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>( \gamma )</td>
<td>( \delta )</td>
<td>( \alpha \rightarrow \gamma )</td>
<td>( \alpha \rightarrow \delta )</td>
<td>( \alpha \rightarrow \delta )</td>
<td>identical corresp. disparities ( akx:bmx, aoy:bpy )</td>
<td></td>
</tr>
<tr>
<td>( \alpha )</td>
<td>( \beta )</td>
<td>( \beta )</td>
<td>( \alpha \rightarrow \beta )</td>
<td>( \beta \rightarrow \beta )</td>
<td>( \beta \rightarrow \alpha )</td>
<td>( \alpha \rightarrow \alpha )</td>
<td>( akx ) has no corresp. disparity in ( bmx ), ( bpy ) has no corresp. disparity in ( aoy )</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>( \beta )</td>
<td>( \beta )</td>
<td>( \alpha \rightarrow \beta )</td>
<td>( \beta \rightarrow \beta )</td>
<td>( \beta \rightarrow \beta )</td>
<td>( \alpha \rightarrow \beta )</td>
<td>( aoy ) has no corresp. disparity in ( bpy ), identical analogous disparities ( aoy:akx ) ( akx ) has no corresp. disparity in ( bmx )</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>( \beta )</td>
<td>( \gamma )</td>
<td>( \alpha \rightarrow \beta )</td>
<td>( \beta \rightarrow \delta )</td>
<td>( \beta \rightarrow \delta )</td>
<td>( \delta \rightarrow \delta )</td>
<td>( \text{non-identical} ) corresp. disparity ( aoy:bpy ) ( \text{non-identical} ) analogous disparity ( aoy:akx ) ( akx ) has no corresp. disparity in ( bmx )</td>
</tr>
<tr>
<td>( \alpha )</td>
<td>( \beta )</td>
<td>( \gamma )</td>
<td>( \ast )</td>
<td>( \alpha \rightarrow \gamma )</td>
<td>( \beta \rightarrow \gamma )</td>
<td>( \ast )</td>
<td>( \ast )</td>
</tr>
</tbody>
</table>

In the last row of (3.65), which is shaded, both \( akx \) and \( bmx \) have disparities on feature F for the corresponding segments, but the disparities are not identical and therefore cannot be corresponding disparities, due to different values for
feature $F$ in $s_a$ and $s_b$: $akx$ has disparity $s_a(\alpha):s_y(\gamma)$, while $bmx$ has disparity $s_b(\beta):s_x(\gamma)$. In that row, the value $F(s_y)$ is immaterial.

In the first row of (3.65), there are no disparities: all four relevant segments have the same value for feature $F$. In rows 2–5, disparities have identical corresponding disparities, between $akx$ and $bmx$, between $aoy$ and $bpy$, or both. In row 4, the disparity $s_a(\alpha):s_y(\gamma)$ of $aoy$ has an identical corresponding disparity in $bpy$, and also an identical analogous disparity in $akx$; the analogous disparity in $akx$ furthermore has an identical corresponding disparity in $bmx$. In row 5, the disparity $s_a(\alpha):s_y(\delta)$ of $aoy$ has an identical corresponding disparity in $bpy$, and also a non-identical analogous disparity $s_a(\alpha):s_x(\gamma)$ in $akx$; the disparity $s_a(\alpha):s_x(\gamma)$ in $akx$ furthermore has an identical corresponding disparity in $bmx$.

In row 6, $aoy$ and $bmx$ have no disparity on the relevant segments. $akx$ has an identity disparity, which thus has no corresponding disparity in $bmx$ and no analogous disparity in $aoy$. $bpy$ also has a feature disparity, with no corresponding disparity in $aoy$.

In row 7, $aoy$ has an identity disparity with no corresponding disparity in $bpy$. That disparity has an identical analogous disparity in $akx$; the analogous disparity in $akx$ further has no corresponding disparity in $bmx$.

In row 8, the identity disparity $s_a(\alpha):s_y(\delta)$ of $aoy$ has a non-identical corresponding disparity $s_b(\beta):s_y(\delta)$ in $bpy$; the disparities have different input segment values for $F$ (the output segment is shared). That disparity of $aoy$ also has a non-identical analogous disparity $s_a(\alpha):s_x(\beta)$ in $akx$; the disparities have different output segment values for $F$ (the input segment is shared). The disparity $s_a(\alpha):s_x(\beta)$ in $akx$ has no corresponding disparity in $bmx$.

From (3.58), (3.64), and (3.65), we can infer that the only identity disparities of $aoy$ that have no corresponding disparities in $bpy$ are those described in row 7 of (3.65). It follows that all identity disparities of $aoy$ with no corresponding disparities in $bpy$ have identical analogous disparities in $akx$ with no corresponding disparity in $bmx$.

(3.66) Each identity disparity $s_a(\alpha):s_y(\beta)$ in $aoy$ without a corresponding disparity in $bpy$ has an identical analogous disparity in $akx$ with no corresponding disparity in $bmx$.

From (3.58), (3.64), and (3.65), we can infer that the only identity disparities of $aoy$ that have non-identical corresponding disparities in $bpy$ are those described in row 8 of (3.65). Each such identity disparity $s_a(\alpha):s_y(\delta)$ of $aoy$
has a non-identical corresponding disparity $s_b(\beta):s_i(\delta)$ in $bpy$, and also a non-identical analogous disparity $s_a(\alpha):s_i(\beta)$ in $akx$ with no corresponding disparity in $bmx$.

\begin{equation}
(3.67) \quad \text{Each identity disparity } s_a(\alpha):s_i(\delta) \text{ in } aoy \text{ with a non-identical corresponding disparity } s_b(\beta):s_i(\delta) \text{ in } bpy \text{ has a non-identical analogous disparity } s_a(\alpha):s_i(\beta) \text{ in } akx \text{ with no corresponding disparity in } bmx.
\end{equation}

3.5.5 Comments/discussion
In a sense, the insertion disparities of $akx$ with no corresponding/analogous disparities “go with” the deletion disparities of $bpy$ with no corresponding disparities. Both involve input segment $s_b$ with output correspondent $s_x$ in $bmx$, where $s_b$ has no input–input correspondent, and $s_b$ has no output correspondent in $bpy$.

3.6 Output-driven preserving constraints: the proofs
3.6.1 Outline of the proof structure
Each proof has the same basic structure. It is an exercise in reasoning with inequalities. The goal in each case is to derive a relationship between the number of violations assessed to candidate $aoy$ and the number of violations assessed to candidate $akx$. The premise in each case is a relationship between the number of violations assessed to candidate $bpy$ and the number of violations assessed to candidate $bmx$. The two cases come from the two conditions for a constraint to be ODP, given in Section 3.3.2 as (3.39) and (3.40), and repeated below.

\begin{align}
(3.39) & \quad C(bpy) < C(bmx) \text{ entails } C(aoy) < C(akx) \\
(3.40) & \quad C(bpy) = C(bmx) \text{ entails } C(aoy) \leq C(akx)
\end{align}

A starting point for the proof structure is a property discussed in Section 3.2.3: every disparity in $aoy$ has either a corresponding disparity in $bpy$ or an analogous disparity in $akx$. The disparities in $aoy$ can be partitioned into those that have corresponding disparities in $bpy$, and those that do not. The disparities in $akx$ can also be partitioned into those that have corresponding disparities in $bmx$, and those that do not.

A divide-and-conquer strategy is then used, separately reasoning about disparities with correspondents and disparities without correspondents. The proof shows that the number of violations assessed to $aoy$ for disparities with $bpy$
correspondents is less than the number of violations assessed to $akx$ for disparities with $bmx$ correspondents, if $bpy$ has fewer violations than $bmx$. The proof also shows that the number of violations assessed to $aoy$ for disparities without $bpy$ correspondents is less than or equal to the number of violations assessed to $akx$ for disparities without $bmx$ correspondents. The partition exhausts the violations for $aoy$, so the total number of violations for $aoy$ must be less than the total number of violations for $akx$. Similar reasoning applies for the other case, with the premise that $bpy$ and $bmx$ have an equal number of violations.

The reasoning for violations assessed to disparities with correspondents relates the violations for $aoy$ to the violations for $akx$ via violations for $bpy$ and $bmx$.

- The violations for $aoy$ are related to the violations in $bpy$ via the correspondence relation between the disparities for $aoy$ and $bpy$.
- The violations for $bpy$ are related to the violations for $bmx$ by hypothesis: $bpy$ has fewer violations than $bmx$ for condition (3.39), and an equal number of violations as $bmx$ for condition (3.40).
- The violations for $bmx$ are related to the violations for $akx$ by the correspondent relations between their disparities, as dictated by the premise that $bmx$ has greater internal similarity than $akx$: every disparity in $bmx$ has an identical corresponding disparity in $akx$.

The reasoning for the violations assessed to disparities without correspondents relates the violations for $aoy$ to the violations for $akx$ directly, because every disparity in $aoy$ without a corresponding disparity in $bpy$ has an analogous disparity in $akx$.

### 3.6.2 MAX

$MAX$ (hereafter $MAX$) evaluates the input–output correspondence relation of a candidate and assesses a violation for every segment of the input that has no output correspondent (McCarthy and Prince 1999). In other words, $MAX$ assesses one violation for each deletion disparity in a candidate. $MAX(akx)$ denotes the number of violations of $MAX$ incurred by $akx$ and therefore is also the number of segments of the input that lack an output correspondent in the candidate.

#### 3.6.2.1 Partition of the deletion disparities

For each candidate, each deletion disparity of that candidate either does or does not have a corresponding deletion disparity in the other candidate with the same output ($akx$ with $bmx$, $aoy$ with $bpy$). Therefore, for each candidate,
the total set of $\text{Max}$ violations can be partitioned into those assessed to disparities with corresponding disparities and those assessed to disparities lacking corresponding disparities.

(3.68) $\text{Max}(akx) = \text{Max}(akx : corr) + \text{Max}(akx : no-corr)$

(3.69) $\text{Max}(bmx) = \text{Max}(bmx : corr) + \text{Max}(bmx : no-corr)$

(3.70) $\text{Max}(aoy) = \text{Max}(aoy : corr) + \text{Max}(aoy : no-corr)$

(3.71) $\text{Max}(bpy) = \text{Max}(bpy : corr) + \text{Max}(bpy : no-corr)$

3.6.2.2 Corresponding deletion disparities for $aoy$ and $bpy$

By (3.71),

$\text{Max}(bpy) = \text{Max}(bpy : corr) + \text{Max}(bpy : no-corr)$

Violation counts must be non-negative, so $0 \leq \text{Max}(bpy : no-corr)$. Therefore, the total number of violations of $\text{Max}$ assessed to $bpy$ must be at least the number of violations assessed to deletion disparities without corresponding disparities in $aoy$.

$\text{Max}(bpy : corr) \leq \text{Max}(bpy)$

By (3.42), only deletion disparities in $aoy$ that involve input segments with input–input correspondents can have corresponding disparities in $bpy$. By (3.48) and (3.49), every deletion disparity in $aoy$ that involves an input segment with an input–input correspondent has a corresponding disparity in $bpy$. Although some of the deletion disparities in $aoy$ may be non-identical to their correspondents in $bpy$, specifically the ones described in (3.49), $\text{Max}$ assesses a separate violation to each deletion disparity regardless of segment identity. The number of violations of $\text{Max}$ assessed to deletion disparities in $aoy$ with corresponding disparities in $bpy$ is equal to the number of violations of $\text{Max}$ assessed to deletion disparities in $bpy$ with corresponding disparities in $aoy$.

$\text{Max}(aoy : corr) = \text{Max}(bpy : corr)$

Substituting into the previous result, we reach the conclusion that

(3.72) $\text{Max}(aoy : corr) \leq \text{Max}(bpy)$

3.6.2.3 Corresponding deletion disparities for $akx$ and $bmx$

By hypothesis, $bmx$ has greater internal similarity than $akx$. Therefore, every disparity of $bmx$ has an identical corresponding disparity in $akx$. 

3.6.2.4 Non-corresponding deletion disparities for \(aoy\) and \(akx\)

From (3.48) and (3.49), we may conclude that deletion disparities of \(aoy\) lack corresponding disparities in \(bpy\) only when they involve input segments that lack input–input correspondents. From (3.43) we may conclude that all such deletion disparities in \(aoy\) have identical analogous disparities in \(akx\) that lack corresponding disparities in \(bmx\).

Although some of the deletion disparities in \(aoy\) may be non-identical to their analogs in \(akx\), \(\text{Max}\) assesses a separate violation to each deletion disparity, so the number of violations of \(\text{Max}\) assessed to deletion disparities in \(aoy\) lacking corresponding disparities in \(bpy\) is at most the number of violations of \(\text{Max}\) assessed to deletion disparities in \(akx\) lacking corresponding disparities in \(bmx\).

(3.74) \(\text{Max}(aoy : \text{no-corr}) \leq \text{Max}(akx : \text{no-corr})\)

3.6.2.5 \(\text{Max}(bpy) < \text{Max}(bmx)\) entails \(\text{Max}(aoy) < \text{Max}(akx)\)

By (3.70),

\[
\text{Max}(aoy) = \text{Max}(aoy : \text{corr}) + \text{Max}(aoy : \text{no-corr})
\]

By (3.72), \(\text{Max}(aoy : \text{corr}) \leq \text{Max}(bpy)\).

\[
\text{Max}(aoy) \leq \text{Max}(bpy) + \text{Max}(aoy : \text{no-corr})
\]

By hypothesis, \(\text{Max}(bpy) < \text{Max}(bmx)\).

\[
\text{Max}(aoy) < \text{Max}(bmx) + \text{Max}(aoy : \text{no-corr})
\]

By (3.73), \(\text{Max}(akx : \text{corr}) = \text{Max}(bmx)\).

\[
\text{Max}(aoy) < \text{Max}(akx : \text{corr}) + \text{Max}(aoy : \text{no-corr})
\]

By (3.74), \(\text{Max}(aoy : \text{no-corr}) \leq \text{Max}(akx : \text{no-corr})\).

\[
\text{Max}(aoy) < \text{Max}(akx : \text{corr}) + \text{Max}(akx : \text{no-corr})
\]

By (3.68), \(\text{Max}(akx) = \text{Max}(akx : \text{corr}) + \text{Max}(akx : \text{no-corr})\).

\[
\text{Max}(aoy) < \text{Max}(akx)
\]

End of Proof
3.6 Output-driven preserving constraints: the proofs

3.6.2.6 \( \max(bpy) = \max(bmx) \) entails \( \max(aoy) \leq \max(akx) \)

By (3.70),
\[
\max(aoy) = \max(aoy: \text{corr}) + \max(aoy: \text{no-corr})
\]

By (3.72), \( \max(aoy: \text{corr}) \leq \max(bpy) \).

\[
\max(aoy) \leq \max(bpy) + \max(aoy: \text{no-corr})
\]

By hypothesis, \( \max(bpy) = \max(bmx) \).

\[
\max(aoy) \leq \max(bmx) + \max(aoy: \text{no-corr})
\]

By (3.73), \( \max(akx: \text{corr}) = \max(bmx) \).

\[
\max(aoy) \leq \max(akx: \text{corr}) + \max(aoy: \text{no-corr})
\]

By (3.74), \( \max(aoy: \text{no-corr}) \leq \max(akx: \text{no-corr}) \).

\[
\max(aoy) \leq \max(akx: \text{corr}) + \max(akx: \text{no-corr})
\]

By (3.68), \( \max(akx) = \max(akx: \text{corr}) + \max(akx: \text{no-corr}) \).

\[
\max(aoy) \leq \max(akx)
\]

End of Proof

3.6.3 \textit{Dep}

\textit{Dep}-IO (hereafter \textit{Dep}) evaluates the input–output correspondence relation of a candidate and assesses a violation for every element of the output that has no input correspondent (McCarthy and Prince 1999). In other words, \textit{Dep} assesses one violation for each insertion disparity in a candidate. \textit{Dep}(akx) denotes the number of violations of \textit{Dep} incurred by \textit{akx}, and therefore is also the number of segments of the output that lack an input correspondent in the candidate.

3.6.3.1 Partition of the insertion disparities

For each candidate, each insertion disparity of that candidate either does or does not have a corresponding insertion disparity in the other candidate with the same output (\textit{akx} with \textit{bmx}, \textit{aoy} with \textit{bpy}). Therefore, for each candidate, the total set of \textit{Dep} violations can be partitioned into those assessed to disparities with corresponding disparities and those assessed to disparities lacking corresponding disparities.

\[
\text{Dep}(akx) = \text{Dep}(akx: \text{corr}) + \text{Dep}(akx: \text{no-corr})
\]

\[
\text{Dep}(bmx) = \text{Dep}(bmx: \text{corr}) + \text{Dep}(bmx: \text{no-corr})
\]
106  Output-driven maps in Optimality Theory

(3.77) \[ \text{Dep}(aoy) = \text{Dep}(aoy : corr) + \text{Dep}(aoy : no-corr) \]

(3.78) \[ \text{Dep}(bpy) = \text{Dep}(bpy : corr) + \text{Dep}(bpy : no-corr) \]

3.6.3.2 Corresponding insertion disparities for aoy and bpy

By (3.78),

\[ \text{Dep}(bpy) = \text{Dep}(bpy : corr) + \text{Dep}(bpy : no-corr) \]

Violation counts must be non-negative, so \(0 \leq \text{Dep}(bpy : no-corr)\). Therefore, the total number of violations of \(\text{Dep}\) assessed to \(bpy\) must be at least the number of violations assessed to insertion disparities without corresponding disparities in \(aoy\).

\[ \text{Dep}(bpy : corr) \leq \text{Dep}(bpy) \]

By (3.53), every insertion disparity in \(aoy\) with a corresponding disparity in \(bpy\) is identical to its corresponding disparity. Every such disparity is a single separate violation of \(\text{Dep}\).

\[ \text{Dep}(aoy : corr) = \text{Dep}(bpy : corr) \]

Substituting into the previous result, we reach the conclusion that

(3.79) \[ \text{Dep}(aoy : corr) \leq \text{Dep}(bpy) \]

3.6.3.3 Corresponding insertion disparities for akx and bmx

By (3.52), every insertion disparity of \(bmx\) has an identical corresponding disparity in \(akx\). Every such insertion disparity is a single separate violation of \(\text{Dep}\).

(3.80) \[ \text{Dep}(akx : corr) = \text{Dep}(bmx) \]

3.6.3.4 Non-corresponding insertion disparities for aoy and akx

From (3.55) and (3.56), we may conclude that every insertion disparity in \(aoy\) without a corresponding disparity in \(bpy\) has an analogous disparity in \(akx\) that has no corresponding disparity in \(bmx\). Although some of the insertion disparities may be non-identical to their analogs, \(\text{Dep}\) assesses a violation to every insertion disparity, so the number of violations of \(\text{Dep}\) assessed to insertion disparities in \(aoy\) lacking corresponding disparities in \(bpy\) is at most the number of violations of \(\text{Dep}\) assessed to insertion disparities in \(akx\) lacking corresponding disparities in \(bmx\).

(3.81) \[ \text{Dep}(aoy : no-corr) \leq \text{Dep}(akx : no-corr) \]
3.6.3.5 Proof: $\text{Dep}(bpy) < \text{Dep}(bmx)$ entails $\text{Dep}(aoy) < \text{Dep}(akx)$

By (3.77),

$\text{Dep}(aoy) = \text{Dep}(aoy : \text{corr}) + \text{Dep}(aoy : \text{no-corr})$

By (3.79), $\text{Dep}(aoy : \text{corr}) \leq \text{Dep}(bpy)$.

$\text{Dep}(aoy) \leq \text{Dep}(bpy) + \text{Dep}(aoy : \text{no-corr})$

By hypothesis, $\text{Dep}(bpy) < \text{Dep}(bmx)$.

$\text{Dep}(aoy) < \text{Dep}(bmx) + \text{Dep}(aoy : \text{no-corr})$

By (3.80), $\text{Dep}(akx : \text{corr}) = \text{Dep}(bmx)$.

$\text{Dep}(aoy) < \text{Dep}(akx : \text{corr}) + \text{Dep}(aoy : \text{no-corr})$

By (3.81), $\text{Dep}(aoy : \text{no-corr}) \leq \text{Dep}(akx : \text{no-corr})$.

$\text{Dep}(aoy) < \text{Dep}(akx : \text{corr}) + \text{Dep}(akx : \text{no-corr})$

By (3.75), $\text{Dep}(akx) = \text{Dep}(akx : \text{corr}) + \text{Dep}(akx : \text{no-corr})$.

$\text{Dep}(aoy) < \text{Dep}(akx)$

End of Proof

3.6.3.6 Proof: $\text{Dep}(bpy) = \text{Dep}(bmx)$ entails $\text{Dep}(aoy) \leq \text{Dep}(akx)$

By (3.77),

$\text{Dep}(aoy) = \text{Dep}(aoy : \text{corr}) + \text{Dep}(aoy : \text{no-corr})$

By (3.79), $\text{Dep}(aoy : \text{corr}) \leq \text{Dep}(bpy)$.

$\text{Dep}(aoy) \leq \text{Dep}(bpy) + \text{Dep}(aoy : \text{no-corr})$

By hypothesis, $\text{Dep}(bpy) = \text{Dep}(bmx)$.

$\text{Dep}(aoy) \leq \text{Dep}(bmx) + \text{Dep}(aoy : \text{no-corr})$

By (3.80), $\text{Dep}(akx : \text{corr}) = \text{Dep}(bmx)$.

$\text{Dep}(aoy) \leq \text{Dep}(akx : \text{corr}) + \text{Dep}(aoy : \text{no-corr})$

By (3.81), $\text{Dep}(aoy : \text{no-corr}) \leq \text{Dep}(akx : \text{no-corr})$.

$\text{Dep}(aoy) \leq \text{Dep}(akx : \text{corr}) + \text{Dep}(akx : \text{no-corr})$
By (3.75), $\text{Dep}(akx) = \text{Dep}(akx : \text{corr}) + \text{Dep}(akx : \text{no-corr})$.

$$\text{Dep}(aoy) \leq \text{Dep}(akx)$$

End of Proof

3.6.4 $\text{Ident}[F_{in} \in V]$

$\text{Ident-IO}[F]$ (hereafter $\text{Ident}[F]$) evaluates the input–output correspondence relation of a candidate and assesses a violation for every corresponding pair of segments that do not have identical values of the feature $F$ (McCarthy and Prince 1999).

$\text{Ident-IO}[F_{in} \in V]$ (hereafter $\text{Ident}[F_{in} \in V]$) evaluates the input–output correspondence relation of a candidate and assesses a violation for every corresponding pair of segments for which the input segment has a value $v$ for feature $F$ that is a member of the set $V$, but the output correspondent does not have the value $v$ for feature $F$. This constraint is like $\text{Ident}[F]$ but is restricted to evaluate only corresponding segment pairs where the input segment has a specific value of the feature being evaluated for identity.

As discussed in Section 3.4.4, Pater 1999 first proposed constraints that were like $\text{Ident}$, but restricted to correspondences in which the input segment had a particular value, and de Lacy 2002 proposed $\text{Ident}$ constraints restricted to correspondences in which the input segment had a value belonging to a subset of the possible values for a feature, in the context of markedness scales and scale category conflation. Here, I analyze the general class of $\text{Ident}$ constraints with input value restrictions. The constraints proposed by Pater are equivalent to having the restriction set of feature values $V$ contain only a single value. If $V$ contains all of the possible values for feature $F$, then $\text{Ident}[F_{in} \in V]$ becomes equivalent to $\text{Ident}[F]$. Thus, the result proven here for $\text{Ident}[F_{in} \in V]$ is a generalization that includes $\text{Ident}[F]$.

$F(s_a)$ represents the value of feature $F$ in the segment $s_a$.

3.6.4.1 Partition of the identity disparities

For each candidate, each identity disparity of that candidate either does or does not have a corresponding identity disparity in the other candidate with the same output ($akx$ with $bmx$, $aoy$ with $bpy$). Therefore, for each candidate, the total set of $\text{Ident}[F_{in} \in V]$ violations can be partitioned into those assessed to disparities with corresponding disparities and those assessed to disparities lacking corresponding disparities.

A further distinction is made here with respect to candidates $aoy$ and $bpy$. Because some of the disparities of $aoy$ with corresponding disparities in $bpy$ are
possibly non-identical to their corresponding disparities, we further partition
the violations assessed to \( aoy \) into those assessed to disparities with identical
Corresponding disparities in \( bpy \), and those assessed to disparities with non-
identical corresponding disparities in \( bpy \). The disparities of \( bpy \) are similarly
partitioned.

(3.82) \[ \text{Ident} \left[ F_{\text{in}} \in V \right] (akx) = \text{Ident} \left[ F_{\text{in}} \in V \right] (akx : \text{corr}) + \text{Ident} \left[ F_{\text{in}} \in V \right] (akx : \text{no-corr}) \]

(3.83) \[ \text{Ident} \left[ F_{\text{in}} \in V \right] (bmx) = \text{Ident} \left[ F_{\text{in}} \in V \right] (bmx : \text{corr}) + \text{Ident} \left[ F_{\text{in}} \in V \right] (bmx : \text{no-corr}) \]

(3.84) \[ \text{Ident} \left[ F_{\text{in}} \in V \right] (aoy) = \text{Ident} \left[ F_{\text{in}} \in V \right] (aoy : \text{id corr}) + \text{Ident} \left[ F_{\text{in}} \in V \right] (aoy : \text{non-id corr}) + \text{Ident} \left[ F_{\text{in}} \in V \right] (aoy : \text{no-corr}) \]

(3.85) \[ \text{Ident} \left[ F_{\text{in}} \in V \right] (bpy) = \text{Ident} \left[ F_{\text{in}} \in V \right] (bpy : \text{id corr}) + \text{Ident} \left[ F_{\text{in}} \in V \right] (bpy : \text{non-id corr}) + \text{Ident} \left[ F_{\text{in}} \in V \right] (bpy : \text{no-corr}) \]

3.6.4.2 Identical corresponding identity disparities for \( aoy \) and \( bpy \)

By (3.85),

\[ \text{Ident} \left[ F_{\text{in}} \in V \right] (bpy) = \text{Ident} \left[ F_{\text{in}} \in V \right] (bpy : \text{id corr}) + \text{Ident} \left[ F_{\text{in}} \in V \right] (bpy : \text{non-id corr}) + \text{Ident} \left[ F_{\text{in}} \in V \right] (bpy : \text{no-corr}) \]

Violation counts must be non-negative, so

\[ 0 \leq \text{Ident} \left[ F_{\text{in}} \in V \right] (bpy : \text{no-corr}) \]
\[ 0 \leq \text{Ident} \left[ F_{\text{in}} \in V \right] (bpy : \text{non-id corr}) \]

Therefore, the total number of violations assessed to \( bpy \) must be at least the
number of violations assessed to identity disparities with identical correspond-
ing disparities in \( aoy \).

\[ \text{Ident} \left[ F_{\text{in}} \in V \right] (bpy : \text{id corr}) \leq \text{Ident} \left[ F_{\text{in}} \in V \right] (bpy) \]

Each pair of identical corresponding identity disparities between \( aoy \) and \( bpy \)
has the same feature values for their input segments: that value is either in \( V \),
in which case both disparities violate the constraint, or the value is not in \( V \), in
which case neither disparity violates the constraint. The number of violations
assessed to disparities with identical correspondents is thus the same for \( aoy \)
and \( bpy \).

\[ \text{Ident} \left[ F_{\text{in}} \in V \right] (aoy : \text{id corr}) = \text{Ident} \left[ F_{\text{in}} \in V \right] (bpy : \text{id corr}) \]

Substituting into the previous result yields
3.6.4.3 Corresponding identity disparities for \textit{akx} and \textit{bmx}

By hypothesis, \textit{bmx} has greater internal similarity than \textit{akx}. Therefore, every disparity of \textit{bmx} has an identical corresponding disparity in \textit{akx}.

(3.87) \textbf{Ident}[F_{in} \in V](\textit{akx} : \textit{corr}) = \textbf{Ident}[F_{in} \in V](\textit{bmx})

3.6.4.4 Non-Corresponding and non-identical corresponding identity disparities for \textit{aoy} and \textit{akx}

By (3.66), each identity disparity \(s_{a}(\alpha):s_{\beta}(\beta)\) in \textit{aoy} without a corresponding disparity in \textit{bpy} has an identical analogous disparity in \textit{akx} with no corresponding disparity in \textit{bmx}. Because the analogous disparities share the same input segment, \(s_{a}\), either \(F(s_{a}) \in V\), in which case both disparities constitute violations, or \(F(s_{a}) \notin V\), in which case neither disparity constitutes a violation.

By (3.67), each identity disparity \(s_{a}(\alpha):s_{\beta}(\beta)\) in \textit{aoy} with a non-identical corresponding disparity \(s_{a}(\alpha):s_{\beta}(\beta)\) in \textit{bpy} has a non-identical analogous disparity \(s_{a}(\alpha):s_{\beta}(\beta)\) in \textit{akx} with no corresponding disparity in \textit{bmx}. The analogous identity disparities for \(F\) involving \(s_{a}\) in \textit{aoy} and \textit{akx} might not have the same values for \(F\) in their respective output segments. However, because these disparities share the same input segment, \(s_{a}\), either \(F(s_{a}) \in V\), in which case both disparities constitute violations, or \(F(s_{a}) \notin V\), in which case neither disparity constitutes a violation.

Every identity disparity in \textit{aoy} lacking an identical corresponding disparity in \textit{bpy}, be it a disparity with no corresponding disparity in \textit{bpy} or a disparity with a non-identical corresponding disparity in \textit{bpy}, has an analogous disparity in \textit{akx} that lacks a corresponding disparity in \textit{bmx}. Each pair of such analogous disparities share the same input segment, and thus either both or neither violate \textbf{Ident}[F_{in} \in V]. Each such analog disparity in \textit{akx} has no corresponding disparity in \textit{bmx}.

(3.88) \textbf{Ident}[F_{in} \in V](\textit{aoy} : \textit{no-corr}) + \textbf{Ident}[F_{in} \in V](\textit{aoy} : \textit{non-id corr}) \leq \textbf{Ident}[F_{in} \in V](\textit{akx} : \textit{no-corr})

3.6.4.5 Proof: \(\textbf{Ident}[F_{in} \in V](\textit{bpy}) < \textbf{Ident}[F_{in} \in V](\textit{bmx})\) entails \(\textbf{Ident}[F_{in} \in V](\textit{aoy}) < \textbf{Ident}[F_{in} \in V](\textit{akx})\)

By (3.84),

\[
\textbf{Ident}[F_{in} \in V](\textit{aoy}) = \textbf{Ident}[F_{in} \in V](\textit{aoy} : \textit{id corr}) + \\
\textbf{Ident}[F_{in} \in V](\textit{aoy} : \textit{non-id corr}) + \textbf{Ident}[F_{in} \in V](\textit{aoy} : \textit{no-corr})
\]
By (3.86), $\text{Ident}[F_{in} \in V](aoy : \text{id corr}) \leq \text{Ident}[F_{in} \in V](bpy)$.

$$\text{Ident}[F_{in} \in V](aoy) \leq \text{Ident}[F_{in} \in V](bpy) + \text{Ident}[F_{in} \in V](aoy : \text{non-id corr}) + \text{Ident}[F_{in} \in V](aoy : \text{no-corr})$$

By hypothesis, $\text{Ident}[F_{in} \in V](bpy) < \text{Ident}[F_{in} \in V](bmx)$.

$$\text{Ident}[F_{in} \in V](aoy) < \text{Ident}[F_{in} \in V](bmx) + \text{Ident}[F_{in} \in V](aoy : \text{non-id corr}) + \text{Ident}[F_{in} \in V](aoy : \text{no-corr})$$

By (3.87), $\text{Ident}[F_{in} \in V](akx : \text{corr}) = \text{Ident}[F_{in} \in V](bmx)$.

$$\text{Ident}[F_{in} \in V](aoy) < \text{Ident}[F_{in} \in V](akx : \text{corr}) + \text{Ident}[F_{in} \in V](aoy : \text{non-id corr}) + \text{Ident}[F_{in} \in V](aoy : \text{no-corr})$$

By hypothesis, $\text{Ident}[F_{in} \in V](bpy) = \text{Ident}[F_{in} \in V](bmx)$.

$$\text{Ident}[F_{in} \in V](aoy) < \text{Ident}[F_{in} \in V](akx)$$

End of Proof

3.6.4.6 Proof: $\text{Ident}[F_{in} \in V](bpy) = \text{Ident}[F_{in} \in V](bmx)$ entails $\text{Ident}[F_{in} \in V](aoy) \leq \text{Ident}[F_{in} \in V](akx)$

By (3.84),

$$\text{Ident}[F_{in} \in V](aoy) = \text{Ident}[F_{in} \in V](aoy : \text{id corr}) + \text{Ident}[F_{in} \in V](aoy : \text{non-id corr}) + \text{Ident}[F_{in} \in V](aoy : \text{no-corr})$$

By hypothesis, $\text{Ident}[F_{in} \in V](bpy) = \text{Ident}[F_{in} \in V](bmx)$.
Output-driven maps in Optimality Theory

\[ \text{Ident}[F_{in} \in V](aoy) \leq \text{Ident}[F_{in} \in V](bmx) + \]
\[ \text{Ident}[F_{in} \in V](aoy : \text{non-id corr}) + \text{Ident}[F_{in} \in V](aoy : \text{no-corr}) \]

By (3.87), \( \text{Ident}[F_{in} \in V](akx : \text{corr}) = \text{Ident}[F_{in} \in V](bmx) \).

\[ \text{Ident}[F_{in} \in V](aoy) \leq \text{Ident}[F_{in} \in V](akx : \text{corr}) + \]
\[ \text{Ident}[F_{in} \in V](aoy : \text{non-id corr}) + \text{Ident}[F_{in} \in V](aoy : \text{no-corr}) \]

By (3.88), \( \text{Ident}[F_{in} \in V](aoy : \text{no-corr}) + \text{Ident}[F_{in} \in V](aoy : \text{non-id corr}) \leq \text{Ident}[F_{in} \in V](akx : \text{no-corr}) \).

\[ \text{Ident}[F_{in} \in V](aoy) \leq \text{Ident}[F_{in} \in V](akx : \text{corr}) + \]
\[ \text{Ident}[F_{in} \in V](akx : \text{no-corr}) \]

By (3.82), \( \text{Ident}[F_{in} \in V](akx) = \text{Ident}[F_{in} \in V](akx : \text{corr}) + \text{Ident}[F_{in} \in V](akx : \text{no-corr}) \).

\[ \text{Ident}[F_{in} \in V](aoy) \leq \text{Ident}[F_{in} \in V](akx) \]

End of Proof

3.6.5 \( \text{Ident}[F_{out} \in V] \)

\( \text{Ident-IO}[F] \) (hereafter \( \text{Ident}[F] \)) evaluates the input–output correspondence relation of a candidate and assesses a violation for every corresponding pair of segments that do not have identical values of the feature \( F \) (McCarthy and Prince 1999).

\( \text{Ident-IO}[F_{out} \in V] \) (hereafter \( \text{Ident}[F_{out} \in V] \)) evaluates the input–output correspondence relation of a candidate and assesses a violation for every corresponding pair of segments for which the output segment has a value \( v \) for feature \( F \) that is a member of the set \( V \), but the input correspondent does not have the value \( v \) for feature \( F \). This constraint is like \( \text{Ident}[F] \) but is restricted to evaluate only corresponding segment pairs where the output segment has a specific value of the feature being evaluated for identity.

As discussed in Section 3.4.4, Pater (1999) first proposed constraints that were like \( \text{Ident} \), but restricted to correspondences in which the output segment had a particular value. Here, I analyze the general class of \( \text{Ident} \) constraints with output value restrictions. The constraints proposed by Pater are equivalent to having the restriction set of feature values \( V \) contain only a single value. If \( V \) contains all of the possible values for feature \( F \), then \( \text{Ident}[F_{out} \in V] \) becomes
equivalent to \( \text{Ident}[F] \). Thus, the result proven here for \( \text{Ident}[F_{\text{out}} \in V] \) is a generalization that includes \( \text{Ident}[F] \).

\( F(s_a) \) represents the value of feature \( F \) in the segment \( s_a \).

### 3.6.5.1 Partition of the identity disparities

For each candidate, each identity disparity of that candidate either does or does not have a corresponding identity disparity in the other candidate with the same output (\( akx \) with \( bmx \), \( aoy \) with \( bpy \)). Therefore, for each candidate, the total set of \( \text{Ident}[F_{\text{out}} \in V] \) violations can be partitioned into those assessed to disparities with corresponding disparities and those assessed to disparities lacking corresponding disparities.

\[
\begin{align*}
\text{Ident}[F_{\text{out}} \in V](akx) &= \text{Ident}[F_{\text{out}} \in V](akx: \text{corr}) + \\
&\quad \text{Ident}[F_{\text{out}} \in V](akx: \text{no-corr}) \\
\text{Ident}[F_{\text{out}} \in V](bmx) &= \text{Ident}[F_{\text{out}} \in V](bmx: \text{corr}) + \\
&\quad \text{Ident}[F_{\text{out}} \in V](bmx: \text{no-corr}) \\
\text{Ident}[F_{\text{out}} \in V](aoy) &= \text{Ident}[F_{\text{out}} \in V](aoy: \text{corr}) + \\
&\quad \text{Ident}[F_{\text{out}} \in V](aoy: \text{no-corr}) \\
\text{Ident}[F_{\text{out}} \in V](bpy) &= \text{Ident}[F_{\text{out}} \in V](bpy: \text{corr}) + \\
&\quad \text{Ident}[F_{\text{out}} \in V](bpy: \text{no-corr})
\end{align*}
\]

### 3.6.5.2 Corresponding identity disparities for \( aoy \) and \( bpy \)

By (3.92),

\[
\begin{align*}
\text{Ident}[F_{\text{out}} \in V](bpy) &= \text{Ident}[F_{\text{out}} \in V](bpy: \text{corr}) + \\
&\quad \text{Ident}[F_{\text{out}} \in V](bpy: \text{no-corr})
\end{align*}
\]

Violation counts must be non-negative, so

\[ 0 \leq \text{Ident}[F_{\text{out}} \in V](bpy: \text{no-corr}) \]

Therefore, the total number of violations assessed to identity disparities in \( bpy \) must be at least the number of violations assessed to identity disparities with corresponding disparities in \( aoy \).

\[
\begin{align*}
\text{Ident}[F_{\text{out}} \in V](bpy: \text{corr}) &\leq \text{Ident}[F_{\text{out}} \in V](bpy)
\end{align*}
\]

Each identity disparity in \( aoy \) with a corresponding identity disparity in \( bpy \) has the same output feature value as its corresponding disparity. This is because, by the definition of disparity correspondence, the two disparities share the same output segment \( s_y \). That output feature value is either in \( V \), in which case both
disparities violate the constraint, or not in $V$, in which case neither disparity violates the constraint. The number of violations assessed to disparities with correspondents is thus the same for $aoy$ and $bpy$.

$$\text{Ident}[F_{out} \in V](aoy : \text{corr}) = \text{Ident}[F_{out} \in V](bpy : \text{corr})$$

Substituting into the previous result yields

$$\text{(3.93)} \quad \text{Ident}[F_{out} \in V](aoy : \text{corr}) \leq \text{Ident}[F_{out} \in V](bpy)$$

3.6.5.3 Corresponding identity disparities for $akx$ and $bmx$

By hypothesis, $bmx$ has greater internal similarity than $akx$. Therefore, every disparity of $bmx$ has an identical corresponding disparity in $akx$.

$$\text{(3.94)} \quad \text{Ident}[F_{out} \in V](akx : \text{corr}) = \text{Ident}[F_{out} \in V](bmx)$$

3.6.5.4 Non-corresponding identity disparities for $aoy$ and $akx$

By (3.66), each identity disparity $s_y(\alpha):s_x(\beta)$ in $aoy$ without a corresponding disparity in $bpy$ has an identical analogous disparity $s_y(\alpha):s_x(\beta)$ in $akx$ with no corresponding disparity in $bmx$. Because the analogous disparities are identical, the value of feature F must be the same in the two output correspondents, $s_y$ and $s_x$: $F(s_y) = F(s_x) = \beta$. Either $\beta \notin V$, in which case both disparities constitute violations, or $\beta \notin V$, in which case neither disparity constitutes a violation.

$$\text{(3.95)} \quad \text{Ident}[F_{out} \in V](aoy : \text{no-corr}) \leq \text{Ident}[F_{out} \in V](akx : \text{no-corr})$$

3.6.5.5 Proof: $\text{Ident}[F_{out} \in V](bpy) < \text{Ident}[F_{out} \in V](bmx)$ entails $\text{Ident}[F_{out} \in V](aoy) < \text{Ident}[F_{out} \in V](akx)$

By (3.91),

$$\text{Ident}[F_{out} \in V](aoy) = \text{Ident}[F_{out} \in V](aoy : \text{corr}) + \text{Ident}[F_{out} \in V](aoy : \text{no-corr})$$

By (3.93), $\text{Ident}[F_{out} \in V](aoy : \text{corr}) \leq \text{Ident}[F_{out} \in V](bpy)$.

$$\text{Ident}[F_{out} \in V](aoy) \leq \text{Ident}[F_{out} \in V](bpy) + \text{Ident}[F_{out} \in V](aoy : \text{no-corr})$$

By hypothesis, $\text{Ident}[F_{out} \in V](bpy) < \text{Ident}[F_{out} \in V](bmx)$.

$$\text{Ident}[F_{out} \in V](aoy) < \text{Ident}[F_{out} \in V](bmx) + \text{Ident}[F_{out} \in V](aoy : \text{no-corr})$$

By (3.94), $\text{Ident}[F_{out} \in V](akx : \text{corr}) = \text{Ident}[F_{out} \in V](bmx)$.
3.6 Output-driven preserving constraints: the proofs

\[ \text{\textsc{ident}}[F_{\text{out}} \in V](aoy) < \text{\textsc{ident}}[F_{\text{out}} \in V](akx : \text{corr}) + \text{\textsc{ident}}[F_{\text{out}} \in V](aoy : \text{no-corr}) \]

By (3.95), \text{\textsc{ident}}[F_{\text{out}} \in V](aoy : \text{no-corr}) \leq \text{\textsc{ident}}[F_{\text{out}} \in V](akx : \text{no-corr}).

\[ \text{\textsc{ident}}[F_{\text{out}} \in V](aoy) < \text{\textsc{ident}}[F_{\text{out}} \in V](akx : \text{corr}) + \text{\textsc{ident}}[F_{\text{out}} \in V](akx : \text{no-corr}) \]

By (3.89), \text{\textsc{ident}}[F_{\text{out}} \in V](akx) = \text{\textsc{ident}}[F_{\text{out}} \in V](akx : \text{corr}) + \text{\textsc{ident}}[F_{\text{out}} \in V](akx : \text{no-corr}).

\[ \text{\textsc{ident}}[F_{\text{out}} \in V](aoy) < \text{\textsc{ident}}[F_{\text{out}} \in V](akx) \]

End of Proof

3.6.5.6 Proof: \text{\textsc{ident}}[F_{\text{out}} \in V](bpy) = \text{\textsc{ident}}[F_{\text{out}} \in V](bmx) entails \text{\textsc{ident}}[F_{\text{out}} \in V](aoy) \leq \text{\textsc{ident}}[F_{\text{out}} \in V](akx)

By (3.91),

\[ \text{\textsc{ident}}[F_{\text{out}} \in V](aoy) = \text{\textsc{ident}}[F_{\text{out}} \in V](aoy : \text{corr}) + \text{\textsc{ident}}[F_{\text{out}} \in V](aoy : \text{no-corr}) \]

By (3.93), \text{\textsc{ident}}[F_{\text{out}} \in V](aoy : \text{corr}) \leq \text{\textsc{ident}}[F_{\text{out}} \in V](bpy).

\[ \text{\textsc{ident}}[F_{\text{out}} \in V](aoy) \leq \text{\textsc{ident}}[F_{\text{out}} \in V](bpy) + \text{\textsc{ident}}[F_{\text{out}} \in V](aoy : \text{no-corr}) \]

By hypothesis, \text{\textsc{ident}}[F_{\text{out}} \in V](bpy) = \text{\textsc{ident}}[F_{\text{out}} \in V](bmx).

\[ \text{\textsc{ident}}[F_{\text{out}} \in V](aoy) \leq \text{\textsc{ident}}[F_{\text{out}} \in V](bmx) + \text{\textsc{ident}}[F_{\text{out}} \in V](aoy : \text{no-corr}) \]

By (3.94), \text{\textsc{ident}}[F_{\text{out}} \in V](akx : \text{corr}) = \text{\textsc{ident}}[F_{\text{out}} \in V](bmx).

\[ \text{\textsc{ident}}[F_{\text{out}} \in V](aoy) \leq \text{\textsc{ident}}[F_{\text{out}} \in V](akx : \text{corr}) + \text{\textsc{ident}}[F_{\text{out}} \in V](aoy : \text{no-corr}) \]

By (3.95), \text{\textsc{ident}}[F_{\text{out}} \in V](aoy : \text{no-corr}) \leq \text{\textsc{ident}}[F_{\text{out}} \in V](akx : \text{no-corr}).

\[ \text{\textsc{ident}}[F_{\text{out}} \in V](aoy) \leq \text{\textsc{ident}}[F_{\text{out}} \in V](akx : \text{corr}) + \text{\textsc{ident}}[F_{\text{out}} \in V](akx : \text{no-corr}) \]

By (3.89), \text{\textsc{ident}}[F_{\text{out}} \in V](akx) = \text{\textsc{ident}}[F_{\text{out}} \in V](akx : \text{corr}) + \text{\textsc{ident}}[F_{\text{out}} \in V](akx : \text{no-corr}).

\[ \text{\textsc{ident}}[F_{\text{out}} \in V](aoy) \leq \text{\textsc{ident}}[F_{\text{out}} \in V](akx) \]

End of Proof
3.7 The map

A close understanding of the relationship between output-driven maps and Optimality Theory is achieved by connecting entailment relations between the grammatical status of candidates (a core element of output-driven maps) with entailment relations between ranking arguments (a core element of Optimality Theory). In particular, the conditions for an OT constraint to be output-driven preserving can be derived from the definition of output-driven maps.

All markedness constraints are output-driven preserving and thus cannot be primarily responsible for the generation of a map that is not output-driven. This makes intuitive sense. Markedness constraints are conditions on the output, so it is to be expected that their effects will be “motivated by conditions on the output.” This also reveals formally that markedness constraints are the wrong place to look when trying to predict the presence/absence of non-output-driven phenomena in a linguistic system. Relating to the discussion of opacity and Optimality Theory in Section 1.2.1, focusing on the status of markedness constraints as generalizations that are or are not true of the output simply doesn’t get you very far in understanding the output-drivenness of the map generated by a grammar.

With respect to the output-drivenness of maps generated by OT grammars, the heart of the matter lies in the constraints making reference to structures apart from the output, primarily those making reference to the input. Several foundational faithfulness constraints were proven in this chapter to be output-driven preserving: Max, Dep, and Ident[F], each in their most basic forms. Numerous constraints that are not output-driven preserving will be analyzed in Chapter 4.

More generally, Chapter 4 examines in detail those constraint behaviors capable of causing a constraint to fail to be output-driven preserving. Under the assumption that Gen is correspondence uniform, this provides a rigorous understanding of the constraint behaviors capable of causing an OT grammar to generate a map which is non-output-driven. As a consequence, the property output-driven preserving provides a unified understanding of a variety of constraints that have been proposed in the literature for the analysis of phenomena that aren’t surface oriented.
4 Analysis of constraint behavior

A constraint can only cause a non-output-driven map (non-ODM) if the constraint is not output-driven preserving (non-ODP). This chapter examines the relationships between non-ODP constraint behavior and non-ODMs.

Sections 4.1 through 4.3 derive the three possible non-ODP constraint behaviors, and for each behavior illustrate how it can cause a map generated by an OT grammar to be non-ODM. Sections 4.4 through 4.8 examine a variety of kinds of non-ODP constraints. Many of the specific constraints examined here were previously proposed in the literature, and some of them were proposed to account for phenomena described in terms of process opacity. Examining these constraints in light of the theory of output-driven maps provides deeper insight and reveals strong formal commonalities among constraints that appear superficially to be quite different.

Sections 4.9 and 4.10 examine a couple of prior proposals for relating some properties of maps to OT grammars. The theory of output-driven maps makes it possible to illuminate and extend the strongest of these results and helps reveal limitations in the other. The discussion also illustrates the central importance of the concept of relative similarity. Section 4.11 revisits the issue of the relationship between specific kinds of non-ODP behaviors and non-ODM patterns, drawing upon the material in Sections 4.4 through 4.10. Section 4.12 briefly discusses possible relationships between the theory of output-driven maps and grammars defined in terms of the composition of maps (such as in Kiparsky’s stratal OT).

4.1 Non-ODP constraints and non-output-driven maps

Recall that for a constraint to be output-driven preserving (ODP), it must have the properties given in (3.39) and (3.40), repeated here:

(3.39) \( C(bpy) < C(bmx) \) entails \( C(aoy) < C(akx) \)

(3.40) \( C(bpy) = C(bmx) \) entails \( C(aoy) \leq C(akx) \)
Any constraint which is not output-driven preserving (non-ODP) must have some instance in which it violates one of these properties. A constraint which has such an instance can be said to exhibit non-ODP behavior in that instance. Based upon the properties in (3.39) and (3.40), there are three kinds of non-ODP constraint behavior.

\[(4.1) \quad [C(bpy) = C(bmx)] \text{ and } [C(aoy) > C(akx)] \quad \text{violates (3.40)}\]

\[(4.2) \quad [C(bpy) < C(bmx)] \text{ and } [C(aoy) > C(akx)] \quad \text{violates (3.39)}\]

\[(4.3) \quad [C(bpy) < C(bmx)] \text{ and } [C(aoy) = C(akx)] \quad \text{violates (3.39)}\]

Constraints that are non-ODP have the potential to cause non-output-driven maps. The mere presence of a non-ODP constraint in a constraint hierarchy does not ensure that the hierarchy defines a non-output-driven map, of course. For instance, if the constraint is dominated by other constraints such that it is never active on any candidate competition, it cannot cause non-output drivenness for that ranking. It is also possible that a non-ODP constraint is active in a particular ranking, but interacts with other constraints such that output drivenness holds.

The three kinds of non-ODP behavior can be distinguished in terms of which key pairs of candidates they distinguish. The non-ODP behavior described in (4.1) distinguishes the competitors for \(in_a\), but not \(in_b\). Thus, (4.1) involves distinction only at lesser similarity: it distinguishes the candidates with the input having lesser similarity to \(out_x\) (\(aoy\) and \(akx\)), but not the candidates with the input having greater similarity to \(out_x\) (\(bpy\) and \(bmx\)). The non-ODP behavior described in (4.3) distinguishes the competitors for \(in_b\), but not \(in_a\), and is labeled distinction only at greater similarity. The non-ODP behavior described in (4.2) distinguishes the competitors for both \(in_a\) and \(in_b\), but in opposite ways relative to the output forms (preferring \(out_x\) for \(in_a\), but \(out_y\) for \(in_b\)), and is labeled distinction conflict.

All instances of non-output drivenness have the following general format:

- \(akx\) is optimal, and thus at least as harmonic as \(aoy\).
- \(bmx\) has greater internal similarity than \(akx\).
- \(bmx\) is less harmonic than \(bpy\), and thus not optimal.

If a constraint exhibits distinction only at lesser similarity, and it is the highest-ranked constraint with a preference in either of the two competitions, then it ensures that \(akx > aoy\) but does not distinguish between \(bmx\) and \(bpy\). If the lower-ranked constraints determine that \(bpy > bmx\), then the map is not output-driven (given the premise that \(akx\) is optimal).
If a constraint exhibits distinction only at greater similarity, and it is the highest-ranked constraint with a preference in either of the two competitions, then it ensures that $bpy > bmx$ but does not distinguish between $akx$ and $aoy$. Given the premise that $akx$ is optimal, the map is not output-driven. The optimality of $akx$ entails that if any lower-ranked constraints have a preference between $akx$ and $aoy$, the highest-ranked one must prefer $akx$.

If a constraint exhibits distinction conflict, and it is the highest-ranked constraint with a preference in either of the two competitions, then the constraint single-handedly ensures that both $akx > aoy$ and $bpy > bmx$, and thus that the map is not output-driven (given the premise that $akx$ is optimal). It does not require specific interaction from the lower-ranked constraints of the sort that the first two types of constraints do (in order to cause a non-output-driven map).

Observe that the only kind of distinction between $bmx$ and $bpy$ involved in non-ODP behavior is $C(bpy) < C(bmx)$. If it were the case that $C(bmx) < C(bpy)$, then the constraint could not play a role in causing $bmx$ to lose to $bpy$. Thus, distinction only at greater similarity presumes that the constraint doesn’t just distinguish the competitors for $in_b$, but does so to the effect that $C(bpy) < C(bmx)$.

Further observe that the only kind of distinction between $akx$ and $aoy$ involved in non-ODP behavior is $C(akx) < C(aoy)$. If it were the case that $C(aoy) < C(akx)$, then the constraint could not play a role in causing $akx$ to beat $aoy$; it would need to be crucially dominated by some other constraint that preferred $akx$ to $aoy$. Thus, distinction only at lesser similarity presumes that the constraint doesn’t just distinguish the competitors for $in_a$, but does so to the effect that $C(akx) < C(aoy)$.

4.2 Illustrating the three ways of non-ODP

4.2.1 Distinction only at lesser similarity

Kirchner (1995) analyzed Etxarri Basque vowel raising in hiatus as a chain shift: $e \rightarrow i \rightarrow i^\forall$, with the $+\text{low}$ vowel $/a/$ exempt from the shift ($a \rightarrow a$). Representationally, there are three vowel features involved: low, hi, and raised. In his analysis, the vowel $[i^\forall]$ is both $+\text{hi}$ and $+\text{raised}$ and is higher than the vowel $[i]$ which is $+\text{hi}$ and $–\text{raised}$. In the analysis, the sensitivity of raising to the hiatus context is contained in the markedness constraint which prefers the candidates with higher vowels, $\text{HiatusRaising}$. This constraint can be viewed as separately violated by occurrences of any of the following vowel feature values: $+\text{low}$, $–\text{hi}$, and $–\text{raised}$. I will focus on the vowel behavior and leave the conditioning phonological context implicit.
Kirchner originally constructed his analysis using Parse-feature constraints. However, they can be translated into correspondence-faithfulness terms; the translation used here comes from Kager (1999: 392–397). The key constraint for realizing the chain shift, [Ident[hi] & Ident[raised]]\textsubscript{V}, is non-ODP.

(4.4) Correspondence-based statement of Kirchner’s constraints

<table>
<thead>
<tr>
<th>Constraint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ident[hi]</td>
<td>corresponding vowels must match in hi (Kirchner’s Parse\textsubscript{hi})</td>
</tr>
<tr>
<td>Ident[low]</td>
<td>corresponding vowels must match in low (Kirchner’s Parse\textsubscript{low})</td>
</tr>
<tr>
<td>Ident[raised]</td>
<td>corresponding vowels must match in raised (Kirchner’s Parse\textsubscript{raised})</td>
</tr>
<tr>
<td>[Ident[hi] &amp; Ident[raised]]\textsubscript{V}, abbreviated Ident[hi&amp;rai]</td>
<td>corresponding vowels must match in at least one of hi and raised</td>
</tr>
</tbody>
</table>

**HiatusRaising**

markedness constraint penalizing non-maximally high vowels by degree ("a" 3 viols, "e" 2 viols, "i" 1 viol, "i\textsuperscript{y}" 0 viol)

The constraint [Ident[hi] & Ident[raised]]\textsubscript{V} is an example of a constraint defined through the operation of local conjunction (Smolensky 1995). The local conjunction of two component constraints (here, Ident[hi] and Ident[raised]) is violated when both of the component constraints are violated within a defined domain (here, the domain is a vowel). Local conjunction was originally part of a larger proposal concerning the structure of Con involving the application of local conjunction to other constraints present in a grammar, but that larger proposal is not of concern here. For present purposes, local conjunction is one way (of several) to understand the behavior of complex constraints in terms of the behavior of other (simpler) component constraints, regardless of whether the component constraints are themselves present as independent constraints in the same grammar as the complex constraint.

Of interest is the fact that the component constraints of [Ident[hi] & Ident[raised]]\textsubscript{V} are both input-referring; each requires identity in the value of a feature between corresponding input and output vowels. Although the conjoined constraint refers to both features (hi and raised), it is satisfied even if only one of them is identical. This is the key to the non-output drivenness in this analysis. Because the constraint is only violated when there are disparities on both features, the constraint will distinguish a candidate with disparities in both features from a candidate with a disparity on only one of the features (lesser similarity), but will not distinguish a candidate with one disparity from a candidate with none (greater similarity). Competitions for the four inputs are shown in (4.5).
4.2 Illustrating the three ways of non-ODP

Conjoined faithfulness exhibits distinction only at lesser similarity

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>/u[a]</td>
<td>***</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/u[e]</td>
<td>*!</td>
<td>**</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/u[i]</td>
<td>*!</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/u[\textipa{\textcc{u}}]</td>
<td>*!</td>
<td>*</td>
<td>***</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/e[a]</td>
<td>*!</td>
<td>***</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/e[e]</td>
<td>***</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aox</td>
<td>/e[i]</td>
<td>*!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/o[i]</td>
<td></td>
<td>*!</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bmx</td>
<td>/i[i]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bpy</td>
<td>/i[\textipa{\textcc{i}}]</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/i[\textipa{\textcc{u}}]</td>
<td>*!</td>
<td>*</td>
<td>***</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>/i[e]</td>
<td>*!</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/i[\textipa{\textcc{i}}]</td>
<td>*!</td>
<td>*</td>
<td>**</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

The map is non-output-driven because candidate *akx* is grammatical, and *bmx* has greater internal similarity than *akx*, yet *bmx* is not grammatical.

The key to understanding the non-output-driven map is understanding why the candidate *aoy, /e[i\textipa{\textcc{i}}]*, isn’t grammatical in the first place. /e[i\textipa{\textcc{i}}] is precisely the candidate for input /e/ eliminated by IDENT[hi&rai] in favor of *akx, /e[i]*. That is, IDENT[hi&rai](aoy) > IDENT[hi&rai](akx). But when the input is changed to /i/, which has greater similarity to [i] than /e/ has, IDENT[hi&rai] goes silent, failing to distinguish between /i[i] and /i[\textipa{\textcc{i}}]. The key cells are shaded in the tableau in (4.5). The competition between /i[i] and /i[\textipa{\textcc{i}}] falls to the lower-ranked HiatusRaising, which decides against /i[i] (HiatusRaising is denied the opportunity to decide the comparison between /e[i] and /e[\textipa{\textcc{i}}] by IDENT[hi&rai]). In other words, IDENT[hi&rai](bpy) = IDENT[hi&rai](bmx) while IDENT[hi&rai](aoy) > IDENT[hi&rai](akx). Thus, IDENT[hi&rai] satisfies the condition in (4.1) (repeated below).

(4.1) \[ C(bpy) = C(bmx) \] and \[ C(aoy) > C(akx) \]

The behavior of IDENT[hi&rai] in this instance constitutes distinction only at lesser similarity: it distinguishes candidates for input /e/ with lesser similarity to
the output [i], but does not distinguish the corresponding candidates for input /i/
with greater similarity to the output [i].

The non-ODP behavior of IDENT[hi&rai] can be traced to its conjunc-
tion structure. IDENT[hi&rai] can be satisfied without enforcing identity to
all of the elements that the constraint evaluates on input–output correspon-
dence. Candidate /e/[i′] mismatches in both hi and raised, incurring a violation
of IDENT[hi&rai]. /e/[i] satisfies the constraint, even though one of the re-
ferenced features, hi, still doesn’t match. In a sense, the constraint evaluates
the disparity on the feature hi differently depending on context (the pres-
ence/absence of a disparity for the feature raised), context which is not itself
purely output in nature. The definition of IDENT[hi&rai] makes this relation-
ship symmetric: the constraint also evaluates a disparity on the feature raised
differently depending on the context (the presence/absence of a disparity for the
feature hi).

4.2.2 Distinction only at greater similarity
Recall the condition for the non-output-driven constraint behavior “distinction
only at greater similarity,” repeated here:

\[(4.3) \quad [C(bpy) < C(bmx)] \text{ and } [C(aoy) = C(akx)]\]

This is “distinction only at greater similarity” because the constraint doesn’t
distinguish the candidates akx and aoy, two candidates for the input with lesser
similarity to output outx, but does distinguish the candidates bmx and bpy, two
candidates for the input with greater similarity to output outx.

“Distinction only at greater similarity” is exhibited by the constraint that is
described in (4.6).

\[(4.6) \quad [\text{IDENT}[\text{low}] | \text{IDENT}[\text{hi}]_V, \text{ abbreviated IDENT}[\text{low}|\text{hi}]: \text{violated once} \text{ for}
\text{each IO correspondent pair that differs in the value of low or in the value of hi.}\]

This constraint will here be described as formed via the operation of local
disjunction. Both the structure of the operation and the operation’s name
are patterned after local conjunction. The local disjunction of two component
constraints (here, IDENT[low] and IDENT[hi]) is violated once when at least
one of the component constraints is violated within a defined domain (here, the
domain is a vowel). What is significant here is that violation is based solely
on the truth or falsity of the logical conditions within the domain: if both
component constraints are violated within a local domain, the local disjunction
is not violated twice, only once. By design, the constraint does not distinguish
between violation of one and violation of both within a local domain.
What is called local disjunction here is essentially the same as what Hewitt and Crowhurst (1996) proposed under the label “constraint conjunction,” not to be confused with local conjunction as defined in Section 4.2.1 (which is also commonly referred to as constraint conjunction). The tableau in (4.7) illustrates the behavior of IDENT[low|hi] relative to the individual constraints IDENT[low] and IDENT[hi], as well as to the local conjunction constraint IDENT[low&hi].

(4.7) Local disjunction

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>/a/[a]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/a/[e]</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>/a/[i]</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>/e/[a]</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>/e/[e]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/e/[i]</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>/i/[a]</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>/i/[e]</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>/i/[i]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The shaded cells highlight some of the differing non-ODP behavior for local conjunction and local disjunction. For both, recall that /e/[e] has greater internal similarity than /a/[e]. The constraint formed by local conjunction, IDENT[low&hi], exhibits distinction only at lesser similarity, distinguishing

---

1 The requirement that the constraint assess at most one violation for each locus of evaluation is important here, and it was important to the proposal of Hewitt and Crowhurst, who used the device to mask the gradient evaluation of one of the constraints being combined. The particular constraints constructed by Hewitt and Crowhurst in Hewitt and Crowhurst 1996 are of no particular interest here: they are constructed from markedness constraints, so the constructed constraints are markedness constraints, and therefore output-driven preserving.

The confusing nomenclature is a consequence of differing perspectives on constraint definition. Smolenskian local conjunction defines the conditions for violation of the constructed constraint as the logical conjunction of the conditions for violation of the factor constraints. Hewitt and Crowhurst style constraint conjunction defines the conditions for satisfaction of the constructed constraint as the logical conjunction of the conditions for satisfaction of the factor constraints. A simple application of the appropriate DeMorgan theorem reveals that [satisfaction of the constructed constraint as the conjunction of conditions for satisfaction of the factor constraints] is equivalent to [violation of the constructed constraint as the disjunction of conditions for violation of the factor constraints].
the candidates for /a/ (in favor of /a[e], crucially) but not for /e/. The constraint formed by local disjunction, \textsc{Ident}[\text{low}\mid\text{hi}], exhibits distinction only at greater similarity, distinguishing the key candidates for /e/ (in favor of /e[e], crucially), but not for /a/. Note that the issue is one of distinction, not violation vs. non-violation: both key candidates for /a/ violate \textsc{Ident}[\text{low}\mid\text{hi}], but they violate it an equal number of times.

This distinction only at greater similarity behavior is active in the non-output-driven map shown in (4.8).

\begin{tabular}{|c|c|c|c|c|c|}
\hline
 & *[\text{+low}] & \textsc{Ident}[\text{low}\mid\text{hi}] & *[\text{–hi}] & \textsc{Ident}[\text{low}] & \textsc{Ident}[\text{hi}] \\
\hline
/a/[a] & *! & * & & & \\
\hline
\textit{aoy} /a/[e] & * & *! & * & & \\
\hline
\textit{akx} /a/[i] & * & * & * & * & \\
\hline
/e/[a] & *! & * & * & * & \\
\hline
\textit{bpy} /e/[e] & *! & * & & & \\
\hline
\textit{bmx} /e/[i] & *! & * & & & \\
\hline
/i/[a] & *! & * & * & * & \\
\hline
/i/[e] & *! & * & & * & \\
\textit{akx} /i/[i] & * & & & & \\
\hline
\end{tabular}

\textit{in}_a = /a/ \quad \textit{in}_b = /e/ \quad \textit{out}_x = [i] \quad \textit{out}_y = [e]

\textit{akx} = /a/[i] \quad \textit{bmx} = /e/[i] \quad \textit{aoy} = /a/[e] \quad \textit{bpy} = /e/[e]

The map is /a/ \to [i], /e/ \to [e], /i/ \to [i]. This is a derived environment effect: the disparity –hi:+hi only occurs in combination with the disparity +low:–low, not on its own.

The “distinction only at greater similarity” behavior is crucially a consequence of the fact that the locally disjoined constraint \textsc{Ident}[\text{low}\mid\text{hi}] does not assess two violations when both of its disjuncts are violated at a single locus of violation (a single pair of input–output corresponding segments). In a candidate like /a/[i], where the locus of violation (the vowel) has identity disparities for the values of both low and hi, the constraint still only assesses a single violation, rendering it incapable of distinguishing violations of both component constraints from violation of only a single component constraint. The constraint cannot distinguish a disparity in low alone from a disparity in low paired with a disparity in hi (in the same vowel), despite the fact that it can distinguish a
disparity in hi from no disparity at all. The constraint exhibits distinction only at the greater similarity input (/e/), not at the lesser similarity input (/a/).

The locally conjoined constraint \textit{Ident[low&hi]} and the locally disjoined constraint \textit{Ident[low|hi]} have some similarities in their non-ODP behavior. Both are violated by candidates with disparities for both low and hi (maximally dissimilar with respect to the two features), and both are satisfied by candidates with disparities for neither low nor hi (maximally similar with respect to the two features). The locally conjoined constraint is unable to distinguish between one disparity and none, while the locally disjoined constraint is unable to distinguish between two disparities and one. Both of them “conflate” regions along the similarity path, but they conflate different regions. Both are \textit{monotonic} with respect to the relevant disparities: adding disparities to the domain never reduces the number of violations. Neither constraint is able to distinguish between a disparity for low alone and a disparity for hi alone: the conjoined constraint is violated by neither, while the disjoined constraint is violated by both.

In an interesting way, the locally disjoined constraint \textit{Ident[low|hi]} combines the two features into a single composite feature and evaluates segments with respect to identity on the composite. In an analysis where these are the only two features used, the effect is similar to that of basing internal similarity solely on segment identity, as was discussed in Section 2.2: from the point of view of the constraint, segments are either identical or they are not, with no further distinctions. The potential for non-output drivenness arises when an input-referring constraint like \textit{Ident[low|hi]}, which makes no further distinctions, is combined with a theory of relative similarity that \textit{does} make further distinctions.

\subsection{Distinction conflict}

The condition for distinction conflict is given in (4.2), repeated here.

\begin{equation}
C(bpy) < C(bmx) \quad \text{and} \quad C(aoy) > C(akx)
\end{equation}

A simple example of a constraint exhibiting distinction conflict is an input–output antifaithfulness constraint.\footnote{Input–output antifaithfulness is different from transderivational antifaithfulness (Alderete 1999a), which does not evaluate input–output correspondence but instead evaluates an output–output correspondence (Benua 1997) between a morphologically derived form and its base form. Constraints on output–output correspondence are non-stationary (see Section 4.8).} The constraint \textit{NonIdent[low]}, defined in (4.9), requires that an output segment be \textit{non-identical} to its input correspondent in the value of the feature low.

\begin{equation}
\text{NonIdent[low]} \quad \text{IO correspondents should differ in the value of low.}
\end{equation}
The tableaux in (4.10) show how \texttt{NonIdent[low]} is able, when sufficiently active, to cause a non-output-driven map.

(4.10) Tableaux showing distinction conflict for \texttt{NonIdent[low]}

<table>
<thead>
<tr>
<th></th>
<th>\texttt{NonIdent[low]}</th>
<th>\texttt{Ident[hi]}</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{aoy}</td>
<td>/a/[a]</td>
<td>*!</td>
</tr>
<tr>
<td>\texttt{akx}</td>
<td>/a/[e]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/a/[i]</td>
<td>*!</td>
</tr>
<tr>
<td>\texttt{bpy}</td>
<td>/e/[a]</td>
<td></td>
</tr>
<tr>
<td>\texttt{bmx}</td>
<td>/e/[e]</td>
<td>*!</td>
</tr>
<tr>
<td></td>
<td>/e/[i]</td>
<td>*!</td>
</tr>
<tr>
<td>\texttt{i}/</td>
<td>/a/[a]</td>
<td></td>
</tr>
<tr>
<td>\texttt{i}/</td>
<td>/e/[a]</td>
<td></td>
</tr>
<tr>
<td>\texttt{i}/</td>
<td>/i/[a]</td>
<td></td>
</tr>
</tbody>
</table>

\( \text{bmx} \) has greater internal similarity than \( \text{akx} \) because \( \text{bmx} \) lacks the mismatch in the value of feature low which \( \text{akx} \) has (there are no other disparities in these candidates). The map is non-output-driven because candidate \( \text{akx} \) is optimal, and \( \text{bmx} \) has greater internal similarity than \( \text{akx} \), yet \( \text{bmx} \) is not optimal.

This map includes a circular chain shift: \( a \rightarrow e, e \rightarrow a \). Moreton (2004) proved that in Optimality Theoretic systems in which all input-referring constraints assess no violations to candidates with no disparities, circular chain shifts are impossible. However, \texttt{NonIdent[low]} \textit{is} violated by candidates with no disparities, like /a/[a]. The violations assessed to candidates with no disparities are a key component of this constraint’s non-ODP behavior.

More fundamentally, \texttt{NonIdent[low]} is non-monotonic with respect to the relevant disparities: adding a disparity can actually reduce the number of violations of the antifaithfulness constraint, as in the comparison between /a/[a] and /a/[e]. The non-monotonicity is a significant difference between this constraint and the local conjunction and local disjunction constraints discussed previously.

\texttt{NonIdent[low]} exhibits “distinction conflict” behavior in this instance: the constraint prefers \( \text{akx} \) to \( \text{aoy} \), but prefers \( \text{bpy} \) to \( \text{bmx} \). The preference between competing candidates with outputs \( x \) and \( y \) reverses for the candidates for the
4.3 Relating non-ODP constraint behaviors to non-ODM map patterns

The result in Section 3.3.3 ensures that, for OT systems that are correspondence uniform, a map can be non-output-driven only through the activity of at least one constraint that is not ODP. Thus, the potential for non-output-driven maps in an OT system can be localized to specific constraints, and further localized to specific behaviors of those constraints. Presuming correspondence uniformity, any instance of non-output drivenness in a map is attributable to a constraint exhibiting one of the three kinds of non-ODP behavior described in Section 4.1.

The mere presence of a non-ODP constraint in a ranking does not by itself ensure that the map defined by the ranking is non-output-driven. The non-ODP constraint must be ranked appropriately with respect to other constraints. First, the constraint must be active; the “potential” for non-output drivenness will remain unrealized if the non-ODP constraint never has an opportunity to eliminate candidates. More specifically, the non-ODP constraint must be active on the candidates that characterize the non-ODP behavior, for at least one instance of that non-ODP behavior.

The non-ODP behaviors “distinction only at lesser similarity” and “distinction only at greater similarity” require cooperation from at least one lower-ranked constraint in order to actually realize non-output drivenness. Both distinguish candidates for two outputs with respect to one input, eliminating one of the candidates from that competition, but do not distinguish the corresponding candidates for the same two outputs with respect to the other input, leaving it to a lower-ranked constraint to decide between them. Non-output drivenness requires that the lower-ranked constraint choose, for the latter input, in favor of the candidate for the output whose candidate was eliminated by the non-ODP constraint for the former input. In a sense, the non-ODP constraint must work in cooperation with a lower-ranked constraint that chooses “the other way” in the case that the non-ODP constraint does not decide. Further, it must be the case that neither of these candidates is eliminated by any other constraint further down in the hierarchy (in favor of some other candidate still active in its respective competition).

In the example of “distinction only at lesser similarity” given in (4.5), the non-ODP constraint, IDENT[hi&rai], chooses [i] over [i’] for input /e/, but does
not distinguish between [i] and [i\textsuperscript{̃}] for input /i/. The lower-ranked constraint, \textsc{HiatusRaising}, chooses “the other way” for input /i/, choosing [i\textsuperscript{̃}] over [i]. In the example of “distinction only at greater similarity” given in (4.8), the non-ODP constraint, \textsc{Ident}[low|hi], chooses [e] over [i] for input /e/, but does not distinguish between [e] and [i] for input /a/. The lower-ranked constraint, *[–hi], chooses “the other way” for input /i/, choosing [i] over [e]. Notice that in both cases, if the lower-ranked constraint were instead ranked above the non-ODP constraint, the resulting map would be output-driven: the choice between the candidates would be decided the same way for both inputs.

Non-output-driven map instances have here been categorized into chain shift and derived environment effect patterns, with circular chain shifts identified as a special case of chain shifts. We can call these categories non-output-driven map patterns, or non-ODM patterns. There are three kinds of non-ODP behavior: distinction only at lesser similarity, distinction only at greater similarity, and distinction conflict. While it is tempting to look for strong, simple relationships between types of non-ODP constraint behaviors and the types of non-ODM map patterns they can cause, such relationships don’t exist. Some key observations are listed in (4.11).

(4.11) Key observations about non-ODP behaviors and non-ODM patterns
- A given non-ODP behavior can give rise to different non-ODM patterns.
- A given non-ODM pattern can result from different non-ODP behaviors.
- A single constraint can exhibit different non-ODP behaviors in different circumstances, and give rise to different non-ODM patterns in different circumstances.

The different non-ODM patterns are distinguished by the relationships between the input and output forms of the key candidates. In chain shifts, the candidate \textit{bmx}, with input \textit{in\textsubscript{b}} and output \textit{out\textsubscript{c}}, has no disparities: \textit{in\textsubscript{b}} = \textit{out\textsubscript{c}}. In other words, the candidate with greater internal similarity that fails to be grammatical has no disparities.\footnote{In a circular chain shift, the candidate that loses to \textit{akx}, \textit{aoy}, also has no disparities; \textit{in\textsubscript{b}} = \textit{out\textsubscript{c}}, as with all chain shifts, but additionally, \textit{out\textsubscript{y}} = \textit{in\textsubscript{a}}.} The Etxarri Basque example in Section 4.2.1 is such a case. In derived environment effects, the candidate with greater internal similarity that fails to be grammatical, \textit{bmx}, does have disparities: \textit{in\textsubscript{b}} \neq \textit{out\textsubscript{c}}. This is true of the examples discussed in Section 4.2.2. It is only the (non-)identity of \textit{in\textsubscript{b}} and \textit{out\textsubscript{c}} that distinguishes a chain shift from a derived environment effect.

The relationships between non-ODP behaviors and non-ODM patterns are further discussed in Section 4.11, after more examples have been presented.
4.4 Faithfulness conditioned on output context

4.4.1 Positional faithfulness I: position-specific Dep

Faithfulness constraints have been proposed that evaluate correspondence relations conditioned upon properties of the output. An example is Head-Dependence (Head-Dep), a version of Dep that is conditioned to be violable only by output segments that are in prosodic head positions (Alderete 1999b). Prosodic heads are frequent targets of stress assignment, and indeed much of the data motivating this constraint involves languages that actively avoid stressing epenthetic vowels. A stressed epenthetic vowel constitutes a violation of Head-Dep because the assignment of stress requires that the vowel be in a prosodic head position, and being epenthetic means having no input correspondent, that is, a violation of Dep.

\[(4.12) \text{Head-Dep: every output segment contained in a prosodic head has an input correspondent.}\]

The non-output-driven effects of Head-Dep are here illustrated with data from Dakota, shown in (4.13). The analysis of the data comes from Alderete 1999b; the data originate with Shaw 1976, 1985.

(4.13) Stress in Dakota (epenthized vowels are underlined)

\[
\begin{align*}
\text{čhikté} & \quad \text{“I kill you”} \\
\text{mayákte} & \quad \text{“you kill me”} \\
\text{wičháyakte} & \quad \text{“you kill them”} \\
\text{owičháyakte} & \quad \text{“you kill them there”}
\end{align*}
\]

Main stress in Dakota falls on the second syllable from the beginning by default.\(^4\) However, if the vowel of the second syllable is epenthetic, then main stress shifts to the first syllable. This requires that the grammar distinguish between surface vowels that are epenthetic and those that are not. The result is a non-output-driven map.

The interaction between epenthesis and stress is realized with Head-Dep, as shown in (4.14). The constraint MainFootLeft requires that the head foot of the word (which assigns main stress) appear at the beginning of the word; this constraint dominates all constraints that might otherwise pull the head foot away from the beginning of the word (like Ident[stress] and Weight-ToStress), thus keeping main stress on one of the first two syllables in longer words in Dakota.

\(^4\) If the word is monosyllabic, stress falls on the first (and only) syllable of the word: /kte/ → kté “s/he, it kills” (Shaw 1976, 1985).
Analysis of constraint behavior

(4.14) Non-output-driven map in Dakota

| $\approx$ akx | /čap/[čápa] | | |
| akx | /čap/[čapá] | $*$ | | |
| aoy | /čap/[čapá] | $*$ | | |
| bmx | /čap/[čapá] | $*$ | | |
| $\approx$ bpy | /čap/[čapá] | | |

$in_x = /čap/ \quad in_y = /čapá/ \quad out_x = [čápa] \quad out_y = [čapá]$

$akx = /čap/[čápa] \quad bmx = /čapá/[čápa] \quad aoy = /čap/[čapá] \quad bpy = /čapá/[čapá]$

bm$ has greater internal similarity than akx, yet it is not grammatical; bpy is grammatical in Dakota. Head-Dep exhibits distinction only at lesser similarity here. It prefers akx to aoy, but has no preference between bmx and bpy.

Head-Dep is an example of a positional faithfulness constraint (Beckman 1999): a faithfulness constraint that is restricted to the evaluation of segments that appear in a specified (normally prominent) position in the output. There are two ways to avoid a potential violation of a positional faithfulness constraint. One is to avoid the relevant faithfulness violation for the segment in the specified position. This is the kind of effect that normally motivates positional faithfulness constraints: the observation of greater contrast in prominent positions (Beckman 1999). However, a positional faithfulness constraint violation can also be avoided by not permitting the output segment with the relevant faithfulness violation to appear in the relevant kind of position. This is the kind of effect exhibited in the analysis of Dakota: the segment exhibiting the faithfulness violation (the epenthetic vowel) is prevented from appearing in the specified position (the prosodic head bearing main stress), accomplished by shifting the prosodic head position to the first syllable. The effect is achieved in the analysis of Dakota because of the existence of another constraint, Iambic, that is violated by the output in which the prominent position is shifted to avoid the faithfulness-violating segment (Iambic prevents the prosodic head from simply always being the first syllable).

The example in (4.14) has a particular pattern, one that will be repeated in several subsequent examples. First, observe that Head-Dep, as defined above, could alternately be expressed as a kind of local conjunction: the conjunction of Dep and $^*$Head in the domain of an output segment. This analysis is not intended to suggest that $^*$Head is an actual, independently
4.4 Faithfulness conditioned on output context

existing constraint; it is just a useful way to analyze the logical structure of Head-Dep. The constraint *Head is violated whenever the conditioning output context on Head-Dep is met: an output segment in a prosodic head position is a target of Head-Dep and violates *Head. In this analysis, we can call *Head a context constraint. Head-Dep is violated by an output segment that is in a head position (violating *Head) and has no input correspondent (violating Dep), and thus the locally conjoined constraint [*Head & Dep] is equivalent.

To repeat in terms of [*Head & Dep], there are two ways that this constraint can be satisfied. One is to avoid the faithfulness violation (the insertion disparity) by ensuring that the output segment has an input correspondent. The other is to avoid violation of the context constraint *Head by ensuring that the output segment is not in a prosodic head position. The stage is set for non-ODP constraint behavior when we compare a candidate that violates the conjoined constraint (by violating both the faithfulness conjunct and the context conjunct) with a candidate that avoids violating the conjoined constraint by satisfying the context conjunct. In (4.14), these are the candidates aoy and akx. Candidate aoy has the inserted second vowel, a violation of Dep, and has it in a head position, a violation of *Head. Candidate akx has the inserted second vowel, a violation of Dep, but that second vowel is not in a head position, avoiding violation of *Head, and thus avoiding violation of the conjoined constraint [Dep & *Head] (i.e., Head-Dep).

Candidate bm\text{x} is then chosen by choosing the input that, relative to out\text{x}, removes the disparity that violated the faith conjunct Dep in akx. In (4.14), that disparity was the insertion disparity of the second vowel. The disparity is removed by adding a vowel to in\text{b} that can serve as an input correspondent for the second vowel, forming input in\text{b} = /ćapa/ and candidate bm\text{x} = /ćapa/[ćápa]. The vowel added to the input is identical to the output correspondent, so that no new identity disparities are introduced. Because the only difference in bm\text{x} relative to akx is the removal of a disparity (the insertion disparity), bm\text{x} has greater internal similarity than akx.

The form of candidate bp\text{y} is jointly determined by akx, aoy, and bm\text{x}. Because it has input in\text{b}, bp\text{y} avoids the insertion disparity of aoy, just as in\text{b} allows bm\text{x} to avoid the same insertion disparity of akx. Because it has output out\text{y}, bp\text{y} violates the context constraint (*Head) in the second output vowel, just as aoy does. The key is that, because the change in input has eliminated the disparity that violated the faith conjunct, bp\text{y} will not violate the conjoined constraint, whether it violates the context conjunct *Head or not. This is the essence of distinction only at lesser similarity: the conjoined constraint (also
known as Head-Dep) distinguishes between $akx$ and $aoy$, (preferring $akx$), but does not distinguish between $bmx$ and $bpy$.

It bears emphasizing that the non-ODP quality of positional faithfulness constraints is not crucially dependent on the lack of constraints which enforce faithfulness to the specification of particular prosodic positions in the input. In fact, lexical stress languages require that the linguistic system allow the capability for the specification of main stress in inputs and constraints capable of preserving lexically specified stress in the output. Faithfulness to underlying stress is effectively faithfulness to a lexically specified prosodic position. In the analysis of Dakota above, the relevant constraints that prefer candidates that preserve lexical stress must be dominated in the ranking by the constraints determining the default stress pattern (here, MainFootLeft and Iambic).

4.4.2 Positional faithfulness II: position-specific Ident[F]

In the analysis of Dakota, the non-ODP constraint was a version of Dep with evaluation restricted to certain prominent positions (prosodic heads). Other positional faithfulness constraints have been proposed involving restrictions of Ident[F] to specified prominent positions. Ident-Onset[voice] (Lombardi 2001, Padgett 1995) evaluates voice identity only for input–output correspondents in which the output segment is in a syllable onset. This constraint is non-ODP. Beckman 1999: 36 note 27, citing personal communications with Rolf Noyer and John McCarthy, anticipated this when she observed that a potential violation of Ident-Onset[voice] can often be avoided in (at least) two ways: forcing the output correspondent in onset position to agree with its input correspondent, or locating the output correspondent in a non-onset position (i.e., in a coda). Syllabifying a consonant as a coda in order to alter its voicing is another instance of the kind of effect seen with Head-Dep in Dakota: the positional faithfulness constraint is satisfied by avoiding the type of position specified in the constraint. This potential for satisfying positional faithfulness through avoidance of the specified position sets the stage for non-ODP constraint behavior. This is illustrated in (4.15). The forms in the illustration are modified from Beckman’s original discussion of Catalan (Beckman 1999: 35–37).5

5 In Beckman’s analysis, /griz-a/ ‘gray (f.)’ surfaces as [gri.za] by virtue of having Onset dominate *VoicedObstruent.
4.4 Faithfulness conditioned on output context

Ident-Onset yields a non-output-driven map

<table>
<thead>
<tr>
<th>Ident-Onset[voice]</th>
<th>*VoicedObstruent</th>
<th>Onset</th>
<th>Ident[voice]</th>
</tr>
</thead>
<tbody>
<tr>
<td>/griza/[gri.zə]</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>aoy /griza/[gri.so]</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>akx /griza/[gris.ə]</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/grisa/[gri.za]</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bmx /grisa/[gris.ə]</td>
<td>*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bmx has greater internal similarity than akx: bmx has no disparities, and the same output as akx. Yet bmx is not grammatical; bpy is grammatical in this map. Ident-Onset[voice] exhibits distinction only at lesser similarity here. It prefers akx to aoy, but has no preference between bmx and bpy.

This analysis crucially depends upon the distinction between [gris.ə] and [gri.so] as outputs, that is, on whether the [s] is in the coda of the first syllable or the onset of the second. Output drivenness would not require bpy to be optimal, because it does not have the same output as akx (due to the difference in syllabification); bpy does not have greater internal similarity than akx. Ident-Onset[voice] distinguishes the candidates for the input yielding a disparity in voicing (/griza/), but does not distinguish the candidates for the input lacking the disparity in voicing (/grisa/). The non-ODP behavior results from satisfying the positional faithfulness constraint by avoiding the specified position (onset) without satisfying the faithfulness condition (identity in voicing).

Just as was done with Head-Dep, the constraint Ident-Onset[voice] can be expressed in the form of local conjunction between a faithfulness constraint (here, Ident[voice]) and a context constraint (here, *Onset) that is violated by any output segment in an onset position. Candidate aoy violates the conjoined constraint, while akx satisfies the conjoined constraint by virtue of satisfying the condition conjunct *Onset, while still violating the faith conjunct Ident[voice]. Candidate bmx alters the input to eliminate the disparity violating Ident[voice], thereby avoiding violation of the conjoined constraint, and making it possible for bpy to violate the condition conjunct.

6 In this example, the input for bmx also removes disparities in the quality of the final vowel; because no new disparities are introduced, bmx has greater internal similarity than akx.
without violating the conjoined constraint. Again, distinction only at lesser similarity is exhibited.

The illustration in (4.15) is subtle, depending on a recognition that two different outputs, [gris.ə] and [gri.sə], are in fact distinct despite the fact that both have the same overt forms. However, it is not hard to modify this example so as to make the distinction more overt, by introducing some other phonological property that differs between onset and coda positions. The illustration in (4.16) replaces the s/z pair in (4.15) with t/d, and then crucially adds a property, aspiration, to distinguish the voiceless alternant in the onset and coda positions. The voiceless alveolar stop is required to be aspirated in onset position and required to be unaspirated in coda position. This is enforced in (4.16) by the markedness constraint \( \text{Asp} \), which is little more than a placeholder for a more sophisticated analysis of laryngeal features.

(4.16) A non-output-driven map with aspiration distinguishing onset from coda

<table>
<thead>
<tr>
<th></th>
<th>IDENT-ONSET[voice]</th>
<th>*VOICEDObs</th>
<th>ONSET</th>
<th>IDENT[voice]</th>
<th>ASP</th>
<th>ID[asp]</th>
</tr>
</thead>
<tbody>
<tr>
<td>aoy</td>
<td>/grida/[gri.də]</td>
<td>*!</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/grida/[gri.tʰə]</td>
<td>*!</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>akx</td>
<td>/grida/[gri.ə]</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/grida/[gri.ta]</td>
<td>*!</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/grida/[gri.tʰa]</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*!</td>
<td>*</td>
</tr>
<tr>
<td>bmx</td>
<td>/grita/[gri.də]</td>
<td>*!</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/grita/[gri.tʰə]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>bpy</td>
<td>/grita/[gri.tʰa]</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

\( in_x = /\text{grida}/ \quad in_y = /\text{gri}/ \quad out_x = [\text{gri.ə}] \quad out_y = [\text{gri.tʰə}] \)

\( akx = /\text{grida}/[\text{gri.ə}] \quad bmx = /\text{gri}/[\text{gri.ə}] \quad aoy = /\text{grida}/[\text{gri.tʰə}] \quad bpy = /\text{gri}/[\text{gri.tʰə}] \)

The only significance to the example in (4.16) is that the outputs \( out_x \) and \( out_y \) are now overtly distinct: \( \text{grida} \rightarrow \text{gri.ə} \rightarrow \text{gri.tʰə} \).

4.4.3 Conjoined markedness and faithfulness

Positional faithfulness constraints restrict the evaluation of faithfulness to specified positions, but the same kind of effect can in principle be achieved by any
output-based restriction on the evaluation of faithfulness. An example is the analysis of Polish spirantization by Łubowicz 2002.7

Łubowicz’s analysis is partly inspired by a process-based analysis by Rubach 1984, which characterizes the phenomenon as the interaction of two processes, velar palatalization and velar spirantization. In Rubach’s analysis, velars change to postalveolars (palatalize) before front vocoids. In the same environment, voiced velars not only palatalize but also spirantize, becoming voiced postalveolar fricatives. Łubowicz represents the changes using the features [coronal] and [continuant]. When an underlying voiced velar /g/ occurs before a front vocoid, it palatalizes to [ǰ] (changing to +coronal), and then spirantizes to [ź] (changing to +continuant). Crucially, an underlying /j/ does not spirantize. Example data are shown in (4.17).

(4.17) Polish spirantization: derived ǰ spirantizes to ź, but underlying j does not

rożek  “horn”  /rog+ek/ → [rożek]
brjiek  “bridge”  (dim.)  /brj+îk+i/ → [brjiek]

The key challenge attacked by Łubowicz is the apparent sensitivity of spirantization to derivation: it only applies to segments derived by palatalization. Łubowicz analyzes this sensitivity with the constraint [*j & IDENT[coronal]]_seg; the local domain is a pair of IO correspondents, with the markedness constraint evaluating only the output correspondent. An analysis of the palatalization process itself will not be given here, and is assumed to be the consequence of higher-ranked constraints. The analysis of spirantization is shown in (4.18); to make the non-output-drivenness more apparent, I’m using a hypothetical form, [rojek], to form a minimal pair of the relevant sort with [rożek].

(4.18) Derived environment effect in Polish spirantization

<table>
<thead>
<tr>
<th></th>
<th>[*j &amp; IDENT[coronal]]</th>
<th>IDENT[continuant]</th>
</tr>
</thead>
<tbody>
<tr>
<td>aoy</td>
<td>/rogek/[rojek]</td>
<td>*!</td>
</tr>
<tr>
<td>akx</td>
<td>/rogek/[rojek]</td>
<td>*</td>
</tr>
<tr>
<td>bpy</td>
<td>/rojek/[rojek]</td>
<td></td>
</tr>
<tr>
<td>bmx</td>
<td>/rojek/[rožek]</td>
<td>*!</td>
</tr>
</tbody>
</table>

\[\text{in}_a = /rogek/ \quad \text{in}_b = /rojek/ \quad \text{out}_x = [rožek] \quad \text{out}_y = [rojek] \]
\[\text{akx} = /rogek/[rožek] \quad \text{bm}x = /rojek/[rožek] \quad \text{aoy} = /rogek/[rojek] \quad \text{bpy} = /rojek/[rojek] \]

7 For additional discussion of constraints formed by the local conjunction of markedness and faithfulness constraints, see Itô and Mester 2003.
The constraint \([*] & \text{IDENT}[\text{coronal}]\) exhibits distinction only at lesser similarity here, and in combination with \text{IDENT}[\text{continuant}] yields a derived environment effect. Candidate /ro\~zek/[\rozek] is optimal, having disparities for coronal and continuant, yet candidate /ro\~jek/[\rojk], with only a disparity in continuant, is not optimal, losing to /ro\~jek/[\rojk].

The constraint \([*j] & \text{IDENT}[\text{coronal}]\) is the local conjunction of a markedness constraint, \(*j\), and a faithfulness constraint, \text{IDENT}[\text{coronal}]. The conjoined constraint is violated by a pair of IO corresponding segments if they disagree on the value of coronal and the output correspondent is the segment \([j]\). Unlike \text{HEAD-DEP} and \text{IDENT-ONSET}[\text{voice}], this constraint already is in local conjunction form. The description could be translated back into the terms of positional faithfulness by taking the markedness constraint to indicate the specified position. In the case of \([*j] & \text{IDENT}[\text{coronal}]\)_{seg}, the “specified position” is an occurrence of the output segment \([j]\): the constraint enforces agreement in the feature coronal in the \([j]\) “position.” The output segment \([j]\) is not a prosodic position, but it is an output context that functions the same way in \([*j] & \text{IDENT}[\text{coronal}]\) as the output contexts prosodic head and onset function in \text{HEAD-DEP} and \text{IDENT-ONSET}[\text{voice}], respectively.

In (4.18), \(a\~yo\) violates both conjuncts within a single local domain, and so violates the conjoined constraint. Candidate \(a\~ko\) avoids violation of the conjoined constraint by satisfying the context conjunct \(*j\), that is, by having the output correspondent not be \([j]\). In this case, a change in the value of continuant is used to distinguish the output correspondent from \([j]\), yielding \([\\~\tilde{z}]\) in the output of \(a\~k\). Candidate \(b\~m\) is formed from \(a\~k\) by changing the input to remove the disparity in coronal that violated the faith conjunct (but not removing the disparity in continuant). This elimination of the coronal disparity eliminates the violation of the faith conjunct, meaning that the conjoined constraint will be satisfied whether the context conjunct is violated (as in \(b\~p\)) or not (as in \(b\~m\)). Once again, distinction only at lesser similarity is exhibited.

One notable difference between Polish spirantization and the previous cases of Dakota stress shift and onset voicing preservation is in the way that the context conjunct violation is avoided. In the Dakota stress shift, the context violation was avoided by changing the location of the prosodic head, a change in output that does not directly affect IO disparities. Likewise, in the onset voicing preservation case, the context violation was avoided by moving the locus output segment out of an onset position, also a change that does not directly affect IO disparities. In the Polish spirantization case, however, the output context is avoided by changing the value of the continuant feature on the
4.4 Faithfulness conditioned on output context

locus output segment. Segment feature values are relevant to IO disparities, and
the result is that violation of the context conjunct *j is avoided by introducing a
different disparity, one in the value of the feature continuant. When the greater
similarity candidate, $b_{mx}$, is constructed by removing the disparity in coronal,
the conjoined constraint is satisfied regardless of the (non)-satisfaction of the
output context conjunct, *j, and thus the motivation for the other disparity, the
disparity in continuant, is removed. When the disparity in coronal is removed,
the disparity in continuant is no longer tolerated.

4.4.4 Value-restricted D E P

As proven in Section 3.6.5, $\text{D e p}[\text{voi}_{\text{out}} \in \{-\text{voi}\}]$ is ODP. What bears
emphasizing here is that the constraint only applies to output segments that
have the feature value $-\text{voi}$ and have input correspondents. For those output
segments, the constraint is only satisfied if the output context feature value,
$-\text{voi}$, is identical in the input correspondent. Thus, the output context referred
to by the constraint, $-\text{voi}$, must be identical to the input correspondent’s voice
value in order to satisfy the constraint.

Things are very different for a constraint here called $\text{D e p}[\text{voi}_{\text{out}} \in \{-\text{voi}\}]$.
This constraint is like $\text{D e p}$, but can only be violated by output segments that
have the feature value $-\text{voi}$ and do not have input correspondents. The key
difference between $\text{D e p}[\text{voi}_{\text{out}} \in \{-\text{voi}\}]$ and $\text{D e n t}[\text{voi}_{\text{out}} \in \{-\text{voi}\}]$ is that,
for an output segment $s_{\text{out}}$ with the feature value $-\text{voi}$, $\text{D e p}[\text{voi}_{\text{out}} \in \{-\text{voi}\}]$
is satisfied if $s_{\text{out}}$ has an input correspondent even if the input correspon-
dent does not have the value $-\text{voi}$. In terms of the disparities that constitute
violations, every disparity that violates $\text{D e p}[\text{voi}_{\text{out}} \in \{-\text{voi}\}]$ also violates
$\text{D e p}$, just as every disparity that violates $\text{D e n t}[\text{voi}_{\text{out}} \in \{-\text{voi}\}]$ also violates
$\text{D e n t}[\text{voi}]$. But the failure to require identity of the context feature value
makes $\text{D e p}[\text{voi}_{\text{out}} \in \{-\text{voi}\}]$ not ODP.

The failure of $\text{D e p}[\text{voi}_{\text{out}} \in \{-\text{voi}\}]$ to be ODP can be illustrated in the
context of coda conditions on voicing. The constraint Final C (Prince and
Smolensky 1993/2004) requires that the final segment in a word be a consonant.
As shown in (4.19), $\text{D e p}[\text{voi}_{\text{out}} \in \{-\text{voi}\}]$ is violated when an output voiceless
segment, here $[k]$, has no input correspondent, but is satisfied when an output
voiced segment, like $[g]$, has no input correspondent. When a candidate with
greater internal similarity is formed by adding an input correspondent /g/ to
the input (eliminating the insertion disparity for $[g]$), $\text{D e p}[\text{voi}_{\text{out}} \in \{-\text{voi}\}]$
is also satisfied by a competing candidate with final output consonant $[k]$
corresponding to input /g/, even though they have non-identical values for
voice. This results in a chain shift: ti $\rightarrow$ tig $\rightarrow$ tik.
(4.19) \( \text{Dep}[\text{voi}_{\text{out}} \in \{-\text{voi}\}] \) induces a chain-shift

<table>
<thead>
<tr>
<th></th>
<th>FinalC</th>
<th>Max</th>
<th>( \text{Dep}[\text{voi}_{\text{out}} \in {-\text{voi}}] )</th>
<th>NoVoiCoda</th>
<th>( \text{Dep} )</th>
<th>( \text{Ident}[\text{voi}] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \varphi ) \text{akx}</td>
<td>/t/\text{[tig]}</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>/t/\text{[ti]}</td>
<td>!</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \text{aoy} )</td>
<td>/t/\text{[tik]}</td>
<td>!</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/t/\text{[i]}</td>
<td>!*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \text{bmx} )</td>
<td>/t/\text{[tig]}</td>
<td>*!</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/t/\text{[ti]}</td>
<td>!</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \varphi ) \text{bpy}</td>
<td>/t/\text{[tik]}</td>
<td>!</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/t/\text{[i]}</td>
<td>!*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( \text{in}_a = /t/ \quad \text{in}_b = /t/\text{[tig]} \quad \text{out}_s = [\text{tig}] \quad \text{out}_r = [\text{tik}] \)

\( \text{akx} = /t/\text{[tig]} \quad \text{bmx} = /t/\text{[tig]} \quad \text{aoy} = /t/\text{[tik]} \quad \text{bpy} = /t/\text{[tig]} \)

\( \text{bmx} \) has greater internal similarity than \( \text{akx} \), yet it is not grammatical.

The constraint \( \text{Dep}[\text{voi}_{\text{out}} \in \{-\text{voi}\}] \) can be expressed as the local conjunction of the faithfulness constraint \( \text{Dep} \) and the (markedness) context constraint *\(-\text{voi}\), in the domain of an output segment. Candidate \( \text{aoy} \) violates the conjoined constraint because it violates both of the conjuncts in the third segment of \( \text{out}_s \), while \( \text{akx} \) avoids violation of the conjoined constraint by avoiding violation of the context constraint *\(-\text{voi}\), inserting a voiced [g] instead of voiceless [k]. When \( \text{bmx} \) is constructed by changing the input to remove the insertion disparity, the violation of the faithfulness conjunct, \( \text{Dep} \), is eliminated, and along with it the pressure from the conjoined constraint to respect the context constraint *\(-\text{voi}\).

4.4.5 \textit{Summary: independent context}

Like \( \text{Dep}[\text{voi}_{\text{out}} \in \{-\text{voi}\}] \), \( \text{Ident}[\text{voi}_{\text{out}} \in \{-\text{voi}\}] \) can also be expressed as a local conjunction, between \( \text{Ident}[\text{voice}] \) and *\(-\text{voi}\). Yet, \( \text{Ident}[\text{voi}_{\text{out}} \in \{-\text{voi}\}] \) is ODP. The reason why is illustrated in (4.20). Candidate \( \text{aoy} \) violates the conjoined constraint in the final consonant: it has a disparity in voicing, and the output correspondent is \(-\text{voi}\). If we follow the pattern used on the other faithfulness constraints conditioned on output context, we start with a candidate that violates the conjoined constraint, say \( \text{aoy} = /\text{rad}/[\text{rat}] \). Candidate \( \text{akx} \) should be constructed so that the violation of the context conjunct, *\(-\text{voi}\), is avoided, while the faithfulness conjunct is violated. This is done by voicing the final segment of the output. But, as shown in (4.20), satisfying the output context conjunct in this way has the consequence of also satisfying the faithfulness conjunct! Candidate \( \text{akx} \) violates neither of the conjuncts and preserves the segment in question unaltered. As a result, it isn’t possible to follow the pattern
Faithfulness conditioned on input context

4.5 Faithfulness conditioned on input context

4.5.1 Value-restricted Max

As proven in Section 3.6.4, Ident[voi in ∈ [+voi]] is ODP. What bears emphasizing here is that the constraint only applies to input segments that have the feature value +voi and have output correspondents. For those input segments, the constraint is only satisfied if the conditioning input context feature value, +voi, is identical in the output correspondent. Thus, the input condition referred to by the constraint, +voi, must be identical to the output in order to satisfy the constraint.

Things are very different for a constraint here called Max[voi in ∈ [+voi]]. This constraint is like Max, but can be violated only by input segments that have the feature value +voi and do not have output correspondents. The key difference between Max[voi in ∈ [+voi]] and Ident[voi in ∈ [+voi]] is that, for an input segment s_in with the feature value +voi, Max[voi in ∈ [+voi]] is satisfied if s_in has an output correspondent even if the output correspondent
does not have the value +voi. In terms of the disparities that constitute violations, every disparity that violates \( \text{Max}[\text{voi} \in \{+\text{voi}\}] \) also violates \( \text{Max} \), just as every disparity that violates \( \text{Ident}[\text{voi} \in \{+\text{voi}\}] \) also violates \( \text{Ident} \). But the failure to require identity of the context feature value makes \( \text{Max}[\text{voi} \in \{+\text{voi}\}] \) not ODP.

\( \text{Max}[\text{voi} \in \{+\text{voi}\}] \) is very different from the constraint \( \text{MaxLaryngeal} \) proposed by Lombardi (Lombardi 2001). Although Lombardi describes this constraint in terms of a direct correspondence between features (specifically, between autosegments in the input and in the output), in her analysis the constraint appears to behave like a combination of \( \text{Ident} \) and \( \text{Max} \): if a segment in the input is +voi, then that segment must have an output correspondent and the output correspondent must be +voi. The latter condition distinguishes \( \text{MaxLaryngeal} \) from \( \text{Max}[\text{voi} \in \{+\text{voi}\}] \).

The failure of \( \text{Max}[\text{voi} \in \{+\text{voi}\}] \) to be ODP can be illustrated in the context of coda conditions on voicing. The constraint \( \text{NoVoiCoda} \) (Itô and Mester 1997) requires that a segment in coda position must be voiceless. As shown in (4.21), \( \text{Max}[\text{voi} \in \{+\text{voi}\}] \) is violated when an underlyingly voiced segment, here /g/, has no output correspondent, but is satisfied if /g/ has a voiceless output correspondent. When a candidate with greater internal similarity is formed by devoicing this segment to /k/ in the input (eliminating the voicing disparity for /g/), \( \text{Max}[\text{voi} \in \{+\text{voi}\}] \) is satisfied by a competing candidate in which /k/ has no output correspondent, because it is not underlyingly voiced. This results in a chain shift: tig → tik → ti.

(4.21) \( \text{Max}[\text{voi} \in \{+\text{voi}\}] \) induces a chain shift

<table>
<thead>
<tr>
<th>( \text{NoVoiCoda} )</th>
<th>Dep</th>
<th>( \text{Max}[\text{voi} \in {+\text{voi}}] )</th>
<th>( \text{NoCoda} )</th>
<th>( \text{Max} )</th>
<th>( \text{Ident}[\text{voi}] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{akx} )</td>
<td>/tig/[tik]</td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>/tig/[tig]</td>
<td>*!</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>( \text{aoy} )</td>
<td>/tig/[ti]</td>
<td></td>
<td>*!</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/tig/[ti.ga]</td>
<td></td>
<td></td>
<td>*!</td>
<td></td>
</tr>
<tr>
<td>( \text{bmx} )</td>
<td>/tik/[tik]</td>
<td></td>
<td></td>
<td></td>
<td>*!</td>
</tr>
<tr>
<td></td>
<td>/tik/[tig]</td>
<td></td>
<td></td>
<td></td>
<td>*!</td>
</tr>
<tr>
<td>( \text{bpy} )</td>
<td>/tik/[ti]</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>/tik/[ti.ka]</td>
<td></td>
<td></td>
<td></td>
<td>*!</td>
</tr>
</tbody>
</table>

\[ \text{in}_a = /\text{tig}/ \quad \text{in}_b = /\text{tik}/ \quad \text{out}_a = [\text{tik}] \quad \text{out}_b = [\text{ti}] \]
\[ \text{akx} = /\text{tig}/[\text{tik}] \quad \text{bmx} = /\text{tik}/[\text{tik}] \quad \text{aoy} = /\text{tig}/[\text{ti}] \quad \text{bpy} = /\text{tik}/[\text{ti}] \]
\( \text{bmx} \) has greater internal similarity than \( \text{akx} \), yet it is not grammatical.

8 Under this interpretation, \( \text{MaxLaryngeal} \) is equivalent to the local disjunctive constraint \( [\text{Max}[\text{voi} \in \{+\text{voi}\}] | \text{Ident}[\text{voi} \in \{+\text{voi}\}] \] \).
The constraint $\text{Max}[\text{voi}_{\text{in}} \in \{+\text{voi}\}]$ can be expressed as the local conjunction of the faithfulness constraint $\text{Max}$ and the input context “constraint” $*(\text{Fin} = +\text{voi})$, in the domain of an input segment. The scare quotes on “constraint” are deserved here, because a constraint that solely evaluates the input is normally of little use in an OT analysis: it necessarily assigns the same number of violations to competing candidates, because competing candidates necessarily have the same input. Its value here is solely as an expression of the input context, so that $\text{Max}[\text{voi}_{\text{in}} \in \{+\text{voi}\}]$ can be expressed as $[\text{Max} \&* (\text{Fin} = +\text{voi})]$.

Candidate $aoy$ violates the conjoined constraint because it violates both of the conjuncts in the third segment of $\text{ina}$, while $akx$ avoids violation of the conjoined constraint by avoiding violation of the faith conjunct $\text{Max}$ (avoiding violation of the input context isn’t an option). Crucial to this example is that other constraints, $\text{NoVoICoda}$ and $\text{Dep}$, force the $\text{Max}$-satisfying output correspondent to introduce a disparity in voice, the very feature that is the focus of the input context. This disparity in voicing in the optimal candidate creates the room for a candidate with a different input that has greater internal similarity, by changing the input to eliminate this disparity.

When $bmx$ is constructed by changing the input to remove the identity disparity in voicing, the violation of the input context conjunct, $*(\text{Fin} = +\text{voi})$, is also eliminated. Because input $\text{in}b$ does not violate the input context conjunct, none of the competitors will violate the input context constraint, meaning that none of the competitors will violate the conjoined constraint, even a competitor that violates the faith conjunct $\text{Max}$, such as $bpy$. This is another instance of distinction only at lesser similarity.

While $\text{Max}[\text{voi}_{\text{in}} \in \{+\text{voi}\}]$ happens to be conditioned on the feature value $+\text{voi}$, there is nothing specific to voicing about the non-ODP behavior. In principle, it could be reproduced with any feature given the appropriate markedness constraints (analogous to $\text{NoVoICoda}$ and $\text{NoCoda}$). Constraints like $\text{Max}[C]$ (C for “consonant”) and $\text{Max}[V]$ (V for “vowel”) have the same potential, provided that $\text{Gen}$ permits candidates in which input vowels can correspond to output consonants (and vice-versa), and the constraints are satisfied by such correspondences. Note that if the candidates with such

9 The origins of $\text{Max}[C]$ and $\text{Max}[V]$ can be traced to their pre-correspondence theory counterparts in containment faithfulness theory, $\text{Parse}^C$ and $\text{Parse}^V$ (Prince and Smolensky 1993/2004: 256).

10 Note that if the conditions defining “C” and “V” aren’t properties of segments themselves and aren’t marked in any way in the input (such as if they were defined solely in terms of syllabic positions in the output), then the constraints become incoherent: you cannot determine if a segment lacking an output correspondent is a “C” or a “V,” and thus cannot determine if the segment constitutes a violation or not.
correspondents simply aren’t permitted by Gen, then Max[C] and Max[V] will not exhibit non-ODP behavior in this fashion (all other things being equal). The non-output-driven map caused by Max[voi in ∈ {+voi}] depends crucially on the existence of candidate /tīg/ [tik], where the conditioning voiced input segment has a voiceless output correspondent. If such a candidate cannot be generated, then it cannot be optimal.

4.5.2 Relation to output-conditioned faithfulness
The pattern for input-conditioned faithfulness to exhibit distinction only at lesser similarity is a bit different from the one for output-conditioned faithfulness. In Section 4.4.4, it was shown that Dep[voi out ∈ {–voi}] can be expressed as [Dep & *(–voi)], or equivalently, [Dep & *(Fout = –voi)]. The non-output-driven situation in (4.19) has aoy violating the conjoined constraint, and akx satisfying the constraint by virtue of avoiding violation of the output context conjunct, *(Fout = –voi). Max[voi in ∈ {+voi}] can be expressed as [Max & *(Fin = +voi)], and the non-output-driven situation in (4.21) also has aoy violating the conjoined constraint. However, it is not possible to form a candidate akx that avoids the input context conjunct: competitors by definition have the same input. Instead, akx avoids violation of the other conjunct, the faith conjunct Max.

In the output-conditioned faithfulness pattern, candidate bmx is formed by changing the input to eliminate the disparity that violated the faith conjunct in akx. In the input-conditioned case, candidate bmx is formed by changing the input to eliminate the input value that violates the context conjunct in akx. In both cases, candidate bmx is formed by changing the input to eliminate the violation of the conjunct that appeared in akx. That elimination then ensures that the conjoined constraint will be satisfied, allowing competitors to bmx to violate the other conjunct (the one not violated in akx) without running afoul of the conjoined constraint.

Max[voi in ∈ {+voi}] is non-ODP in part because it does not preserve its conditioning context, (Fin = +voi): it can be satisfied by an output correspondent with a different voice feature value. It is similar in this respect to Dep[voi out ∈ {–voi}], which can be satisfied by an input correspondent with a different voice feature value. Both constraints have a conditioning context that is independent of the kind of faithfulness being evaluated; precisely the independence that the ODP constraints Ident[voi in ∈ {+voi}] and Ident[voi out ∈ {–voi}] lack.
4.6 Multiply conditioned faithfulness

4.6.1 Joint input–output value restrictions

Constraints of the form $\text{IDENT}[F_{in} \in \{\alpha\}, F_{out} \in \{\delta\}]$ have value restrictions on both the input and output correspondents. The constraint is vacuously satisfied unless both the input and output restrictions are satisfied. It is only possible to non-vacuously satisfy such a constraint if there is some overlap in feature values between the input and output restrictions. If there is no overlap, such as in $\text{IDENT}[\text{voi}_{in} \in \{+\text{voi}\}, \text{voi}_{out} \in \{-\text{voi}\}]$, then any correspondents that don’t vacuously satisfy the constraint necessarily have distinct input and output values and violate the constraint. If the input and output correspondents are restricted to have the same single value of the feature, such as in $\text{IDENT}[\text{voi}_{in} \in \{+\text{voi}\}, \text{voi}_{out} \in \{+\text{voi}\}]$, the constraint will never be violated: satisfaction of the value restrictions necessarily entails that the input and output feature values are identical.

To get a constraint of this sort to be non-ODP requires that the feature have at least three distinct values. This is illustrated in (4.22), using a place feature pl with three values: coronal, labial, and dorsal.

(4.22) $\text{IDENT}[\text{pl}_{in} \in \{\text{dor}\}, \text{pl}_{out} \in \{\text{cor}\}]$ induces a chain shift

<table>
<thead>
<tr>
<th></th>
<th>$\text{Max}$</th>
<th>*$\text{Dor}$</th>
<th>$\text{IDENT}[\text{pl}<em>{in} \in {\text{dor}}, \text{pl}</em>{out} \in {\text{cor}}]$</th>
<th>*$\text{Lab}$</th>
<th>$\text{IDENT}[\text{pl}]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varnothing$ akx $/\text{tik}/[\text{tip}]$</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$/\text{tik}/[\text{tik}]$</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$/\text{tik}/[\text{tit}]$</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$/\text{tik}/[\text{ti}]$</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>$\varnothing$ bmx $/\text{tip}/[\text{tip}]$</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$/\text{tip}/[\text{tik}]$</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$/\text{tip}/[\text{tit}]$</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\text{in}_{a} = /\text{tik}/ \quad \text{in}_{b} = /\text{tip}/ \quad \text{out}_{x} = [\text{tip}] \quad \text{out}_{y} = [\text{tit}]$

$\text{akx} = /\text{tik}/[\text{tip}] \quad \text{bmx} = /\text{tip}/[\text{tip}] \quad \text{aoy} = /\text{tik}/[\text{tit}] \quad \text{bpy} = /\text{tip}/[\text{tit}]$

$bmx$ has greater internal similarity than $akx$, yet it is not grammatical.

The key behavior of the constraint $\text{IDENT}[\text{pl}_{in} \in \{\text{dor}\}, \text{pl}_{out} \in \{\text{cor}\}]$ lies in the shaded cells, which reveal distinction only at lesser similarity. In the first competition, the constraint penalizes the output coronal coda relative to
the output labial coda, while in the second competition the constraint does not distinguish the two. In both competitions, the output coronal coda does not match its input correspondent in the value of place, but the disparity only violates the constraint in the first competition, when the input correspondent has dorsal place.

One way to think about this example is with respect to a place markedness scale, with dorsal more marked than labial more marked than coronal. If dorsal place in the input is altered solely to avoid the markedness of dorsal, one would by default expect the place feature to be changed to the least marked value, coronal. The constraint $\text{Ident}[\text{pl}_{\text{in}} \in \{\text{dor}\}, \text{pl}_{\text{out}} \in \{\text{cor}\}]$, however, explicitly penalizes a change in place from most marked to least marked (on this simple three-valued scale). The result is an output place feature value of labial, which is more marked than coronal but less marked than dorsal. When an input segment has labial place, however, it vacuously satisfies $\text{Ident}[\text{pl}_{\text{in}} \in \{\text{dor}\}, \text{pl}_{\text{out}} \in \{\text{cor}\}]$, and place is allowed to change to coronal.

If the constraint is viewed as penalizing candidates which change a place feature “two steps” along the place markedness scale, then the constraint resembles Gnanadesikan’s $\text{Ident-Adj}[F]$ constraint (Gnanadesikan 1997), which is violated whenever input and output correspondents have feature values that are more than one step apart on the scale of values for that feature.

4.6.2 Conditioning on disparities

Instead of conditioning $\text{Ident}[F]$ on both input and output values of $F$ for a given pair of IO correspondents, it is possible to condition $\text{Ident}[F]$ on the presence of another disparity, such as an $\text{Ident}$ disparity with respect to a different feature. In fact, we’ve already seen a constraint of this sort: the conjoined faithfulness constraint $[\text{Ident}[\text{hi}] \& \text{Ident}[\text{raised}]]_V$ in Section 4.2.1. The distinction between faithfulness conjunct and conditioning conjunct is no longer significant: both conjuncts are faithfulness constraints, and either one can “condition” the other.

The non-ODP behavior of $[\text{Ident}[\text{hi}] \& \text{Ident}[\text{raised}]]_V$ can be characterized in terms of the independence of the two conditions. Recall the chain shift map of (172): $/e/ \rightarrow [i] \rightarrow [i^\prime]$. For $/e/ \rightarrow [i]$, the conjoined constraint prevents the optimality of the candidate $/e/[i^\prime]$, with disparities in both features $\text{hi}$ and $\text{raised}$. Put another way, the conjoined constraint penalizes a disparity in $\text{raised}$ in the context of a disparity in $\text{hi}$. However, the features $\text{hi}$ and $\text{raised}$
have a limited degree of independence. In combination with [–hi] only the value [–raised] is possible, but in combination with [+hi] both values [–raised] and [+raised] are possible. This limited degree of independence is sufficient to allow the construction of a candidate with a disparity in hi but not in raised, /e/[i]. Once a new input is selected and the disparity in hi is eliminated, via /i/[i], then a competitor with a raised output, /i/[i^y], will not violate the conjoined constraint. In this case, partial independence is enough to permit distinction only at lesser similarity.

4.7 Conditioned antifaithfulness

4.7.1 Material implication constraints

Wolf 2006, 2007 has shown that it is possible to get circular chain shifts with constraints constructed by material implication. A complex constraint constructed via material implication has the logical form $C_1 \rightarrow C_2$: for a specified locus of evaluation (such as a segment), if constraint $C_1$ is satisfied, then constraint $C_2$ must also be satisfied in order to satisfy the complex constraint (Archangeli et al. 1998, Balari et al. 2000). Note that here the conditions for the satisfaction of the complex constraint are characterized in terms of the conditions for satisfaction of the component constraints. Wolf discusses constraints of the form $F \rightarrow M$, where $F$ is a faithfulness constraint and $M$ is a markedness constraint.

Wolf’s result can be illustrated with the implicationally defined constraint $\text{Ident}[\text{low}] \rightarrow *[–hi]$. The faithfulness constraint $\text{Ident}[\text{low}]$ is the same as defined in (4.4). The markedness constraint $*[–hi]$ is violated by any output vowel with the feature value –hi. The constraint $\text{Ident}[\text{low}] \rightarrow *[–hi]$ has as its locus of violation a pair of IO corresponding vowels, and if the vowels match on the value of the feature low, then the output correspondent vowel must not have the feature value –hi, or else a violation (of $\text{Ident}[\text{low}] \rightarrow *[–hi]$) is assessed to that pair of IO corresponding vowels. Any part of a candidate that vacuously satisfies $\text{Ident}[\text{low}]$ (such as an output vowel with no input correspondent) also vacuously satisfies $\text{Ident}[\text{low}] \rightarrow *[–hi]$. The use of this constraint to achieve a circular chain shift map is shown in (4.23).

11 The definition of construction via implication discussed here is distinct from the definition proposed by Crowhurst and Hewitt (1997); see (Balari et al. 2000, Wolf 2006) for explanatory discussion.
(4.23) Circular chain shift with an implicationally defined constraint

<table>
<thead>
<tr>
<th></th>
<th>*[hi]</th>
<th>IDENT[low] → *[-hi]</th>
<th>IDENT[hi]</th>
</tr>
</thead>
<tbody>
<tr>
<td>aoy</td>
<td>/a[a]</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>♂ akx</td>
<td>/a[e]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/a[i]</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>♂ bpy</td>
<td>/e[a]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>bmx</td>
<td>/e[e]</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td></td>
<td>/e[i]</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>♂ i</td>
<td>/i[a]</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>/i[e]</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td></td>
<td>/i[i]</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

This map is a circular chain shift. From the point of view of ODP constraint behavior, we can observe that the implicationally defined constraint, IDENT[low] → *[-hi], is non-ODP: it exhibits distinction conflict in this instance. Furthermore, the map in (4.23) is identical to the one in (4.10) involving NONIDENT[low]. The distinction conflict behavior of IDENT[low] → *[-hi] is also identical to that of NONIDENT[low], because the two constraints evaluate the four key candidates akx, bmx, aoy, and bpy identically. The two candidates evaluated differently by NONIDENT[low] and IDENT[low] → *[-hi], /e[i] and /i/i[i], are handled by the higher-ranked *[hi] in (4.23). The connection between IDENT[low] → *[-hi] and NONIDENT[low] is not accidental, as will be shown in Section 4.7.2: IDENT[low] → *[-hi] is equivalent to antifaithfulness NONIDENT[low] conditioned on the output context *[-hi].

4.7.2 F → M as output-conditioned antifaithfulness

To see how a constraint F → M can result in output-conditioned antifaithfulness, it helps to convert the expression of the constraint to a more familiar form. This will require several steps. First, recall that the constraint labeled “F → M,” which I will call the composite constraint, is satisfied if it is true that whenever constraint F is satisfied in a local domain, then constraint M is also satisfied in that local domain. This way of expressing it gives the conditions under which the composite is satisfied, expressed in terms of the conditions of satisfaction for F and M. It will be to our advantage to convert this to an expression of the conditions under which the composite is violated, expressed in terms of the conditions of violation of M and F.

I will use Ms and Fs to denote the logical conditions under which the constraints M and F, respectively, are satisfied. I will use Mv and Fv to denote the
logical conditions under which the constraints M and F, respectively, are violated. The relationships are straightforward: \( M_s = \neg M_v \) and \( F_s = \neg F_v \), with the same domains of evaluation. We can thus take the expression of the conditions for satisfaction of the composite, shown in (4.24), and substitute appropriately for the conditions of satisfaction of M and F, and get an expression of the conditions of satisfaction of the composite, expressed in terms of the conditions of violation of M and F, as shown in (4.25). In other words, the composite constraint is satisfied if whenever F is not violated, M is not violated.

\[
\text{(4.24)} \quad \text{Composite is satisfied when: } F_s \rightarrow M_s
\]

\[
\text{(4.25)} \quad \text{Composite satisfied when: } (\neg F_v) \rightarrow (\neg M_v)
\]

Next, we convert the entailment expression to its equivalent disjunctive form. Entailment \( A \rightarrow C \) is logically equivalent to \( (C \text{ or } \neg A) \): the entailment is true whenever the consequent is true or the antecedent is false. Applying that conversion to (4.25) yields (4.26) and can be simplified to (4.27).

\[
\text{(4.26)} \quad \text{Composite satisfied when: } (\neg M_v) \text{ or } (\neg F_v)
\]

\[
\text{(4.27)} \quad \text{Composite satisfied when: } (\neg M_v) \text{ or } (F_v)
\]

For the locus of evaluation, to satisfy the composite constraint, either the markedness constraint must not be violated, or else the faithfulness constraint must be violated.

Finally, we want to convert the expression of the conditions for satisfaction of the composite constraint into an expression of the conditions for violation of the composite constraint. The conditions for violation are simply the negation of the conditions for satisfaction. That expression, and its simplification, are given in (4.28).

\[
\text{(4.28)} \quad \text{Composite violated when: } \neg((\neg M_v) \text{ or } (F_v))
\]

\[
= \neg(\neg M_v) \text{ and } \neg(F_v)
\]

\[
= M_v \text{ and } \neg F_v
\]

The expression in (4.28) is exactly the meaning of the local conjunction of the constraints M and \( \neg F \); in concise notation, \([M \& \neg F]_{\text{dom}}. Given constraints M and F, the semantics of the constraint expression \([M \rightarrow F]_{\text{dom}}\) are the same as the semantics of the constraint expression \([M \& \neg F]_{\text{dom}}. Thus the constraint \( \text{IDENT[low]} \rightarrow *[-hi] \) can be expressed as the locally conjoined constraint \([*[-hi] \& \neg\text{IDENT[low]}], with the local domain being a pair of IO corresponding segments.

It is important to be clear about the interpretation of the expression \( \neg F_v \). It should be interpreted to mean that for any locus of violation of F the logical
conditions for violation of $\neg F_v$ are the opposite of those for $F_v$. The locus of violation for $\text{Ident}[\text{low}]$ is a pair of IO corresponding vowels. Thus, given any pair of IO corresponding vowels, if $\text{Ident}[\text{low}]$ is satisfied then $\neg \text{Ident}[\text{low}]$ is violated, and vice-versa. However, any structure which is not a locus of violation (like an output vowel without an input correspondent) vacuously satisfies both constraints. The expression $\neg \text{Ident}[\text{low}]$ is thus equivalent to the antifaithfulness constraint $\text{NonIdent}[\text{low}]$. Therefore, we can (finally) give the desired form of the material implication constraint $\text{Ident}[\text{low}] \rightarrow \ast[\neg \text{hi}]$ as the locally conjoined constraint in (4.29).

(4.29)  \[ \ast[\neg \text{hi}] \& \text{NonIdent}[\text{low}] \]

This constraint if violated if both conjuncts are violated. Satisfying the constraint requires satisfaction of either $\ast[\neg \text{hi}]$ or $\text{NonIdent}[\text{low}]$. The constraint can be made to behave like $\text{NonIdent}[\text{low}]$ by externally forcing violation of $\ast[\neg \text{hi}]$, which leaves satisfaction of $\text{NonIdent}[\text{low}]$ as the only alternative means of non-vacuously satisfying the constraint.\(^{12}\) The constraint $[\ast[\neg \text{hi}] \& \text{NonIdent}[\text{low}]]$ is an antifaithfulness constraint conditioned on output context, in the same way that the locally conjoined constraints discussed in Section 4.4 are faithfulness constraints conditioned on output context. This constraint requires that an output segment disagree with its input correspondent on the feature low when the output segment has the feature value $\neg \text{hi}$. The non-ODP behavior of distinction conflict exhibited by the material implication constraint in (4.23) is a consequence of the antifaithfulness hidden within the constraint’s logical structure.

### 4.8 Reference to other forms: sympathy

Some proposed OT constraints make reference to forms apart from the input or the output. Sympathy constraints (McCarthy 1999) are an example. I will give only a very brief description here. A sympathy constraint, when evaluating a candidate, makes reference to an independent candidate, the $\bigstar$ (flower) candidate. The $\bigstar$ candidate is the most harmonic candidate that fully satisfies some designated faithfulness constraint. The sympathy constraint then evaluates all candidates with respect to a correspondence relation between the outputs of those candidates and the output of the $\bigstar$ candidate, typically evaluating a

\(^{12}\) The vacuous alternative is to eliminate the locus of violation, as in a candidate in which the input and output vowels in question aren’t IO correspondents. In that case, $\text{Ident}[\text{low}]$, $\text{NonIdent}[\text{low}]$, and $\text{Ident}[\text{low}] \rightarrow \ast[\neg \text{hi}]$ are all vacuously satisfied.
4.8 Reference to other forms: sympathy

4.8 Reference to other forms: sympathy

different dimension of correspondence than the one used to determine the ◆ candidate.

Sympathy constraints pose challenges for analysis in terms of output drivenness, because they are **non-stationary**: their evaluation of a candidate depends not only on the candidate, but on an assumed ranking of the constraints.\(^\text{13}\) Changing the ranking can change the number of violations a sympathy constraint assesses to a candidate. This is due to the fact that the ◆ candidate referred to by a sympathy constraint is determined with reference to harmony (which is defined by a full ranking); changing the ranking can change the ◆ candidate. Thus, the criteria for ODP cannot be applied to the constraint in isolation.

However, one can observe a sympathy constraint in a particular context, with respect to an assumed ranking. If the sympathy constraint plays a key role in determining a non-output-driven mapping, it is because it exhibits a non-ODP behavior in that context. This is illustrated here using the Tiberian Hebrew example from McCarthy (1999). The two key mappings for the example are given in (4.30) and (4.31).

\begin{align*}
(4.30) & /dešʔ/ \rightarrow [deše] \\
(4.31) & /deš/ \rightarrow [deš]
\end{align*}

This is a non-output-driven situation, because /deš/[deše] has greater internal similarity than the grammatical /dešʔ/[deše], but is not itself grammatical. Simplifying the analysis to just the relevant elements yields the tableaux in (4.32).

\begin{align*}
(4.32) & \text{The sympathy constraint exhibits distinction only at lesser similarity}
\end{align*}

\[
\begin{array}{|c|c|c|c|}
\hline
\text{V} & \text{MAX-V} & \text{CODA-COND} & \text{DEP-V} \\
\hline
akx & /dešʔ/[deše] &  & * \\
\hline
aoy & /dešʔ/[deš] & *! &  \\
\hline
\text{◆} & /dešʔ/[dešeʔ] & *! & * \\
\hline
bmx & /deš/[deše] &  & *! \\
\hline
\text{◆} & /deš/[deš] &  &  \\
\hline
\end{array}
\]

\[
in_a = /dešʔ/ \quad in_b = /deš/ \quad out_x = [deše] \quad out_y = [deš] \\
akx = /dešʔ/[deše] \quad bmx = /deš/[deše] \quad aoy = /dešʔ/[deš] \quad bpy = /deš/[deš] \\
bmx \text{ has greater internal similarity than } akx, \text{ yet it is not grammatical.}
\]

\(^{13}\) Other non-stationary constraints pose the same challenges of analysis, including output–output faithfulness constraints (Benua 1997) and targeted constraints (Wilson 2001).
The shaded cells indicate the non-ODP behavior: the sympathy constraint \( \text{Max-V} \) \( \text{Max-C} \) is exhibiting distinction only at lesser similarity, giving rise to the non-output-driven map. The lesser similarity input, /deşʔ/, differs by having the final glottal stop. The \( \text{Max-V} \) \( \text{Max-C} \) candidate is the one which best satisfies \( \text{Max-C} \), so that glottal stop will not be deleted in the \( \text{Max-C} \) candidate; instead, a vowel is epenthesized.\(^{14}\) The sympathy constraint evaluates a correspondence between each of the candidates and the \( \text{Max-C} \) candidate with respect to \( \text{Max-V} \) (requiring each vowel of the output of the \( \text{Max-C} \) candidate to have a correspondent in the output of the candidate being evaluated). Crucially, the sympathy constraint can be satisfied without preserving the glottal stop that is necessarily preserved in the \( \text{Max-C} \) candidate, so the glottal stop does not appear in the output of the optimal candidate, [deše].

The greater similarity input, /deš/, does not have the glottal stop, therefore \( \text{Max-C} \) can be fully satisfied without any vowel epenthesis in the \( \text{Max-C} \) candidate. The two candidates shown for /deš/ are not distinguished by the sympathy constraint, because the \( \text{Max-C} \) candidate has one vowel, with a correspondent in the outputs of each of the two candidates. The greater similarity is due to the lack of the glottal stop, the lack of the glottal stop results in the lack of a second vowel in the \( \text{Max-C} \) candidate, and the lack of a second vowel in the \( \text{Max-C} \) candidate results in a failure of the sympathy constraint to distinguish the two candidates.

4.9 Eventual idempotency

Moreton (2004) demonstrated that there is a significant class of OT grammars, the classical OT grammars, that are eventually idempotent: if you repeatedly feed the output back into the grammar as input, after some finite number of steps you will reach a form that is mapped to itself by the grammar. Thus, classical OT grammars are incapable of realizing circular chain shifts and infinite chain shifts. The result, labeled the characterization theorem, is impressive given the relatively modest conditions that are imposed on constraints. The result relates directly to claims that purely phonological circular and infinite chain shifts are empirically unattested; see Moreton (2004) and Section 4.9.3 below for further discussion of such claims.

Output-driven maps are fully idempotent, not permitting chain shifts of any kind, while eventually idempotent maps include maps with finite chain shifts:

\(^{14}\) There is, of course, more than one candidate minimally violating \( \text{Max-C} \). The choice among them is made on the basis of overall harmony, with respect to the full ranking.
the class of eventually idempotent maps includes many maps that are not output-driven. Nevertheless, examining the characterization theorem in light of developments in output-driven map theory is illuminating, in at least two ways.

- A more general condition on grammars than that used in the characterization theorem can be identified which includes some grammars that are not classical OT grammars but are nevertheless necessarily eventually idempotent.
- A map pattern, which I will call a derived environment exchange, exists which is eventually idempotent, but has the same intuitive property that is empirically questionable in circular chain shifts. Such maps can be realized by classical OT grammars, but they are not output-driven.

4.9.1 Classical OT grammars are eventually idempotent

In Moreton’s analysis, a constraint-hierarchy grammar is a 3-tuple $G = (A \times B, \text{Gen}, C)$ with the properties in (4.33), where $A$ is the set of inputs, $B$ is the set of outputs, the candidates are elements of $A \times B$, and $2^B$ is the power set of $B$ (the set of all subsets of $B$).

\begin{equation}
(4.33) \quad \text{Properties of a constraint-hierarchy grammar}
\end{equation}

(i) $A \times B$ is a countable set
(ii) $\text{Gen} : A \to 2^B$ is such that $\forall a \in A, \text{Gen}(a) \neq \{\}$
(iii) $C$ is a finite constraint hierarchy over $A \times B$.

Condition (ii) asserts that $\text{Gen}$ maps each input to a non-empty subset of the set of outputs $B$. Condition (iii) assumes that a finite constraint hierarchy is a total ordering of a finite set of constraints, each of which assesses zero or more violations to each candidate. It is also assumed that the grammar follows a standard OT architecture: for each input $a \in A$, the candidates are precisely $\text{Gen}(a)$, and the set of grammatical outputs for /a/ are defined by optimization on the lexicographic ordering of the candidates with respect to the constraint hierarchy.

For purposes of analysis, Moreton imposes some conditions on constraint-hierarchy grammars, shown in (4.34), to define the class of classical OT grammars.

\begin{equation}
(4.34) \quad \text{Defining Properties of a classical OT grammar}
\end{equation}

- Exact: every input has exactly one optimal output.
- Homogeneous: Inputs and outputs are the same types of representations (any representation that is an input can be an output, and vice-versa).
Central to this analysis are the definitions of the constraint types, shown in (4.35). Conservativity requires that each constraint be of one of these two types. It excludes constraints that refer to the input but do not always assign zero violations to identity candidates (candidates where the input is identical to the output).

(4.35) Conservative constraints

- Markedness constraint: only evaluates the output of a candidate.
- Faithfulness constraint: assigns no violations to any candidate where the input is identical to the output.

The defining properties are suitable for reasoning about chain shifts: they focus on grammars in which outputs can be identical to inputs, and the (non-)identity of outputs with their inputs is what chain shifts are all about. To satisfy conservativity, every input-referring constraint must assign zero violations to every identity candidate. In Moreton’s analysis, candidates are defined as input–output pairs, devoid of any explicit IO correspondence relation, and an identity candidate is one in which the input and output are identical. Extending this analysis to include explicit recognition of IO correspondence would require that an identity candidate be defined as one with zero disparities: a candidate in which input and output are identical and in appropriate (bijective) IO correspondence. Conservativity would then require that faithfulness constraints assign no violations to identity candidates so defined, but would not require that they assign no violations to candidates with disparities (even if the input and output are identical).

Once we have as premises that a grammar $G$ is exact and homogeneous, it becomes convenient to describe $G$ as a unary operation on a set $S$, that is, a function $G:S \to S$ (rather than as a function from $A$ to $2^B$). The use of the same representations as both inputs and outputs reflects homogeneity, and the output of the function being a single candidate, rather than a non-empty set of candidates, reflects exactness.

Input–output identity is sufficient to describe idempotence. A function $f:S \to S$ is **idempotent** if, for all $s$ in $S$, $f(f(s)) = f(s)$. Note that it is not required that for all $s$ in $S$, $s = f(s)$. What is required is that for any member $f(s)$ of $S$ that is the output for some element $s$, $f(f(s)) = f(s)$. It is useful to use
an exponent to represent multiple iterations of a function: \( f^2(s) \) means \( f(f(s)) \), \( f^3(s) \) means \( f(f(f(s))) \) which is the same as \( f(f^2(s)) \), and so forth.

Classical OT grammars that are not idempotent certainly exist. But there is a weaker property which does hold of all classical OT grammars. A function \( f : S \rightarrow S \) is **eventually idempotent** if, for any \( s \) in \( S \), there is a finite number \( p \) such that \( f^p(s) = f^{p+1}(s) \). This result is stated in (4.36).

(4.36) The characterization theorem (Moreton 2004).
Let \( f : S \rightarrow S \) be any function. Then \( f \) is computable by a classical OT grammar if and only if \( f \) is eventually idempotent.

What kinds of unary functions aren’t eventually idempotent? Circular chain shifts and infinite chain shifts. A grammar \( G \) isn’t eventually idempotent if there is an \( s \) such that the sequence of iterations of the grammar on \( s \) never reaches a value which maps to itself. If that sequence of iterations ever repeats a value, then it is an instance of a circular chain shift; otherwise, it is an infinite chain shift.

To summarize the proof in Moreton 2004 of why any classical OT grammar \( G : S \rightarrow S \) must be eventually idempotent, consider some identity candidate \( /s/ [s] \). The part of the constraint hierarchy consisting of faithfulness constraints, \( C^F \) must assess no violations to \( /s/[s] \), by the definition of conservative faithfulness constraint in (4.35). Therefore, none of the faithfulness constraints can prefer any other candidate over \( /s/[s] \). If \( G(s) \neq s \), then the highest-ranked constraint with a preference must prefer \( /s/[G(s)] \) over \( /s/[s] \). Since the highest-ranked constraint with a preference cannot be a faithfulness constraint, it must be a markedness constraint. Therefore, the part of the constraint hierarchy consisting of markedness constraints, \( C^M \), must prefer \( /s/[G(s)] \) to \( /s/[s] \). Because markedness constraints only evaluate outputs, we could just as easily say that \( C^M \) prefers \( G(s) \) to \( s \). That is, the violation profile assessed by \( C^M \) to \( G(s) \) must be “below” the violation profile assessed by \( C^M \) to \( s \) with respect to the usual Optimality Theoretic interpretation of the constraint hierarchy \( C^M \). For a classical OT grammar, \( s \neq G(s) \) entails that output \( G(s) \) must be less marked than \( s \): \( C^M[s] > C^M[G(s)] \). For any \( s \) in \( S \), either \( G(s) = s \), satisfying eventual idempotency, or \( C^M[s] > C^M[G(s)] \).

Because the markedness reduction observation applies to any input in \( S \), it applies repeatedly over iterations of the grammar so long as the output is not identical to the input: \( C^M[s] > C^M[G(s)] > C^M[G^2(s)] \), so long as

---

15 The usual duality applies here. If \( G(s) \) is “less marked than” \( s \) with respect to hierarchy \( C^M \), then \( G(s) \) is “more harmonic than” \( s \) with respect to \( C^M \).
s \neq G^1(s) \neq G^2(s)$. If $G^n(s) \neq G^{n+1}(s)$, then $C^M[G^n(s)] > C^M[G^{n+1}(s)]$. The markedness of successive iterations has to keep going down until a fixed point is reached. The final step is to observe that a descending sequence of markedness violation profiles has to come to an end after a finite number of steps; it cannot keep going down forever.\(^\text{16}\) Since the sequence cannot go back up to a higher markedness profile, it must, after a finite number of iterations, reach a fixed point where further iterations return the same markedness profile. Because at that point $C^M[G^{p+1}(s)]$ is not less than $C^M[G^p(s)]$, it follows that $G^{p+1}(s) = G^p(s)$. Thus, the grammar must be eventually idempotent.

4.9.2 Absolute vs. relative satisfaction of constraints

By definition, a grammar is conservative if all non-markedness constraints always assign zero violations to identity candidates. No other restrictions are imposed regarding constraints. This is notable for the absolute nature of the satisfaction required. An identity candidate must receive exactly zero violations across all faithfulness constraints. The role played by this restriction in the characterization theorem is to ensure that the faithfulness constraints as a whole never prefer a competitor over an identity candidate. Clearly, an identity candidate cannot lose on faithfulness if it incurs no violations of faithfulness constraints. But an absolute condition, zero violations assessed to identity candidates by all faithfulness constraints, is being used to fill a fundamentally relative role, ensuring identity candidates never lose on faithfulness to competitors.

A more general condition on faithfulness is obtained by following the relative nature of the role: $C^F$ must not prefer a competitor over an identity candidate. Said differently, an identity candidate must incur minimal violation of the faithfulness constraints relative to its competitors (the other candidates for the same input). So long as the faithfulness constraints don’t prefer any competitor over the identity candidate, it must be the case that the markedness constraints do prefer some competitor if the identity candidate is to be suboptimal. A simpler, but slightly less general, condition would be to make minimality a requirement of each faithfulness constraint individually: each faithfulness constraint must assess minimal violation to an identity candidate relative to its competitors.

\(^\text{16}\) This is due to the fact that the natural numbers (the ordered set of possible numbers of violations assessed by a single constraint) are a well-ordered set, and any lexicographic order constructed from well-ordered sets is itself a well-ordered set.
One simple, if uninteresting, example would be to take a classical OT grammar and change the definitions of the faithfulness constraints by adding 1 to the number of violations assessed to every candidate. The identity candidates, which previously were assessed zero violations by each faithfulness constraint, would now each be assessed one violation by each faithfulness constraint. Candidates which were previously assessed 1 violation by a given constraint would now be assessed 2 violations, and so forth. This new grammar is not a classical OT grammar, because it is not conservative: identity candidates are assessed violations by faithfulness constraints. But the grammar defines the same language as before, because the relative degree of violation of each constraint is still the same for any pair of competitors. What the classical OT grammars and their augmented counterparts have in common is that identity candidates always have minimal violation of the faithfulness constraints with respect to their competitors.

Requiring that identity candidates have minimal violation of faithfulness is a relative, rather than absolute, characterization. But Optimality Theory is defined in terms of relative harmony among competitors. The conditions for a constraint to be ODP, (3.39) and (3.40), are purely relative in terms of the number of violations. That is not by accident: the conditions were derived directly from the conditions for optimality in the theory (see Section 3.2.5). Respecting the relative nature of optimization in Optimality Theory makes it possible to identify more general classes of grammars that exhibit particular behaviors. In the case of eventual idempotency, a more general class of eventually idempotent grammars can be identified if the absolute condition of conservatism is replaced with the relative condition requiring minimal violation of faithfulness for identity candidates with respect to their competitors.

4.9.3 Derived environment exchanges

A process in which two distinct segment types are each replaced with the other, in the same environment, can be called a segmental exchange. In such a case, successive iterations of the grammar cause one output position to alternate back and forth between two different segment types; each exchanged for the other, as it were. In rule-based phonology, a rule which performs such an exchange, changing A in the input into B, and changing B in the input into A, has been called an exchange rule. The possibility of exchange rules has long been of interest in phonology. Chomsky and Halle defended the use of exchange rules in their analysis of English (Chomsky and Halle 1968: 256–259). Anderson and Browne 1973 argued that exchange rules are never purely phonological, but always conditioned in some way on the morphological environment.
A map that, for a pair of forms, exhibits an exchange with no other disparities, embodies a circular chain shift and therefore cannot be realized by a classical OT grammar. But the situation can change if an exchange is wedded to other disparities introduced by the grammar. An example of such a pattern comes from an analysis by Zonneveld 1976 of data from Flemish Brussels Dutch. In this pattern, long vowels shorten in certain environments (e.g., __ C₁C₂), and when the back vowels shorten they also reverse their value of the feature hi (–hi → +hi, +hi → –hi). The assumed pattern, as a map, is given in (4.37). This map gives only the vowels themselves, leaving out the conditioning environment(s) for the shortening, to keep the presentation simple.

(4.37) The Flemish Brussels Dutch vowel shortening map

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>/o:/</td>
<td>[u]</td>
</tr>
<tr>
<td>/u:/</td>
<td>[o]</td>
</tr>
<tr>
<td>/o/</td>
<td>[o]</td>
</tr>
<tr>
<td>/u/</td>
<td>[u]</td>
</tr>
</tbody>
</table>

Note that there is no circular chain shift here, as there is no mapping that lengthens a short vowel; once a vowel shortens, there is no going back (in the relevant environments). But the switch in the values of the feature hi is similar in spirit to an exchange. It is an exchange melded with a derived environment effect. The reversal of the feature hi (the exchange part) only occurs where vowel shortening is taking place (the derived environment part).

The pattern is of interest here because, despite the exchange-like nature, it can be realized by a classical OT grammar, but is not output-driven. One such classical OT grammar is given here. It depends upon the constraint in (4.38).

(4.38) [Id[long] & NonId[hi]]+back: for IO corresponding vowels where the output vowel is +back, this constraint is violated when they mismatch on the feature long and match on the feature hi.

This constraint is stated in the form of a locally conjoined constraint. The abstract structure is the same as the material implication constraints discussed in Section 4.7.1. While the constraints discussed in Section 4.7.1 originated with the form F → M, the constraint in (4.38) is equivalent to one of the form F → F: the constraint could have been labeled [Id[hi] → Id[long]]+back. If violation of Id[long] is unavoidable, then satisfaction of NonId[hi] is required in order to satisfy [Id[long] & NonId[hi]]+back.

The grammar has three other constraints. *[+long], the lone markedness constraint, is violated by any output vowel that is +long. Id[hi] and Id[long], as expected, require IO corresponding vowels to match on the feature hi and the feature long, respectively. It is assumed that a constraint such as Id[back]
is sufficiently dominant to prevent that feature’s value from being changed for these inputs; that constraint is not included in the rankings and tableaux here. One successful ranking of the constraints is given in (4.39), and the analysis of the map is given in (4.40). More than one ranking of these constraints will work; the crucial rankings are \( *[+\text{long}] \gg \text{ID}[\text{hi}], *[+\text{long}] \gg \text{ID}[\text{long}], \text{[ID}[\text{long}] \& \text{NONID}[\text{hi}]+\text{back} \gg \text{ID}[\text{hi}] \).

(4.39) \[ *[+\text{long}] \gg \text{ID}[\text{long}] \& \text{NONID}[\text{hi}]+\text{back} \gg \text{ID}[\text{hi}] \gg \text{ID}[\text{long}] \]

(4.40) A classical OT grammar realizing the Flemish Brussels Dutch pattern

<table>
<thead>
<tr>
<th></th>
<th>([+\text{long}])</th>
<th>\text{[ID}[\text{long}] &amp; \text{NONID}[\text{hi}]+\text{back} )</th>
<th>\text{ID}[\text{hi}]</th>
<th>\text{ID}[\text{long}]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{aoy} )</td>
<td>/o:/[o]</td>
<td>*!</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>(\text{akx} )</td>
<td>/o:/[u]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\text{bpy} )</td>
<td>/u:/[o]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(\text{bmx} )</td>
<td>/u:/[u]</td>
<td>*!</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\( \text{in}_a = /o:/ \quad \text{in}_b = /u:/ \quad \text{out}_a = [u] \quad \text{out}_b = [o] \)

\( \text{akx} = /o:/[u] \quad \text{bmx} = /u:/[u] \quad \text{bpy} = /u:/[o] \quad \text{aoy} = /o:/[o] \)

The markedness constraint \(*[+\text{long}]\) bans long vowels on the surface (in the relevant environment) and ensures that underlyingly long vowels will shorten. It also ensures that the first conjunct of \([\text{ID}[\text{long}] \& \text{NONID}[\text{hi}]+\text{back}, \text{ID}[\text{long}],\) is

\( \text{17} \) The full implications of the choice of locus of violation (IO corresponding vowels with output +back, with input +back, or with both input and output +back) are left unexplored for the present.
violated, requiring that the second conjunct, $\text{NonId}[\text{hi}]$, be satisfied if violation of $[\text{Id}[\text{long}] \& \text{NonId}[\text{hi}]]_{+\text{back}}$ is to be avoided. It is this forced satisfaction of $\text{NonId}[\text{hi}]$ that results in the polarity reversal of hi for underlyingly long vowels.

For the underlyingly short vowels, markedness rules out lengthening, and for the candidates with short output vowels, the lack of change in the feature long ensures satisfaction of $[\text{Id}[\text{long}] \& \text{NonId}[\text{hi}]]_{+\text{back}}$, in virtue of satisfaction of $\text{Id}[\text{long}]$. $\text{Id}[\text{hi}]$ then prevents any change in the feature hi.

The grammar just described satisfies all of Moreton’s conditions for a classical OT grammar. It is a constraint hierarchy grammar, with $S = \{o, u, o:, u:\}$, and $\text{Gen}$ and $\text{Con}$ as indicated (exhaustively) in (4.40). It is homogeneous, as $S$ is both the set of possible inputs and the set of possible outputs. $\text{Gen}$ is inclusive, and in fact every member of $S$ is both an input and a candidate output for every input. The grammar is exact: each input has exactly one optimal candidate. Most significantly, the grammar is conservative. One of the constraints, $[^{+}\text{long}]$, is a markedness constraint. The other three constraints are faithfulness constraints: in each of the four competitions in (4.40), the identity candidate for that competition incurs zero violations of all three constraints.

It is particularly noteworthy that $[\text{Id}[\text{long}] \& \text{NonId}[\text{hi}]]_{+\text{back}}$, the constraint with the embedded antifaithfulness conjunct, is conservative. That is because, unlike $[^{-}\text{hi}] \& \text{NonIdent}[\text{low}]$, a local conjunction of antifaithfulness with a markedness constraint, $[\text{Id}[\text{long}] \& \text{NonId}[\text{hi}]]_{+\text{back}}$ is a local conjunction of antifaithfulness with a faithfulness constraint. Satisfying the conjoined constraint only requires satisfying the antifaithfulness conjunct if the faithfulness conjunct is violated, which entails the presence of a disparity (here, an identity disparity on the feature long). The constraint can only be violated when there is a disparity violating the faithfulness conjunct; it cannot be violated by an identity candidate.

The map itself is not output-driven. This can be shown by considering the candidates /o://[u] and /u:/[u]. Note first that /u:/[u] has greater internal similarity than /o:/[u] with respect to a relative similarity relation based on identity disparities: /u:/[u] has a disparity in the feature long, while /o:/[u] has the same disparity in long and an additional disparity in the feature hi. Yet /o:/[u] is part of the map (it is grammatical), while /u:/[u] is not, violating the fundamental requirement of output drivenness: the grammaticality of one mapping entails the grammaticality of any mapping of greater internal similarity.

In the OT grammar generating the map, the responsible non-ODP constraint is $[\text{Id}[\text{long}] \& \text{NonId}[\text{hi}]]_{+\text{back}}$, and the kind of non-ODP behavior it exhibits in this case is distinction conflict, as highlighted by the shaded cells in (4.40): the
constraint assesses fewer violations to /u:/[o] than to /u:/[u] \((C(bpy) < C(bmx))\) but assesses more violations to /o:/[o] than to /o:/[u] \((C(aoy) > C(akx))\). The conditions defining distinction conflict rely on relative similarity and relative numbers of violations. They are blind to the fact that, in this case, all four candidates contain another disparity (in length) that results in a violation of another faithfulness constraint \((\mathrm{Id}[\mathrm{long}])\). They scrutinize the assessment by the constraint of candidates in relative similarity relations, regardless of how close to or far away from identity candidates those candidates are (that is, regardless of how many other disparities they contain). Conservative grammars only require that faithfulness constraints not assess any violations to identity candidates and therefore cannot as a class require anything interesting regarding patterns in maps not involving identity candidates. The restrictive reach of classical OT grammars is limited to things like eventual idempotence.

### 4.9.4 Absolute vs. relative characterizations of similarity

The analysis of classical OT grammars works in part because it implicitly assumes a particular form of internal similarity. The analysis of classical OT grammars does not make any particular reference to the comparison of two candidates sharing the same output, but it does make particular reference to the comparison of two candidates sharing the same input. The comparison is between a candidate that has identical input and output and a competing candidate which does not. If the input and output of a candidate are identical, then they are as similar as input and output can be, and therefore an identity candidate has greater internal similarity than any candidate it is compared to. In this form, internal similarity is in a sense binary: either a candidate is an identity candidate, or it is not. Identity candidates have greater internal similarity than non-identity candidates.

This binary internal similarity can be understood as the coarsest level of disparity analysis possible. A disparity in this analysis consists of an entire input and a non-identical entire output. The candidate /bapaka[/afnobis] would have exactly one disparity, the disparity bapaka:afnobis, with no internal decomposition. Each such input/output pair is a distinct disparity. Two distinct candidates, each with an output not identical to its input, necessarily are not related to each other in terms of similarity: each has a (single) disparity that the other lacks. Candidates not identical to each other can only be related in terms of similarity

18 The analysis is thus an instance in which distinction conflict is responsible for a non-output-driven pattern other than a circular chain shift.
if one has no disparities and the other has one disparity. This notion of similarity is coarse enough that it can be used to compare any pair of candidates, including candidates that have different inputs but identical outputs, as with the relative similarity relation relevant to output drivenness.

The heart of the characterization theorem is the observation that, in classical OT grammars, faithfulness constraints cannot prefer a non-identity competitor over an identity candidate. That is what forces a winning non-identity candidate to be less marked than its identity candidate competitor. Since an identity candidate, by definition, has no disparities, it is trivially true that every disparity of an identity candidate has a corresponding disparity in any other candidate. Identity candidates will always have greater internal similarity than non-identity competitors, and faithfulness constraints never disprefer identity candidates. If the only substantive distinction in similarity is between an identity candidate and a non-identity candidate, then faithfulness constraints will never prefer candidates of lesser internal similarity (necessarily non-identity) to candidates of greater internal similarity (necessarily identity).

In this way, the relative similarity implicit in the analysis has a kind of “absolute” character to it. Although, as the name implies, relative similarity is at heart a relative notion, the characterization used here is absolute: either a candidate’s input and output are identical, or they are not. More precisely, either a candidate has zero disparities, or it does not. It is analogous to the “absolute” characterization of faithfulness constraints discussed in Section 4.9.2: a faithfulness constraint must assign zero violations to an identity candidate. In both instances, an absolute characterization (zero violations; zero disparities) is used to ensure a relative outcome (less than or equal to the number of violations; every disparity has a correspondent).

Basing relative similarity solely on identity/non-identity allows the characterization theorem to apply rather generally, to a wide variety of OT analyses, because the assumptions it makes about OT analyses are rather spare. The trade-off for that generality is a strong limit on what the theorem can say about the OT analyses it applies to. The characterization theorem is limited to properties involving identity candidates, because identity vs. non-identity is the only kind of distinction in similarity that is assumed. That is enough for circular and infinite chain shifts, which are characterized in terms of (non)identity of forms. But it is inadequate to address phenomena like derived environment exchanges, which involve distinguishing the relative similarity of candidates neither of which is an identity candidate. Addressing such cases appears to require moving away from the purely “absolute” characterization of relative similarity assumed by the characterization theorem.
4.10 The role of relative similarity

Moreton discusses the Flemish Brussels Dutch pattern as shown in (4.37), and claims that this pattern cannot be realized by a classical OT grammar (Moreton 2004: 156). Interpreting this as an empirical challenge to the conditions defining classical OT grammars, Moreton goes on to question the veracity of Zonneveld’s description of the data, citing Loey 1933 and Mazereel 1931 as relevant sources. My concern here is not the exact empirical facts of Flemish Brussels Dutch, but the claim that the Flemish Brussels Dutch pattern, as described in (4.37), cannot be realized by a classical OT grammar. This claim is false, given the existence of the grammar in (4.40), which realizes the pattern and is a classical OT grammar. In this section, I argue that such a misunderstanding can result from a failure to properly appreciate the role of relative similarity in relating properties of maps to properties of grammars.

4.10.1 Distinguishing faithfulness from faithfulness constraints

The claim that the Flemish Brussels Dutch pattern cannot be realized by a classical OT grammar is based on a statement labeled the rotation theorem, given in (4.41).

(4.41) The rotation theorem (Moreton 2004). Let S be a countable set, and let \( N : S \to 2^S \) be exact. Suppose \( A \) and \( A' \) are non-empty subsets of \( S \), and suppose there exist functions \( f, g : A \to A' \) such that

(i) \( f \) and \( g \) are bijections from \( A \) to \( A' \)

(ii) \( \forall a \in A, f(a) \neq g(a) \)

(iii) \( \forall a \in A, f(a) = N(a) \)

(iv) \( \forall a \in A, C^F/a[g(a)] \subseteq C^F/a[f(a)] \)

Then there does not exist any classical OT grammar \( G = (S \times S, C) \) such that \( G(a) = N(a) \) for all \( a \in S \).

The symbol ‘\( \preceq \)’ in (4.41)(iv) is a bit tricky; it indicates that the first argument, \( C^F/a[g(a)] \), is more harmonic than the second argument, \( C^F/a[f(a)] \). To be consistent with the rest of the book (and common practice), I will rewrite condition (iv) as in (4.42).

(4.42) (iv) \( \forall a \in A, C^F/a[g(a)] \preceq C^F/a[f(a)] \)

Moreton asserts that the Flemish Brussels Dutch vowel shortening map is “a perfect configuration for the rotation theorem,” and concludes that “Flemish Brussels Dutch cannot be modeled by a classical OT grammar” (Moreton 2004: 156). Yet such a grammar was given in (4.40). What went wrong?

A careful examination of the statement of the rotation theorem reveals a problem. The notation \( C^F/a[g(a)] \), as it appears in condition (iv), is defined...
as the ordered (by ranking) violation vector for the faithfulness constraints of some grammar. But there is no indication prior to condition (iv) what grammar that might be in reference to. As stated, the $C_F$ in condition (iv) is an unbound variable or an undefined constant, and the theorem doesn’t really have any clear content.

The textual presentation of the rotation theorem suggests that conditions (i)–(iv) characterize the function $N$, such that $N$ cannot be realized by any classical OT grammar. But condition (iv) explicitly invokes faithfulness constraints, presumably to say something about how $N$ relates to faithfulness. These conditions defining $N$ are then used to evaluate the capacity of the classical OT grammars, whose most notable characteristic is their conservative faithfulness constraints. It appears to be an attempt to use faithfulness constraints to characterize faithfulness in a map, and then to use that characterization of faithfulness to evaluate faithfulness constraints.

Evaluating faithfulness constraints using a measure that is defined in terms of those very same constraints invites trouble. If faithfulness is defined directly in terms of faithfulness constraints, then one cannot evaluate claims about one candidate being more faithful than another without knowing what the faithfulness constraints are (and how they are ranked). If one goes ahead and commits to what those faithfulness constraints are (and how they are ranked), then faithfulness is at the mercy of those faithfulness constraints; one’s intuitions about faithfulness have been rendered irrelevant by definition. In the grammar in (4.40), it is literally the case that $C_F/o:/u] > C_F/o:/o]$, and if $C_F$ is taken as the definition of faithfulness, then $/o:/u]$ is more faithful than $/o:/o]$.

If one’s goal is to evaluate what classical OT grammars can or can’t do with respect to faithfulness, what is needed is a characterization of faithfulness that is independent of any constraints at all (faithfulness or otherwise). Faithfulness constraints, or more generally input-referring constraints, can then be evaluated by how they do (or don’t) accord with the independent characterization of faithfulness.

4.10.2 The rotation theorem isn’t about faithfulness

Moreton’s discussion preceding the rotation theorem suggests that what he intended is that there should be an intuitive sense of which input/output pairs are “more faithful” than others, and that the faithfulness constraints should somehow reflect that intuitive sense. Moreton’s statements about Flemish Brussels Dutch also suggest this: “For any phonotactically permissible $X$ and $Y$, we expect $[XuY]$ to be more faithful to $/Xu:Y/$ than $[XoY]$ is, and we expect $[XoY]$ to be more faithful to $/Xo:Y/$ than $[XuY]$ is” (Moreton 2004: 156). No explicit definition of faithfulness that supports those expectations is given.
Along these lines, one could attempt to reinterpret the rotation theorem by having \( C^F \) bound by the variable \( G \), and altering the statement following condition (iv) to be something like that in (4.43).

(4.43) Let \( N \) be such that \( A, A', f, \) and \( g \) exist that collectively satisfy conditions (i)–(iii). Then no classical OT grammar \( G \) exists which simultaneously realizes \( N \) and satisfies condition (iv).

The idea would be that conditions (i)–(iv) collectively define one sense of a failure to be faithful in the “expected” way, with the desired result that a function satisfying (i)–(iv), and thus failing to be faithful in the expected way, cannot be realized by a classical OT grammar. The argument that the Flemish Brussels Dutch pattern cannot be realized by a classical OT grammar would then follow from demonstrating that any grammar realizing the pattern satisfies conditions (i)–(iv). This line of reasoning fails, however, because there exists a grammar realizing the pattern, namely the one in (4.40), that does not satisfy conditions (i)–(iv).

Let the function \( N \) be the Flemish Brussels Dutch pattern: \( N(\alpha) = u, N(\mu) = o \). Let \( A = \{\alpha, \mu\} \) and \( A' = \{u, o\} \). Let \( f(\alpha) = N(\alpha) = u, \) and \( f(\mu) = N(\mu) = o, \) so that \( f \) is a bijection from \( A \) to \( A' \), and \( f(a) = N(a) \) for all \( a \) in \( A \). Let \( g(\alpha) = o, \) and \( g(\mu) = u, \) so that \( g \) is a bijection from \( A \) to \( A' \), and \( f(a) \neq g(a) \) for all \( a \) in \( A \). Thus, \( f \) and \( g \) satisfy conditions (i)–(iii) of the rotation theorem.

The grammar in (4.40) does not satisfy condition (iv). The highest-ranked faithfulness constraint in the grammar, the conjoined constraint \([ID[long] & NONID[hi]] + back\), is violated by \( C^F/\alpha:/g(\alpha) \) = \( C^F/\alpha:/o \) and satisfied by \( C^F/\alpha:/f(\alpha) \) = \( C^F/\alpha:/u \), so it is not the case that \( C^F/a/[g(a)] \geq C^F/a/[f(a)] \) when \( a = \alpha \). Similarly, the conjoined constraint is violated by \( C^F/\mu:/g(\mu) \) = \( C^F/\mu:/u \) and satisfied by \( C^F/\mu:/f(\mu) \) = \( C^F/\mu:/o \), so it is not the case that \( C^F/a/[g(a)] \geq C^F/a/[f(a)] \) when \( a = \mu \).

The failure of the grammar in (4.40) to satisfy condition (iv) means that the claim in (4.43) does not entail anything about the grammar in (4.40). Because \( A \) contains only two elements, the form of function \( g \) is forced by the form of function \( f \) (which is determined by \( N \)).

Not surprisingly, it is also the case that grammars clearly meeting the intuitive sense of expected faithfulness behavior do not satisfy conditions (i)–(iv). The conditions do nothing to distinguish derived environment exchanges (like the Flemish Brussels Dutch map) from ordinary idempotent maps.

The grammar in (4.44) is much like (4.40) but with the constraint \([ID[long] & NONID[hi]] + back\) removed. The result is a very simple map in which long vowels shorten, and that’s it. The map is idempotent and furthermore is output-driven.
(4.44) A simple grammar failing to satisfy condition (iv) of the rotation theorem

<table>
<thead>
<tr>
<th></th>
<th>*{+long}</th>
<th>Id[hi]</th>
<th>In[long]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ὁ</td>
<td>/o:ʃ[o]</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>/owl</td>
<td>/o:ʃ[u]</td>
<td>*!</td>
<td>*</td>
</tr>
<tr>
<td>ὁ</td>
<td>/o:ʃ[o:]</td>
<td>*!</td>
<td>*</td>
</tr>
<tr>
<td>/owl</td>
<td>/o:ʃ[u:]</td>
<td>*!</td>
<td>*</td>
</tr>
<tr>
<td>/u:ʃ[o]</td>
<td>*!</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>/owl</td>
<td>/u:ʃ[u]</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>ὁ</td>
<td>/u:ʃ[o]</td>
<td>*!</td>
<td>*</td>
</tr>
<tr>
<td>/owl</td>
<td>/u:ʃ[u]</td>
<td>*!</td>
<td>*</td>
</tr>
<tr>
<td>/u:ʃ[ʃ]</td>
<td>*!</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

As described in (4.4): N(o:) = o, N(u:) = u, N(o) = o, and N(u) = u. Let A = {o, u} and A' = {o, u}. Let f(o:) = N(o:) = o, and f(u:) = N(u:) = u, so that f is a bijection from A to A', and f(a) = N(a) for all a in A. Let g(o:) = u, and g(u:) = o, so that g is a bijection from A to A', and f(a) ≠ g(a) for all a in A. Thus, f and g satisfy conditions (i)–(iii) of the rotation theorem.

Condition (iv), however, is violated for both members of A. For instance, if C^F/ʃ/o:/g(a)\] = C^F/ʃ/o:/u\] incurs a violation of Id[hi], the highest-ranked faithfulness constraint, while C^F/ʃ/o:/f(o:)\] = C^F/o:/ʃ/o\] incurs no violations of Id[hi]. This violates condition (iv), because it is not the case that C^F/a[:g(a)] ≥ C^F/a[:f(a)] when a = o:. Similarly, it is not the case that C^F/a[:g(a)] ≥ C^F/a[:f(a)] when a = u:.

If C^F is interpreted as defining the meaning of being more faithful, then the relationships that violate condition (iv) for (4.44) are precisely what Moreton expects in his discussion quoted above: [u] is more faithful to /u:/ than [o] is, and [o] is more faithful to /o:/ than [u] is. But the conditions that violate condition (iv) for (4.40) are the reverse: [o] is more faithful to /u:/ than [u] is, and [u] is more faithful to /o:/ than [o] is. Clearly, conditions (i)–(iv) do not
4.10 The role of relative similarity

distinguish grammars that generate derived environment exchange maps from grammars that generate expected maps. The grammars in (4.40) and (4.44) both fail to satisfy the conditions of the rotation theorem.

In fact, conditions (i)–(iv) do not even distinguish classical OT grammars from non-conservative grammars if we allow $C^F$ to include all non-markedness constraints. A very simple example is the grammar in (4.45), which realizes a circular chain shift. It does so via the non-conservative constraint $\text{NonId}[\text{long in } \{-\text{long}\}]$, which is conditioned on the input segment of an input–output pair such that if the input correspondent is $-\text{long}$, then the output correspondent must differ from the input in the value of $\text{long}$ (segments without IO correspondents vacuously satisfy the constraint, just as they do with $\text{Ident}$). In other words, if the input correspondent is $-\text{long}$, then the output correspondent must be $+\text{long}$. That this constraint is non-conservative is demonstrated by the fact that the only violation it assesses in (4.45) is to an identity candidate. The grammar is not a classical OT grammar.

(4.45) Circular chain shift grammar that does not satisfy conditions (i)–(iv)

<table>
<thead>
<tr>
<th></th>
<th>$\text{NonId}[\text{long in } {-\text{long}}]$</th>
<th>$*\text{[+long]}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f^r$</td>
<td>$/o:/[o]$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$/o:/[o:]$</td>
<td>$*!$</td>
</tr>
<tr>
<td>$f$</td>
<td>$/o/[o]$</td>
<td>$*!$</td>
</tr>
<tr>
<td>$f^r$</td>
<td>$/o/[o:]$</td>
<td>$*$</td>
</tr>
</tbody>
</table>

With respect to the rotation theorem, let $A = \{o:, o\}$ and $A' = \{o:, o\}$. Let $f(o:) = o$ and $f(o) = o:$. Let $g(o:) = o:$. Functions $f$ and $g$ are both bijections from $A$ to $A'$, and are never equal. Thus, they satisfy conditions (i)–(iii) of the rotation theorem. Condition (iv), however, is violated for input $/o/$. $C^F/o/[g(o)] = C^F/o/[o]$ violates the antifaithfulness constraint, while $C^F/o/[f(o)] = C^F/o/[o:]$ does not, so it is not the case that $C^F/a/[g(a)] \succ C^F/a/[f(a)]$ when $a = o$. This non-conservative grammar fails conditions (i)–(iv) in the same way that the grammars in (4.40) and (4.44) did, and thus the statement of the rotation theorem says nothing about it. The grammar is not a classical OT grammar, but the rotation theorem doesn’t claim that it is not a classical OT grammar (nor that it is).

The rotation theorem doesn’t say anything about maps themselves. As a result, it cannot say anything about derived environment exchanges on their own. The grammar in (4.40) that realizes a derived environment exchange does not contradict the rotation theorem. The conditions of the rotation theorem
do not capture “expected” faithfulness behavior in the way that was intended. Condition (iv) is not satisfied in the straightforward grammar of (4.44), when $C^F/o:/[u]$ is not at least as harmonic as $C^F/o:/[o]$, as intended. But condition (iv) is also not satisfied in the derived environment exchange grammar in (4.40), when $C^E/o:/[o]$ is not at least as harmonic as $C^E/o:/[u]$, clearly not what was intended. What changes between the two cases is $C^F$, the content of the faithfulness constraints. As argued at greater length in the next section, the rotation theorem fails to achieve the intended result because it tries to use the faithfulness constraints of a grammar as a definition of faithfulness itself, rather than providing an independent characterization of what is “expected” of faithfulness. The rotation theorem refers to faithfulness constraints, but it doesn’t say anything about faithfulness.

4.10.3 Relative similarity links grammars to maps

The characterization theorem, that classical OT grammars must be eventually idempotent, succeeds because it relies on an implicit relative similarity relation, one based on identity of disparities defined in terms of entire representations: a pair of identical representations are more similar to each other (0 disparities) than a pair of non-identical representations are to each other (1 disparity). This relation fills the role of relative similarity based on correspondence of disparities in a trivial way. If the input and the output of a candidate are identical, then that candidate has no disparities. It is then trivially true that every disparity of that candidate has a corresponding disparity in whatever other candidate it is compared with. A candidate with no disparities has greater internal similarity than every candidate with the same output; any candidate with one (representation-wide) disparity does not have greater internal similarity than any candidate other than itself.

This similarity relation implicit in the characterization theorem makes no reference to grammars or constraints and can be used to evaluate maps on their own. This similarity relation is sufficient to characterize idempotency: the output for an input is either identical to the input (greater internal similarity) or it is not (lesser internal similarity). It is also the basis for the key condition on classical OT grammars, that of conservativity: a faithfulness constraint must not prefer a non-identity candidate (lesser internal similarity) over an identity candidate (greater internal similarity). Relative similarity is the key link between the properties of the maps and the properties of the grammars.

By contrast, the attempt to reason about the Flemish Brussels Dutch pattern using the rotation theorem fails because there is no relative similarity relation that links the two together. The rotation theorem does not characterize
more on relating non-ODP behaviors to non-ODM patterns

constraints in terms of any relative similarity relation, it simply asserts that there are faithfulness constraints. There is no relative similarity relation in the rotation theorem that links it to maps.

The purpose of a relative similarity relation is to specify when certain mappings are more faithful than others, independent of any constraints. The definition of an output-driven map makes crucial reference to a relative similarity relation, and no reference at all to constraints (or even Optimality Theory). Thus the Flemish Brussels Dutch map can be judged to be non-output-driven with respect to a relative similarity relation based on disparities in feature values of corresponding vowels (which matches the intuitive sense of faithfulness that Moreton almost certainly had in mind). The conditions on constraints given in Section 3.3.2 are stated with respect to relative similarity. It is relative similarity, stated independently, that links output drivenness in maps and constraint behavior in grammars.

The intuitions Moreton expressed about “expected” faithfulness, such as that [u] is expected to be more faithful to /u:/ than [o] is, line up with relative similarity based on identity disparities: /u:/[u] has one disparity, while /u:/[o] has two disparities. In this instance, the failure to lay out an independent conception of relative similarity that expressed those intuitions led to a concrete, demonstrably false conclusion: that the Flemish Brussels Dutch map cannot be generated by a classical OT grammar.

The purpose of this extended examination of the rotation theorem is to argue for the central importance of relative similarity in understanding faithfulness relationships between grammars and maps. Relative similarity relations aren’t just an idiosyncratic aspect of the definition of output-driven maps. A relative similarity relation (explicit or implicit) is the faithfulness relation between representations that is adopted by a given linguistic theory. Any coherent examination of relationships between components of grammars and faithfulness in maps requires a clear understanding of what faithfulness in maps is, namely the relative similarity relation being used. The (mis)analysis of the Flemish Brussels Dutch pattern illustrates the importance of relative similarity, and the perils of ignoring it.

4.11 More on relating non-ODP behaviors to non-ODM patterns

A given type of non-ODP behavior can be responsible for different non-ODM patterns in different circumstances. In both the chain shift analysis of Etxarri Basque (4.5) and the derived environment effect analysis of Polish spirantization (4.18), the key non-ODP constraint exhibits distinction only at lesser
similarity. It is even possible for a circular chain shift to be the result of distinction only at lesser similarity, as with the value-restricted \textsc{NonId} constraint in (4.45), repeated below.

(4.45) Circular chain shift

\[
\begin{array}{|c|c|}
\hline
\text{longin} & \text{NonId[longin } \in \{-\text{long}\}] \quad * [+\text{long}] \\
\hline
\hline
\varnothing & /o:/[o] \\
\hline
/\alpha / & /o:/[\alpha ] \quad *! \\
\hline
/\alpha / & *! \\
\hline
\varnothing & /o:/[\alpha ] \\
\hline
\end{array}
\]

The case in (4.45) is particularly interesting, because it can be argued that the constraint exhibits both distinction only at lesser similarity and distinction only at greater similarity. Which one results depends on how the situation is analyzed, in particular on which grammatical candidate is selected as the candidate with lesser internal similarity, \(\alpha x\). If \( /o:/[\alpha ]\) is selected as \(\alpha x\), then \(bmx = /o:/[\alpha ]\) has greater internal similarity than \(\alpha x\), but is not grammatical. Since \textsc{NonId[longin } \in \{-\text{long}\}] makes a distinction between the candidates for \(o/\) but not for \(o:/\), it is exhibiting distinction only at lesser similarity, analyzed in that way. On the other hand, if \( /o:/[o] \) is selected as \(\alpha x\), then \(bmx = /o:/[o]\) has greater internal similarity than \(\alpha x\), but is not grammatical. Analyzed in this way, \textsc{NonId[longin } \in \{-\text{long}\}] is exhibiting distinction only at greater similarity.

Given that the circular chain shift shown in (4.10) results from distinction conflict, the examples in (4.45) and (4.10) combine to show that even the special case of circular chain shifts can result from any of the three non-ODM behaviors.

The derived environment effect pattern can also be the result of any of the three non-ODM behaviors. A derived environment effect pattern results from distinction only at lesser similarity in the analysis of Dakota stress in (4.14), from distinction only at greater similarity in the local disjunction analysis in (4.8), and from distinction conflict in the analysis of Flemish Brussels Dutch in (4.40).

The same constraint can exhibit different non-ODP behaviors in different contexts. Arguably, the case of \(\textsc{NonId[longin } \in \{-\text{long}\}]\) in (4.45) shows that the same constraint can exhibit different non-ODM behaviors in the same context, depending on how that context is analyzed.

The same constraint can also cause different non-ODM patterns in different contexts. In the situation described for Dakota stress (4.14), the constraint
4.11 More on relating non-ODP behaviors to non-ODM patterns

**Head-Dep** exhibits distinction only at lesser similarity, resulting in a derived environment effect: \( bmx = /\text{čáp}/[\text{čápa}] \) has a single disparity, the stress on the first vowel. However, the same constraint, again exhibiting distinction only at lesser similarity, can also cause a chain shift, in the same language, if input \( /\text{čápa/} \) is chosen: the optimal candidates \( /\text{čáp}/[\text{čápa}] \) and \( /\text{čápa}/[\text{čápá}] \) form the basis for a chain shift, as illustrated in (4.46).

(4.46) **Head-Dep** causes a chain shift

<table>
<thead>
<tr>
<th>Input</th>
<th>MainFootLeft</th>
<th>Head-Dep</th>
<th>Iambic</th>
</tr>
</thead>
<tbody>
<tr>
<td>( /\text{čáp}/[\text{čápa}] )</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>( /\text{čáp}/[\text{čápá}] )</td>
<td></td>
<td>*!</td>
<td></td>
</tr>
<tr>
<td>( /\text{čápa}/[\text{čápa}] )</td>
<td></td>
<td>*!</td>
<td></td>
</tr>
<tr>
<td>( /\text{čápa}/[\text{čápá}] )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The analysis of stress/epenthesis interaction in Dakota shows that the same constraint, exhibiting the same non-ODP behavior, can be responsible for both chain shift patterns and derived environment effect patterns, with the difference residing in details having little to do with the non-ODP constraint’s behavior. The same non-ODP constraint, **Head-Dep**, exhibits the same behavior for both the derived environment effect in (4.14) and the chain shift in (4.46). The only difference between the two is the specification of underlying stress in the input of the greater-similarity input. In the chain shift, the greater-similarity candidate \( bmx = /\text{čáp}/[\text{čápa}] \) has underlying stress in the same position as the output for the lesser-similarity candidate (the first link of the chain) \( akx = /\text{čáp}/[\text{čápa}] \), so that \( In_b \) matches \( OUT_s \). In the derived environment effect, the lesser-similarity candidate \( bmx = /\text{čáp}/[\text{čápá}] \) does not have underlying stress specified, so a disparity in stress remains between \( In_b \) and \( OUT_s \). That single distinction between the two cases, the (lack of) specification of underlying stress on the first vowel, is not even referred to by the constraint **Head-Dep**. The distinction between chain shift and derived environment effect isn’t relevant to a proper understanding of the non-output drivenness here.

If a constraint exhibits behavior that is not ODP, then there is potential for a non-output-driven map to result. Whether the resulting map is in fact non-output-driven depends upon the interaction of that constraint with the others in the relevant grammatical context (the relevant grammatical context being the candidates of the relevant competitions, and the ranking of the constraints). If the result is a non-output-driven map, the particular kind of non-output-driven map pattern that results also depends on the interaction of
the non-ODP constraint with the other constraints in the relevant grammatical context.

4.12 Maps and grammars

It is possible to define an overall grammar as the composition of several component maps, with the overall grammar itself also being a map. This has multiple precedents in phonological theory. A system of ordered rules (Chomsky and Halle 1968) defines a composition of maps. Each rule of the grammar defines a phonological map, with each mapping relating an input (the representation before the application of the rule) to an output (the representation after the application of the rule). The map defined by the grammar results from the ordered composition of the maps defined by the rules. Lexical phonology (Kiparsky 1982, 1985) breaks a grammar into composed maps along morphological lines. In these cases, the map defined by the grammar results from the composition of component maps.

When a grammar consists of the composition of maps, there are at least two ways that the grammar can give rise to non-output drivenness. One is to have a basic component map that is non-output-driven. The other is to have the component maps interact in a way that makes the overall map non-output-driven. The composition of two or more identical output-driven maps is always equivalent to a single instance of that map, because output-driven maps are idempotent: a possible output of the map always maps to itself. But two distinct output-driven maps, when composed, can result in an overall map that is not output-driven. Indeed, traditional analyses of process opacity/transparency focus on interactions between component maps (the rules) as determinants of surface orientedness in the map for the overall grammar.

An Optimality Theoretic grammar is often taken to be a grammar for a language, but a grammar for a language can also be defined as a composition of Optimality Theoretic component grammars (Itô and Mester 2001, Kiparsky 2003, McCarthy and Prince 1993b), where the output of a grammatical candidate of one component grammar serves as input to another component grammar. Kiparsky’s theory of stratal OT (2003, forthcoming) conceives of a grammar as a serial composition of maps, each of which is realized by an Optimality Theoretic map. Bermúdez-Otero has argued that each of the individual OT maps in stratal OT should be non-opaque, in the sense of process-based phonological opacity, and that opacity should arise only as the consequence of interaction between non-identical OT strata within the grammar (Bermúdez-Otero 2003). An alternative approach, also employing Optimality Theory, is proposed by
Itô and Mester. They also envision a grammar consisting of the composition of at least two maps each of which is defined by OT rankings. However, they suggest that different kinds of phonological opacity should be handled in different ways, with some the result of opaque maps defined by individual OT rankings, and others the result of interaction between the composed rankings of the grammar (Itô and Mester 2001, 2003).

It is not hard to imagine analogous claims for the structure of linguistic theory being made with respect to output drivenness. A proposal analogous to that of Bermudez-Otero would claim that all input-referring constraints are ODP, and that non-output drivenness only arises as a consequence of interactions between composed stratal maps.

4.13 The map

Each of the core non-output-driven map patterns (chain shifts, derived environment effects) can be the result of any of the three types of non-ODP constraint behavior (distinction only at lesser similarity, distinction only at greater similarity, distinction conflict). This highlights the extent to which the map patterns, the constraint behaviors, and their relationships are variations on a single phenomenon. As described in Section 4.11, the very same violation pattern for the non-ODP constraint in (4.45) can be construed as either distinction only at lesser similarity or as distinction only at greater similarity, depending on which output of the map is focused upon. Either way you look at it, the same map is not output-driven, a consequence of the same non-ODP constraint. The discussion of Dakota at the end of Section 4.11 observed that the same constraint exhibiting the same non-ODP behavior can be responsible for both a chain shift and a derived environment effect. This coheres with the earlier discussion of Section 2.1.3: the chain shift and derived environment effect patterns are minor variations of the same basic thing, a non-output-driven map.

The examination of derived environment exchanges in Sections 4.9 and 4.10 are one demonstration of the power and insight that are provided by the theory of output-driven maps, and especially by the concept of relative similarity. Relative similarity transforms the simple distinction between identity candidates and non-identity candidates into a full relational structure on the space of candidates sharing an output. Relative similarity provides a way to compare and relate candidates that each have multiple disparities, providing a concrete sense of when one candidate is “more faithful than” another. This makes it possible to reason about constraints in OT grammars more generally, describing with some generality the map implications of faithfulness constraint
behavior even in cases where multiple disparities of distinct kinds are involved, as in derived environment exchanges.

The rest of this book demonstrates the power of the same set of ideas in language learning. The basic situation faced by a language learner is actually somewhat similar to the situation used in describing output drivenness. A learner observes overt outputs and needs to determine the correct inputs and the map that produces the correct outputs for the correct inputs. Given an observed output, the learner’s space of possible analyses of that form consist of the space of candidates that share that output, the same grouping property that determines the space of candidates that can possibly be in a relative similarity relation. The theory of output drivenness provides structure on the space of possible underlying forms that can be exploited to great effect by learners learning the lexicon of underlying forms for their language.

Work in phonotactic learning (Hayes 2004, Prince and Tesar 2004) has proceeded largely on the view that early learners can make progress in learning the map of their language by presuming that the language is idempotent: given an output, the learner can presume that an identity mapping on that output is part of the map, even if the identity input ultimately proves not to be the correct input for that particular observed word. This is analogous in some respects to Moreton’s characterization theorem, which arrived at the conclusion of eventual idempotency from the basic distinction between identity and non-identity candidates. Just as the theory of output drivenness pushes our understanding of grammar–map relations into the full space of non-identity mappings, it also pushes our structured understanding of phonological learning beyond identity maps, into the full space of possible underlying forms for morphemes and words.
5 Learning phonotactics

This chapter and the next three concern language learnability. Phonological underlying forms and their mutual dependence with constraint-ranking relations constitute a fundamental issue in the learning of phonological grammars. Output-driven maps impose structure on the space of inputs. That structure has significant consequences for learning: if a learner knows in advance that possible grammars all produce output-driven maps, it can exploit the imposed structure to great effect in learning.

Chapter 5 presents key concepts in the learning of rankings and lays out a specific approach to phonotactic learning. Chapter 6 focuses on the learning of non-phonotactic information: the underlying forms for morphemes, and non-phonotactic ranking information. Chapter 7 then shows how output drivenness can greatly benefit the learning of both underlying forms and constraint rankings, yielding a learning algorithm that exploits output drivenness while building on elements introduced in the preceding two chapters.

In the present chapter, Section 5.1 lays out the basic learning issues to be addressed, and Section 5.2 presents an Optimality Theoretic system involving stress and vowel length that will be used for purposes of illustration throughout the rest of the book (the full typology of the stress/length OT system is given in an appendix, in Section 5.10). Sections 5.3 through 5.5 present selected prior work on the learning of constraint hierarchies. Section 5.6 provides discussion of restrictiveness biases in learning, including a review of selected prior work on restrictiveness biases in the learning of constraint rankings. Section 5.7 presents previously unpublished work on phonotactic contrast, work that will prove extremely useful for connecting the learning of constraint rankings with the learning of underlying forms. Section 5.8 discusses the fundamental limitations of purely phonotactic information for the learning of phonologies, setting the stage for Chapter 6, which will discuss the use of paradigmatic information in learning.
5.1 An overview of the learning problem

Learnability is fundamentally about choosing from among a set of possibilities, based on data. The ease or difficulty of a particular learning problem is determined by the relations that hold between the different possibilities with respect to the kind of data that will be available to the learner. Linguistic theory is of central importance to language learning, because it determines the set of possibilities that the learner has to choose from.

The data typically taken to be available to a phonological learner are the overt portions of (some of) the outputs of a grammar. In word-level phonology, an output of a grammar is a word (possible or actual) of the language, so a set of learning data would contain grammatical words of the language. Words are constructed out of morphemes. The phonological input for a word is formed by combining (in accordance with the morphology) the underlying forms for the morphemes of the word, and the output for a word can be divided into parts such that each part is the surface realization of a morpheme of the word. For present purposes, I will say simply that an output segment is part of the surface realization of a morpheme if it has an input correspondent which is part of the underlying form for that morpheme.

This book intentionally focuses on the aspects of learning that are most directly affected by output drivenness. As a consequence, there are a number of issues that are significant to the overall project of explaining child language acquisition, but that I will not be addressing in any depth in this book. One such issue is the distinction between a complete output and its overt form, the audible portion of the output. The major differences between output forms and their associated overt forms are prosodic structure and morpheme identity. Child learners need to infer the prosodic structure associated with the overt forms they hear, overt forms which can be ambiguous from among several different prosodic interpretations. Child learners also need to determine what the morphemes are, and how those morphemes are realized in the various words containing them. These important and challenging issues are beyond the scope of this book (but see Section 9.4.3 for some discussion of learning prosodic structure). The particular learning problem addressed in this book involves learning data consisting of full outputs, in which the surface realizations of the morphemes and the prosodic structure are included.

A grammar has two parts that need to be learned: a constraint hierarchy and a lexicon of underlying forms. In keeping with the principle of Richness of the Base (Prince and Smolensky 1993/2004: 225), the space of possible phonological inputs is universal. The lexicon of underlying forms, however,
is language-specific, and in a sense dataset-specific: a learner will end up with an underlying form in their lexicon for each morpheme actually observed in the dataset. The learning problem is to find a constraint hierarchy and a lexicon of underlying forms that satisfy a couple of criteria. First, for each word in the data, the input for that word (the combined underlying forms for the morphemes of the word) must be mapped to the observed output by the constraint hierarchy. Put another way, the grammar must be consistent with the data. Second, the grammar must be (one of) the most restrictive of the grammars that are consistent with the data.

Data consistency (the first criterion) evaluates the constraint hierarchy of a grammar with respect to the lexicon of that grammar. Restrictiveness (the second criterion) evaluates the constraint hierarchy not with respect to the lexicon, but with respect to the rich base of possible inputs, and in particular the overt forms for the set of outputs generated for the possible inputs. One hierarchy is more restrictive than another if the former generates a set of overt forms that is a strict subset of the set of overt forms generated by the latter. Other things being equal, a constraint hierarchy that only generates words with stress on the initial syllable is more restrictive than a hierarchy which generates both words with stress on the initial syllable and words with stress on the final syllable (assuming multisyllabic words).

What makes this learning problem interesting and challenging is the interdependence of the ranking relations and the underlying forms. Data consistency requires that the learner properly capture morphemic alternations. When a morpheme surfaces differently in different words, the grammar must map the underlying form for the morpheme to its correct surface realization for each word. The underlying form for one morpheme is interdependent with the underlying forms of other morphemes, as well as with the constraint hierarchy. Restrictiveness requires that the learner capture phonotactic restrictions even when the enforcement of those restrictions is not directly indicated by alternations. If no morphemes are observed to alternate in vowel length, because there are no surface long vowels in any of the words, the learner must ensure that their learned constraint hierarchy enforces the pattern, so that even inputs with long vowels do not surface as outputs with long vowels.

5.2 Stress and length: a system for illustration

This section defines an Optimality Theoretic system, the Stress/Length system, that will be used for purposes of illustration throughout the rest of the book. It
Learning phonotactics exhibits the complexities that are of interest, while remaining small enough to be the basis for reasonably sized illustrations of the learning proposals.

Each word consists of a root and a suffix (both monosyllabic). The consonants are not of concern in this system; as a readability aid, all root syllables have initial consonant “p,” and all suffix syllables have initial consonant “k.” Each vowel has two features. The length feature has the values long (+) and short (−). The main stress feature has the values stressed (+) and unstressed (−). The constraints are as shown in (5.1).


<table>
<thead>
<tr>
<th>Constraint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MainLeft</strong></td>
<td>main stress on the initial syllable</td>
</tr>
<tr>
<td><strong>MainRight</strong></td>
<td>main stress on the final syllable</td>
</tr>
<tr>
<td><strong>NoLong</strong></td>
<td>no long vowels</td>
</tr>
<tr>
<td><strong>WSP</strong></td>
<td>long vowels must be stressed (weight-to-stress principle)</td>
</tr>
<tr>
<td><strong>Ident[stress]</strong></td>
<td>IO correspondents have equal stress value</td>
</tr>
<tr>
<td><strong>Ident[length]</strong></td>
<td>IO correspondents have equal length value</td>
</tr>
</tbody>
</table>

This system defines a typology of twenty-four languages; all twenty-four languages are listed in an appendix in Section 5.10. One of the languages, language L20, is shown in (5.2). It is generated by the constraint ranking given in (5.3). Briefly stated, L20 has lexical stress, with stress on the initial syllable by default, and long vowels shorten in unstressed position.

(5.2) L20

<table>
<thead>
<tr>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
<th>s1 = /-ka/</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>pá:ka</td>
<td>páka</td>
<td>pá:ka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>páka</td>
<td>pá:ka</td>
<td>páka</td>
<td>pá:ka</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>páka</td>
<td>pá:ka</td>
<td>s3 = /-ká/</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>páka</td>
<td>pá:ka</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>

(5.3) WSP $\gg$ Ident[stress] $\gg$ MainLeft $\gg$ MainRight $\gg$ Ident[length] $\gg$ NoLong

Because this system only permits feature value disparities (no deletion or insertion is allowed), a given input has only a finite number of candidates. The correspondence relation is an order-preserving bijection: each segment of the
input has exactly one corresponding segment in the output. Thus, for a given input–output pair, there is only one possible IO correspondence relation (the obvious one).

One restriction on GEN results in a distinction between the space of possible inputs and the space of possible outputs. Possible outputs are constrained to have exactly one stressed vowel; no such restriction applies to possible inputs. The possible inputs and outputs are shown in (5.4).\(^1\)

(5.4) Inputs and outputs for the Stress/Length system (inputs in top row, outputs in bottom row)

<table>
<thead>
<tr>
<th>paka</th>
<th>pa:ka</th>
<th>paka:</th>
<th>pa:ka:</th>
<th>páka</th>
<th>pá:ka</th>
<th>pá:ka:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>paka</td>
<td>p:ka</td>
<td>p:ka:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>pa:ká</td>
<td>paká:</td>
<td>pa:ká:</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.3 Constructing a constraint hierarchy from winner–loser pairs

5.3.1 Recursive Constraint Demotion

Recall, from Section 3.1.1, the concept of a winner–loser pair: a pair of competitors (they must have the same input) with one identified as the winner and the other identified as the loser. A winner–loser pair is the basic unit of constraint ranking information: it specifies what must be true of the ranking in order for the winner to be more harmonic than the loser. Whenever a constraint hierarchy is such that the winner of a winner–loser pair is more harmonic than the loser, the winner–loser pair will be said to be satisfied by that constraint hierarchy.

\(^1\) It is straightforward to generalize this system to allow morphemes (and words) with differing numbers of vowels (syllables). GEN is partitioned by the number of vowels in forms: all of the possible one-vowel output forms constitute the possible outputs for a one-vowel input, all of the possible two-vowel output forms constitute the possible outputs for a two-vowel input, and so forth. For any given number of vowels, the subspace of possible inputs will be larger than the subspace of possible outputs. For a given number \(v\) of vowels, the number of possible inputs is \(2^{2v}\); there are two binary features per vowel for a total of \(2v\) features, and all possible combinations of values of the features are possible, yielding \(2^{2v}\) possible outputs. The number of possible outputs is \(v2^v\). There is one length feature per vowel for a total of \(v\) length features, all possible combinations of values of the length features are possible, yielding \(2^v\) combinations. Exactly one vowel must be stressed, and there are \(v\) vowels to choose from, yielding \(v\) combinations. Free combination of the length feature possibilities and the stress possibilities yields a total of \(v2^v\) possible output forms.
A winner–loser pair is profitably viewed as an ERC which must be satisfied by the target grammar. Consider a winner–loser pair with winner candidate /pá-ká:/ [páka] and loser candidate /pá-ká:/ [paká:], shown in (5.5). The Input column shows the input shared by both winner and loser, the Winner column shows the output of the winner, and the Loser column shows the output of the loser. To aid readability, constraints that are indifferent on a pair have the corresponding cell left blank (rather than marked with an “e”). This pair is satisfied when at least one of MainLeft and NoLong dominates both MainRight and Ident[length].

(5.5) A winner–loser pair

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
<th>WSP</th>
<th>Ident[length]</th>
<th>Ident[stress]</th>
</tr>
</thead>
<tbody>
<tr>
<td>pá-ká:</td>
<td>páka</td>
<td>paká:</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In general, a constraint hierarchy adequate to resolve the conflicts of a particular language cannot be determined by a single winner–loser pair. Recursive Constraint Demotion, or RCD, is an algorithm for finding a constraint hierarchy consistent with a list of winner–loser pairs when one exists (Tesar 1995, Tesar and Smolensky 1998). RCD returns a hierarchy with a particular property relative to the set of winner–loser pairs: it finds the hierarchy in which every constraint is ranked as high as possible. When there are multiple constraint hierarchies that are consistent with the winner–loser pairs, RCD will always select that one; it is biased toward the hierarchy (consistent with the winner–loser pairs) with each constraint ranked as high as possible.

RCD repeatedly iterates through two steps. The first step is to identify those constraints that can be ranked above the others, consistent with the current set of winner–loser pairs, and to place those constraints into the ranking. The second step is to identify those winner–loser pairs that are now satisfied because of the constraints that were just ranked, and remove them from the working list. The two steps constitute a pass through the list of pairs by RCD. Each pass places more constraints into the ranking and removes more winner–loser pairs from the list, until no winner–loser pairs remain, and all the constraints have been ranked. The constructed hierarchy satisfies all of the winner–loser pairs, and has each constraint ranked as high as possible.

The constraints which can be ranked above the other remaining constraints will be precisely the constraints which do not prefer the loser for any of the winner–loser pairs. If a loser-preferring constraint is ranked above the others,
then that loser will beat its corresponding winner, which would be inconsistent. Any constraint not preferring any losers could consistently be ranked above the others. The “as high as possible” bias of RCD requires that all such constraints be ranked highest. Effectively, the first step of RCD on a given pass finds those constraints that do not assign L to any of the remaining winner–loser pairs.

Consider the winner–loser pairs in (5.6). There are six winner–loser pairs, and six constraints. Of the six constraints, two do not prefer any losers: WSP and IDENT[stress]. Therefore, those two constraints can be ranked at the top of the hierarchy. The data provide no basis for choosing one to dominate the other: the two constraints do not conflict on any of the given winner–loser pairs. Because there is no conflict to be resolved, the two constraints are grouped together into a single stratum of the hierarchy. Putting these two constraints into the top stratum of the hierarchy ensures that they will dominate every other constraint, as RCD will necessarily place the other constraints into lower strata.

(5.6) The set of winner–loser pairs before the first pass of RCD

<table>
<thead>
<tr>
<th>Pair</th>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
<th>WSP</th>
<th>IDENT[length]</th>
<th>IDENT[stress]</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>pá-ka: páka</td>
<td>páka:</td>
<td></td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b.</td>
<td>pá-ka</td>
<td>páka</td>
<td>paká</td>
<td></td>
<td></td>
<td></td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c.</td>
<td>pa-ká</td>
<td>paká</td>
<td>páka</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d.</td>
<td>pá-ká</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e.</td>
<td>pá-ká:</td>
<td>páka</td>
<td>paká:</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td>L</td>
</tr>
<tr>
<td>f.</td>
<td>pa-ká:</td>
<td>paká:</td>
<td>paká</td>
<td></td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A winner–loser pair is satisfied by having WSP and IDENT[stress] dominate all of the other constraints if at least one of WSP and IDENT[stress] prefers the winner of the pair. If one of the just selected constraints prefers the winner, then the winner must win, because the constraint necessarily dominates any constraint that would prefer the loser. Note that it is not possible for one of the selected constraints to prefer the winner and another of the selected constraints to prefer the loser for the same winner–loser pair, because to be selected a constraint cannot prefer any of the losers for the remaining pairs. In the example, three of the six pairs have their losers for the remaining pairs. In the example, three of the six pairs have their winners preferred by one of {WSP, IDENT[stress]}: (5.6)a, (5.6)b, and (5.6)c. Because those pairs have now been accounted for, they can be (temporarily) removed: they are accounted for by
the ranking relations just established, so there is nothing further to learn from them. The result of the first pass of RCD is the partial hierarchy in (5.7), with a single stratum containing two constraints, and the reduced set of winner–loser pairs in (5.8), with three pairs remaining and four constraints not yet ranked in the hierarchy.

\[(5.7) \quad \{\text{WSP, Ident[stress]}\}\]

\[(5.8) \quad \text{The remaining winner–loser pairs after the first pass of RCD}\]

<table>
<thead>
<tr>
<th>Pair</th>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
<th>Ident[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>d.</td>
<td>pá-ká</td>
<td>páká</td>
<td>paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>e.</td>
<td>pá-ká: páká</td>
<td>paká:</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>f.</td>
<td>pa-ká: paká:</td>
<td>paká</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note that the constraints that were placed into the hierarchy have been removed from the table showing the remaining winner–loser pairs. That is because those constraints necessarily have no preferences for any of the remaining winner–loser pairs: they cannot have preferred any of the losers, and pairs in which they preferred the winner have been removed.

The learner is now confronted with a smaller version of the same problem: the remaining constraints need to be ranked as high as possible consistent with the remaining winner–loser pairs. RCD therefore does the same thing over again, on the remaining pairs and constraints. This is the “recursive” part of RCD. The second pass of RCD thus determines that, of the four remaining constraints, there is one, **MainLeft**, that does not prefer the loser for any of the remaining pairs. **MainLeft** is placed into the next stratum of the hierarchy, ensuring that it will be dominated by WSP and **Ident[stress]** but will dominate the other three constraints. With **MainLeft** now ranked, two of the three remaining winner–loser pairs are now accounted for, the ones for which **MainLeft** prefers the winner. The state of things after the second pass is shown in (5.9) and (5.10): **MainLeft** occupies the second stratum of the constraint hierarchy, and one winner–loser pair remains.

\[(5.9) \quad \{\text{WSP, Ident[stress]}\} \gg \{\text{MainLeft}\}\]

\[(5.10) \quad \text{The remaining winner–loser pairs after the second pass of RCD}\]

<table>
<thead>
<tr>
<th>Pair</th>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>MR</th>
<th>NoLong</th>
<th>Ident[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>f.</td>
<td>pa-ká:</td>
<td>paká:</td>
<td>paká</td>
<td>L</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>
RCD then continues, making a third pass, which places \texttt{MainRight} and \texttt{Ident[length]} into the next stratum, but not \texttt{NoLong}. The state of things after the third pass is shown in (5.11) and (5.12): \texttt{MainRight} and \texttt{Ident[length]} occupy the third stratum of the hierarchy, and no winner–loser pairs remain unaccounted for.

(5.11) \( \{\text{WSP, Ident[stress]}\} \gg \{\text{MainLeft}\} \gg \{\text{MainRight, Ident[length]}\} \)

(5.12) The remaining winner–loser pairs after the third pass of RCD

<table>
<thead>
<tr>
<th>Word</th>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>\texttt{NoLong}</th>
</tr>
</thead>
</table>

No winner–loser pairs remain; all have been accounted for. Therefore, any constraints that have not yet been ranked may now be put into the bottom stratum; there are no losers left to prefer. The fourth pass is thus the final one for RCD in this example. The final constraint hierarchy derived by RCD is shown in (5.13).

(5.13) \( \{\text{WSP, Ident[stress]}\} \gg \{\text{MainLeft}\} \gg \{\text{MainRight, Ident[length]}\} \gg \{\text{NoLong}\} \)

5.3.2 Stratified constraint hierarchies

The constraint hierarchy in (5.13) is a stratified constraint hierarchy. A stratified hierarchy is a totally ordered set of strata, where each stratum contains one or more constraints. Ranking relations between constraints are entailed by ranking relations between strata. In (5.13), \texttt{WSP} \( \gg \texttt{MainRight} \), because the stratum containing \texttt{WSP} is above the stratum containing \texttt{MainRight} in the hierarchy.

Stratified hierarchies are not necessarily total rankings of constraints; a stratified hierarchy must have only one constraint per stratum to be a total ranking. A hierarchy fails to establish a ranking relation between two constraints only if the two constraints inhabit the same stratum of the hierarchy. The hierarchy in (5.13) is a non-total ranking, but it does resolve all of the constraint conflicts in the winner–loser pairs. This can be seen more easily if the constraint columns are arranged so that they mirror the positions of the constraints in the constraint hierarchy left-to-right, as in (5.14); the solid vertical lines indicate stratum boundaries, while multiple columns within a stratum are separated by dotted vertical lines. Each row with a winner–loser pair has a W as its left-most non-blank evaluation; every L in the tableau is dominated by a W in a higher stratum.
(5.14) The winner–loser pairs, with constraint columns sorted according to the hierarchy returned by RCD

<table>
<thead>
<tr>
<th>Pair</th>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>pá-ka:</td>
<td>páka</td>
<td>pákα:</td>
<td>W</td>
<td></td>
<td></td>
<td>L</td>
<td></td>
<td>W</td>
</tr>
<tr>
<td>b.</td>
<td>pá-ka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td>W</td>
</tr>
<tr>
<td>c.</td>
<td>pa-ká</td>
<td>paká</td>
<td>pákα</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d.</td>
<td>pá-ká</td>
<td>pákα</td>
<td>paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e.</td>
<td>pá-ká:</td>
<td>pákα</td>
<td>paká:</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td>W</td>
</tr>
<tr>
<td>f.</td>
<td>pa-ká:</td>
<td>paká:</td>
<td>paká</td>
<td>W</td>
<td></td>
<td></td>
<td>L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For any given consistent list of winner–loser pairs, RCD will derive the stratified hierarchy in which each constraint is as high as possible. It does so by constructing the hierarchy from the top down, first building the top stratum, then the second stratum, and so forth. Each pass results in the construction of a stratum of the hierarchy, so the number of passes that RCD will make is the number of strata, which is at most the number of constraints.

If the list of winner–loser pairs is sufficient to resolve all constraint conflicts in a language, then the constraint hierarchy produced by RCD will be a sufficient hierarchy: it will fully decide the candidate competition for any input. If the list of winner–loser pairs is not sufficient to resolve all constraint conflicts in the language, then there may be some inputs for which the constraint hierarchy doesn’t fully decide the competition. A constraint hierarchy can fail to resolve a conflict when more than one constraint exists in a stratum; it is precisely those pairs of constraints that coexist in a single stratum that have their dominance relations undefined. It is not the case that all stratified hierarchies with a stratum containing more than one constraint fail to fully decide all candidate competitions; some pairs of constraints may simply never conflict in some languages. If a list of winner–loser pairs does not require a dominance relation between two constraints, it is possible (but not necessary) that the hierarchy produced by RCD will have those constraints in the same stratum.

The virtue of constructing a constraint hierarchy for a list of winner–loser pairs is that a hierarchy directly and compactly represents ranking relations that are implicit in the winner–loser pairs. If a hierarchy does not resolve all conflicts present in the language, then some further criteria are needed if one wants to use the hierarchy to decide candidate competitions for inputs, specifically for those competitions in which the unresolved conflicts arise. Such criteria go beyond the concerns of Optimality Theory proper, which is concerned with the
properties of fully defined grammars (with all conflicts resolved), but are well within the concerns of language learning and processing. Section 5.4 discusses issues and proposals concerning the use of general stratified hierarchies to evaluate candidate competitions.

The stratified constraint hierarchy produced by RCD is directly determined by the list of winner–loser pairs given as input to RCD. The hierarchy is not generally an exact expression of the information in the list of winner–loser pairs, however; information loss can occur with RCD, a consequence of RCD’s imperative of producing a stratified hierarchy. For example, while the winner–loser pairs of (5.6) are consistent with the hierarchy in (5.13) as shown above, they are also consistent with the hierarchy in (5.15), as shown in the tableau in (5.16) (note that again, for each winner–loser pair, each L is dominated by a W in a higher stratum). In (5.15), the constraint WSP is dominated by three constraints, IDENT[stress], MAINLEFT and MAINRIGHT; in (5.13), WSP is not dominated by any constraints. In fact, the two hierarchies contain some direct reversals. For instance, (5.13) has WSP dominating MAINLEFT, while (5.15) has MAINLEFT dominating WSP. Neither dominance relation is entailed by the winner–loser pairs; both are consistent with the winner–loser pairs.

\[(5.15) \{\text{IDENT}[\text{stress}]\} \gg \{\text{MAINL}\} \gg \{\text{MAINR}\} \gg \{\text{WSP}\} \gg \{\text{IDENT}[\text{length}]\} \gg \{\text{NoLONG}\}\]

\[(5.16) \text{The winner–loser pairs, with constraint columns sorted according to the hierarchy in (5.15)}\]

<table>
<thead>
<tr>
<th>Pair</th>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>WSP</th>
<th>IDENT[length]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>pà-ka:</td>
<td>pàka</td>
<td>pàka:</td>
<td></td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b.</td>
<td>pà-ka</td>
<td>pàka</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c.</td>
<td>pà-ká</td>
<td>paká</td>
<td>pàka</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d.</td>
<td>pà-ká</td>
<td>pàka</td>
<td>paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e.</td>
<td>pà-ká:</td>
<td>pàka</td>
<td>paká:</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>f.</td>
<td>pa-ká:</td>
<td>paká</td>
<td>paká:</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The stratified hierarchy returned by RCD, (5.13), can be interpreted as representing a set of possible total rankings, those resulting from the free reranking of constraints in the same stratum relative to each other. Such total rankings are sometimes labeled “refinements” of the stratified hierarchy. The hierarchy in (5.13) has four refinements, shown in (5.17).
(5.17) The four refinements of (5.13)

\[
\begin{align*}
\text{WSP} & \gg \text{Ident}[\text{stress}] \gg \text{MainL} \gg \text{MainR} \gg \text{Ident}[\text{length}] \gg \text{NoLong} \\
\text{Ident}[\text{stress}] & \gg \text{WSP} \gg \text{MainL} \gg \text{MainR} \gg \text{Ident}[\text{length}] \gg \text{NoLong} \\
\text{WSP} & \gg \text{Ident}[\text{stress}] \gg \text{MainL} \gg \text{Ident}[\text{length}] \gg \text{MainR} \gg \text{NoLong} \\
\text{Ident}[\text{stress}] & \gg \text{WSP} \gg \text{MainL} \gg \text{Ident}[\text{length}] \gg \text{MainR} \gg \text{NoLong}
\end{align*}
\]

The four refinements are all consistent with the winner–loser pairs, but none of them is the same as the total ranking in (5.15). It is often not possible to represent all total rankings consistent with a set of winner–loser pairs as refinements of a single stratified hierarchy. In the example at hand, WSP has no entailed ranking relations with Ident[stress] or MainLeft. Ident[stress], however, must dominate MainLeft. Attempting to express the lack of dominance relations between WSP and the other two by putting all three into a single stratum has the highly problematic consequence of losing the required ranking relation between Ident[stress] and MainLeft; such a stratified hierarchy is simply not consistent with the winner–loser pairs.

These observations suggest that, while a constructed hierarchy is very useful, one should retain the list of winner–loser pairs if one does not want to suffer significant information loss. The learning theory developed in the rest of this chapter relies heavily on the retention and processing of lists of winner–loser pairs.

5.3.3 Constraint conflict and ranking relations

In Optimality Theory, constraints can conflict in the comparison of a winning candidate and one of its losing competitors. A set of constraints conflict on such a comparison if at least one of the constraints prefers the winner and at least one of the constraints prefers the loser. A conflict is resolved when ranking relations are posited between the constraints sufficient to determine that one of the constraints preferring the winner dominates all of the constraints preferring the loser. The dominating constraint decides the comparison in favor of the winner.

The purpose of positing ranking relations is to resolve constraint conflicts between winners and their competitors. If conflicts never arise, then there is no point to having ranking relations, as the winners will win no matter what ranking relations are adopted. A set of ranking relations fully determines a language when it resolves all constraint conflicts in favor of the winners of the language.\(^2\)

\(^2\) If none of the constraints has a preference between a pair of candidates, then no ranking of the constraints will distinguish the two. Either both candidates will be co-winners, or both will lose
A common occurrence in Optimality Theoretic grammars is that some constraints conflict on comparisons with winners while others do not. In such a case, the ranking relations necessary to determine the language will not include a ranking relation between every pair of constraints. There is no obvious harm in positing such ranking relations, so long as they stay consistent with the other ranking relations, but the language itself does not require them.

The significance of total rankings in Optimality Theory is that a total ranking determines a ranking relation between every pair of constraints, and thus ensures that every possible constraint conflict is resolved. Every language generated by an Optimality Theoretic system is generated by at least one total ranking of the constraints, because the necessary ranking relations can always be enhanced by additional ones to arrive at a total ranking (often in multiple different ways).

Despite the aesthetic appeal of total rankings, there is no definitional requirement in Optimality Theory that a learner commit to a total ranking of constraints in order to have learned a language. What is required is that the learner commit to ranking relations sufficient to resolve all constraint conflicts between the winners and their competitors.

### 5.4 Selecting winner–loser pairs

RCD takes as input a list of winner–loser pairs. A learner, however, will not directly observe winner–loser pairs, only winners, those winners that are actually occurring forms in the language. The learner must actively construct appropriate winner–loser pairs, based on the forms it observes. The number of possible winner–loser pairs is large for even modest Optimality Theoretic systems, and infinite for systems permitting outputs of unbounded size relative to their input (due to epenthesis, for instance). However, the entire space of possible winner–loser pairs will generally contain a massive amount of redundancy: many winner–loser pairs will contain the same information as other winner–loser pairs, and some pairs may contain no information at all (when the loser is harmonically bounded by the winner). There is a modest provable bound (a function of the number of constraints) on the number of winner–loser pairs necessary to determine the ranking for a language. The challenge for the learner is to efficiently assemble a compact list of winner–loser pairs that determines the correct ranking.

to some other candidate. Candidates that are not distinguished by any of the constraints are said to have identical constraint violation profiles.
In this section, the selection of winner–loser pairs will be described in terms of the learner being provided with full winners as data, meaning full structural descriptions, including the correct input forms. The challenge of simultaneously learning the correct input forms is discussed in Chapter 6 and Chapter 7.

5.4.1 Error detection
A common strategy for gaining information about a grammar based on positive data is error-driven learning. Error-driven learning presumes that, at any given time, a learner has a complete hypothesized grammar. A form is consistent with a hypothesis if there exists some structural description for the form that is grammatical with respect to the hypothesis. This approach has an implicit assumption: if nothing else, a learner can use a grammar hypothesis to render a grammaticality judgment for a form.

Error-driven learning proceeds by confronting data forms sequentially (for instance, as they are encountered). It confronts a form by checking its current hypothesis for consistency with the form, a process called error detection. If the hypothesis is consistent with the form, nothing changes, and the learner proceeds to the next form. If the hypothesis is inconsistent with the form, the learner contemplates changing its hypothesis (it may or may not actually do so); if the learner decides to change its hypothesis, it does so before proceeding to the next form. If the learner at some point selects a hypothesis which is correct, then it should not subsequently detect any inconsistencies and will retain that hypothesis from that point forward. Typically, the learner does not at any point conclude that learning is finished; it continues indefinitely, verifying the consistency of its hypothesis with each form it encounters.

Error-driven learning is a very general learning strategy. The term error-driven learning comes from Wexler and Culicover (1980), with “error” referring to the inconsistency between the observed form and the learner’s hypothesis; this is an error in the sense that the learner has made a prediction (based upon their current hypothesis) that has been contradicted by the observed form. The general idea goes back further. The use of such a strategy dates back at least to Gold (1967) in formal language learning, and even earlier beyond language, such as in some versions of the perceptron learning algorithm (Rosenblatt 1958). The general strategy makes no particular commitment as to how the learner responds to an error, save that the occurrence of an error is the only occasion on which the learner considers changing their hypothesis. Despite this generality, not all learning approaches are variants of error-driven learning: one example within language learning of a non-error-driven approach is cue learning (Dresher 1999, Dresher and Kaye 1990).
Informative losers for winners can be selected using error detection (Tesar 1998b). Given a winner, the learner determines whether or not the winner is most harmonic under the learner’s current hypothesis. The learner does this by taking the input of the winner and computing the candidates that are most harmonic with respect to the learner’s current hypothesis. This computation is the standard generation of an optimal candidate for an input by a grammar, and is often called production-directed parsing (Tesar and Smolensky 1998).

If the learner’s hypothesis generates a different candidate than the observed winner, then the learner’s hypothesis is inconsistent with the winner; an “error” has been detected (in the sense of error-driven learning).

This form of error detection has the virtue of producing both parts of the winner–loser pair at once. When an error occurs, the winner is the observed form that the learner’s hypothesis fails to generate, while the loser is the form that the learner’s hypothesis does generate for the same input. Because the learner’s hypothesis assigns greater or equal harmony to the loser than to the winner, the winner–loser pair comparing the two is guaranteed to provide information about the ranking that is not currently reflected in the learner’s hypothesis; the selected winner–loser pair is guaranteed to be informative. This is a very significant benefit that comes from an optimization-based linguistic theory: the generated loser doesn’t just indicate that the winner is inconsistent with the learner’s current hypothesis, it forms a winner–loser pair indicating how the hypothesis needs to be changed in order to rectify the error.

5.4.2 Production-directed parsing with stratified hierarchies
An OT-based learner that always maintained a single total ranking of constraints as its grammar hypothesis could straightforwardly use production-directed parsing for error detection. However, there are other significant problems that arise when attempting to maintain a single total ranking during learning (Broihier 1995). As a consequence, most approaches to learning OT rankings instead use stratified hierarchies, or some other more flexible representation (such as the ranking values of Stochastic OT, or the numeric weights of Harmonic Grammar).

A complication arises for learners that represent hypotheses using stratified hierarchies. A stratified hierarchy does not always fully determine ranking relations between all constraints. Even if a learner possesses a stratified hierarchy that resolves all constraint conflicts for the data it has seen, it might well have two constraints in the same stratum that conflict with respect to some other form not (yet) seen. If stratified hierarchies are used to represent hypotheses, then
using production-directed parsing to identify informative losers requires some way of interpreting stratified hierarchies for purposes of production-directed parsing.

The next three subsections describe three techniques that have been used to evaluate stratified hierarchies for production-directed parsing: Mark Pooling, Conflicts Tie, and Variationist EDCD. All three techniques have the property of being equivalent to standard lexicographic evaluation when given a stratified hierarchy defining a total ranking, with only one constraint per stratum. Each has quirks; for further discussion see (Boersma 2009, Tesar 2000). None of the three yields a “perfect” solution to error detection for the generation of informative losers; this will be illustrated below.

5.4.2.1 Mark pooling
Mark pooling, abbreviated Pool, treats all of the constraints within a stratum as if they were one composite constraint (Tesar 1995). The violations assessed a candidate by all of the constraints in the stratum are “pooled” together, and treated as if they were all violations of a single constraint (the “stratum constraint”).

The tableau in (5.18) shows the candidates for input /páka/, along with their violation profiles. The tableau in (5.20) shows the competition for /páka/ with respect to the monostratal stratified hierarchy, shown in (5.19), using Pool. A monostratal hierarchy is one with only a single stratum containing all of the constraints, the extreme case in which no ranking relation commitments are made.

(5.18) Violation profiles for the competition for /páka/

<table>
<thead>
<tr>
<th>/páka/</th>
<th>WSP</th>
<th>NoLong</th>
<th>ML</th>
<th>MR</th>
<th>Ident[length]</th>
<th>Ident[stress]</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>paká</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>**</td>
</tr>
<tr>
<td>pá:ka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>pa:ká</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>pá:ka:</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>paká:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>pá:ka:</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>pa:ká:</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
</tbody>
</table>

(5.19) {WSP, Ident[stress], MainLeft, MainRight, Ident[length], NoLong}
5.4 Selecting winner–loser pairs

(5.20) Competition for /páka/ when all constraints are in a single stratum, using Pool

<table>
<thead>
<tr>
<th>/páka/</th>
<th>Stratum1</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>*</td>
</tr>
<tr>
<td>paká</td>
<td>* *! *</td>
</tr>
<tr>
<td>pák:a</td>
<td>* *! *</td>
</tr>
<tr>
<td>pa:ká</td>
<td>* *! * * *</td>
</tr>
<tr>
<td>pá:ka</td>
<td>* *! * *</td>
</tr>
<tr>
<td>pa:ká:</td>
<td>* *! * * *</td>
</tr>
<tr>
<td>pa:ká:</td>
<td>* *! * * * * *</td>
</tr>
</tbody>
</table>

Using the Pool criterion, the learner would generate the output [páka], because it has the fewest violations on the (single) stratum.

When a hierarchy contains more than one stratum, the pooling is performed stratum by stratum, and evaluation proceeds from the top stratum down. The candidates that tie for minimal violation on the first stratum are then compared with respect to the second stratum, and so forth. The tableau in (5.21) shows the candidates for input /paká:/:, along with their violation profiles, with columns arranged in accordance with the stratified hierarchy in (5.22). The tableau in (5.23) shows the competition for /paká:/ with respect to the stratified hierarchy shown in (5.22), using Pool.

(5.21) Violation profiles for the competition for /paká:/

<table>
<thead>
<tr>
<th>/paká:/</th>
<th>NoLong</th>
<th>Ident[length]</th>
<th>WSP</th>
<th>ML</th>
<th>Ident[stress]</th>
<th>MR</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td>**</td>
<td>*</td>
</tr>
<tr>
<td>paká</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pák:a</td>
<td></td>
<td>* *</td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>pa:ká</td>
<td></td>
<td>* *</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>pá:ka</td>
<td></td>
<td>* *</td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>pa:ká:</td>
<td></td>
<td>* *</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pa:ká:</td>
<td></td>
<td>* *</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>

(5.22) \{Ident[length], NoLong\} \gg \{WSP\} \gg \{MainLeft\} \gg \{Ident[stress], MainRight\}
Learning phonotactics

(5.23) Competition for /pak´a:/, using Pool and hierarchy (5.22)

Of the eight candidates, only four survive the first stratum, those with a total of only one violation. Of those four, one is eliminated on the second stratum, and two more are eliminated on the third stratum, leaving the candidate [páka] as the sole generated candidate. Notice that the generated candidate is not the candidate with the fewest number of violations across the entire constraint set. The constraints of the first stratum collectively have priority over the lower strata, and so forth.

It is possible for non-total constraint hierarchies to generate multiple candidates with non-identical violation profiles when using Pool. The tableau in (5.24) shows the candidates for input /pak´a:/, along with their violation profiles, with columns arranged in accordance with the stratified hierarchy in (5.25). The tableau in (5.26) shows the competition for /p´aka/ with respect to the stratified hierarchy shown in (5.25), using Pool.

(5.24) Violation profiles for the competition for /pak´a:/

<table>
<thead>
<tr>
<th>/pak´a/</th>
<th>Stratum1</th>
<th>Stratum2</th>
<th>Stratum3</th>
<th>Stratum4</th>
</tr>
</thead>
<tbody>
<tr>
<td>p´aka</td>
<td>*</td>
<td></td>
<td></td>
<td>***</td>
</tr>
<tr>
<td>paká</td>
<td>*</td>
<td></td>
<td>*!</td>
<td></td>
</tr>
<tr>
<td>p´a:ka</td>
<td>* ! *</td>
<td></td>
<td></td>
<td>***</td>
</tr>
<tr>
<td>pa:ká</td>
<td>* ! *</td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>p´aka:</td>
<td></td>
<td></td>
<td>*!</td>
<td>***</td>
</tr>
<tr>
<td>paká:</td>
<td></td>
<td></td>
<td>*</td>
<td>* *</td>
</tr>
<tr>
<td>p´a:ka:</td>
<td></td>
<td></td>
<td></td>
<td>***</td>
</tr>
<tr>
<td>pa:ká:</td>
<td></td>
<td></td>
<td></td>
<td>***</td>
</tr>
</tbody>
</table>

(5.26) Violation profiles for the competition for /p´aka/:
5.4 Selecting winner–loser pairs

(5.25) \{\text{WSP, IDENT[stress], IDENT[length], NoLong}\} \Rightarrow \{\text{MainLeft, MainRight}\}

(5.26) Competition for /paká:, using Pool and hierarchy (5.25)

<table>
<thead>
<tr>
<th>/paká:</th>
<th>Stratum1</th>
<th>Stratum2</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>* * ! *</td>
<td>*</td>
</tr>
<tr>
<td>paká</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>pá:ka</td>
<td>* * ! * *</td>
<td>*</td>
</tr>
<tr>
<td>pa:ká</td>
<td>* * ! *</td>
<td>*</td>
</tr>
<tr>
<td>pá:ka</td>
<td>* * ! * *</td>
<td>*</td>
</tr>
<tr>
<td>paká</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>pá:ka</td>
<td>* * ! * * *</td>
<td>*</td>
</tr>
<tr>
<td>pa:ká</td>
<td>* * ! * * *</td>
<td>*</td>
</tr>
</tbody>
</table>

Two candidates are generated, [paká] and [paká:]. They have non-identical violation profiles, but the differences in their violation profiles are fully obscured when the violations are pooled within each stratum. Such “ties” between candidates are actually quite useful when searching for informative winner–loser pairs. Even if one of the generated candidates is identical to the winner, the other candidate is not losing to the winner. Forming a winner–loser pair between the winner and the generated candidate distinct from the winner is guaranteed to provide additional ranking information to the learner.

5.4.2.2 Conflicts Tie

Conflicts Tie, abbreviated CTie, tries to make the learner aware of any unresolved constraint conflicts on a stratum (Tesar 2000). Candidates with unresolved constraint conflicts on a stratum are said to tie, and all will be generated if they have not already been eliminated by a higher stratum. CTie is designed to intentionally increase the likelihood of multiple candidates tying, making it more likely that an informative loser will be identified.

Like Pool, CTie evaluates a competition with respect to the top stratum first. It compares candidates on a stratum by first eliminating any candidate that is harmonically bounded by another candidate on that stratum: one candidate is eliminated by another if every constraint of the stratum either prefers the other (eliminating) candidate or is indifferent between the two.
The tableau in (5.28) shows the candidates for /pá:ka/ evaluated with respect to the monostratal hierarchy using CTie. The candidates that are shaded have been eliminated due to harmonic bounding; for each, the candidate that eliminated them is listed in the rightmost column. The third candidate, [pá:ka], is eliminated by the first candidate, [pá:ka]: the only constraints with a preference between them are NoLong and Ident[length], and both of those prefer [pá:ka].

\[(5.27) \{\text{WSP, Ident[stress], MainLeft, MainRight, Ident[length], NoLong}\}\]

\[(5.28) \text{Harmonically bounded candidates are eliminated (shaded)}\]

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pá:ka</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>pá:ka</td>
</tr>
<tr>
<td>paká</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>paká</td>
</tr>
<tr>
<td>pát:ka</td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>pát:ka</td>
</tr>
<tr>
<td>pán:ka</td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>pán:ka</td>
</tr>
<tr>
<td>pán:ka</td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>pán:ka</td>
</tr>
</tbody>
</table>

The first two candidates, with outputs [pá:ka] and [paká], are not harmonically bounded. They also have unresolved conflicts with each other: MainLeft and Ident[stress] prefer [pá:ka], while MainRight prefers [paká]. For the CTie criterion, the fact that more constraints prefer one than prefer the other is irrelevant, as is the fact that one of the candidates has more total violations than the other. Optimality Theory resolves conflicts via ranking, not summation. Thus, CTie determines that two candidates, [pá:ka] and [paká], tie; both are generated. Notice the difference with Pool, which for the same competition on the same monostratal hierarchy generated only [pá:ka], as shown in (5.20). Again, such ties can be useful, because the learner can pick whichever candidate in the tie doesn’t match the winner, and use that candidate as an informative loser.

When a hierarchy has more than one stratum, the harmonic bounding evaluation is performed with respect to a single stratum at a time. The tableau in (5.29) shows the competition for /paká:/ with respect to the stratified hierarchy in (5.22), just as was done with Pool in (5.21). The four shaded candidates are
eliminated due to harmonic bounding on the first stratum, and indicated in the final column.

(5.29) Candidates that are harmonically bounded on the first stratum are eliminated

<table>
<thead>
<tr>
<th>/paká:/</th>
<th>NoLong</th>
<th>Ident[length]</th>
<th>WSP</th>
<th>ML</th>
<th>Ident[stress]</th>
<th>MR</th>
<th>Eliminated by</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pá:ka</td>
<td>*</td>
<td>**</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>páka on Stratum1</td>
</tr>
<tr>
<td>pak:á</td>
<td>*</td>
<td>**</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>páka on Stratum1</td>
</tr>
<tr>
<td>pá:ka:</td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pak:á:</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pá:ka:</td>
<td>**</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td>páka: on Stratum1</td>
</tr>
<tr>
<td>pak:á:</td>
<td>**</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>páka: on Stratum1</td>
</tr>
</tbody>
</table>

Of the remaining four candidates, there is an unresolved conflict between the constraints of the first stratum. None is harmonically bounded by any of the others, but some have non-identical violation profiles. NoLong prefers the first two candidates, while Ident[length] prefers the other two. When CTie detects an unresolved conflict among non-eliminated candidates on a stratum, it declares a tie among all of them, and ends the competition at that point; it does not evaluate the remaining candidates with respect to lower strata. The goal is to return at least one candidate that will provide the learner with ranking information to resolve the conflict, without interference from lower-ranked constraints. Where Pool generated only one candidate, as shown in (5.23), CTie generates four candidates.

CTie does not always produce a different result from Pool, even on hierarchies for non-total rankings. The tableau in (5.31) shows the competition for input /paká:/ with respect to the constraint hierarchy (5.30). The candidate for output [páka] is harmonically bounded on the first stratum by candidate [paká]. Note that [páka] is not harmonically bounded by [paká] over all of the constraints: MainLeft prefers [páka], but is not in the top stratum. With respect to the top stratum, there are two candidates not harmonically bounded by other candidates, [paká] and [paká:].

(5.30) \{WSP, Ident[stress], Ident[length], NoLong\} \gg \{MainLeft, MainRight\}
In (5.31), the CTie criterion declares that candidates [paká] and [paká:] are both generated after the first stratum, because they conflict on constraints in the first stratum.

5.4.2.3 Variationist EDCD

Anttila (Anttila 1997, Anttila and Cho 1998) proposed an interpretation of stratified hierarchies as adult grammars defining a combinatorial set of total rankings. A stratum containing more than one constraint represents a space of possibilities, the possible ranking permutations of the constraints in the stratum. The constraints in a stratum must all be dominated by all the constraints in higher strata, and must dominate all the constraints in lower strata. A candidate is grammatical if it is the optimal candidate for at least one of the total rankings resulting from selecting a permutation of the constraints within each stratum.

The stratified hierarchy in (5.32) has two strata with more than one constraint, the top stratum and the bottom stratum. Each stratum contains two constraints, and thus there are two possible permutations of each stratum independently. A total refinement of a stratified hierarchy into a total ranking results from selecting, for each stratum, a permutation of the constraints of that stratum (a stratum containing only one constraint has only one permutation). The stratified hierarchy in (5.32) has four total refinements, listed in (5.33). In Anttila’s theory, a candidate is grammatical for the grammar in (5.32) if it is optimal for at least one of the rankings in (5.33).

(5.32) \[ \{\text{WSP, IDENT[stress]}\} \gg \{\text{IDENT[length]}\} \gg \{\text{NoLong}\} \gg \{\text{MainLeft, MainRight}\} \]

<table>
<thead>
<tr>
<th>/paká:/</th>
<th>WSP</th>
<th>NoLong</th>
<th>IDENT[length]</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>Eliminated by</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td></td>
<td></td>
<td>*</td>
<td>**</td>
<td></td>
<td></td>
<td>paká on Stratum1</td>
</tr>
<tr>
<td>paká</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>paká on Stratum1</td>
</tr>
<tr>
<td>pá:ka</td>
<td></td>
<td></td>
<td>*</td>
<td>**</td>
<td></td>
<td></td>
<td>paká on Stratum1</td>
</tr>
<tr>
<td>pac:á</td>
<td></td>
<td></td>
<td>*</td>
<td>**</td>
<td></td>
<td></td>
<td>paká on Stratum1</td>
</tr>
<tr>
<td>pá:ka:</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>paká: on Stratum1</td>
</tr>
<tr>
<td>pac:á:</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>paká: on Stratum1</td>
</tr>
<tr>
<td>pá:ka:</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>paká: on Stratum1</td>
</tr>
<tr>
<td>pac:á:</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>paká: on Stratum1</td>
</tr>
</tbody>
</table>
Anttila’s theory essentially holds that a grammar is a set of total rankings, but not just any set: the set must contain all and only the total refinements for some stratified hierarchy. The theory further holds that, when variation occurs, that is, when the same input has more than one grammatical candidate, the relative frequency of the different candidates should directly reflect the proportion of the total refinements for which each candidate is optimal.

The issues in language variation that motivated Anttila’s theory are beyond the scope of this book. Boersma (2009) took Anttila’s theory as the basis for an approach to processing in support of error-driven learning, giving it the label Variationist EDCD (error-driven constraint demotion). Variationist EDCD, when performing parsing, randomly selects a totally ranked refinement of a stratified hierarchy, and parses with respect to that totally ranked hierarchy.\(^3\) The random selection is separately performed for each parse operation, so the same stratified hierarchy can give rise to different total rankings on different occasions. When the learner is working with a non-total constraint hierarchy, whether or not an error is detected for a given winner can depend upon which total refinement is chosen. Repeated processing of the same winner with the same hierarchy will likely parse that winner with respect to different total refinements, increasing over repetition the likelihood of finding an informative loser if one exists.

5.4.3 MultiRecursive Constraint Demotion (MRCD)
A few different versions of error-driven learning for Optimality Theory have been proposed, including Error-Driven Constraint Demotion (Tesar 1998b, Tesar and Smolensky 1998) and the Gradual Learning Algorithm (Boersma 1998, Boersma and Hayes 2001). These versions share the use of production-directed parsing to select informative losers to pair with winners.\(^4\) Both

---

\(^3\) Given the nature of Anttila’s theory, specifically its predictions about variation, one would expect that the random selection is done with respect to a uniform distribution over the possible total refinements.

\(^4\) An approach to the selection of winner–loser pairs that does not involve error-driven learning uses the Contenders algorithm (Riggle 2004); see Section 5.4.5.
Error-Driven Constraint Demotion (EDCD) and the Gradual Learning Algorithm (GLA) retain only a single representation of domination relations among constraints (a stratified hierarchy in the case of EDCD, a set of ranking values in the case of the GLA). For both EDCD and the GLA, detection of an error triggers construction of a winner–loser pair, and then the learner directly modifies their stored domination relations in accordance with the information in that winner–loser pair. As soon as the modification is complete, the winner–loser pair is discarded, leaving the (modified) domination relations as the sole reflection of the data the learner saw.

MultiRecursive Constraint Demotion (Tesar 1997, Tesar 2004), or MRCD, differs from EDCD and the GLA (and from traditional error-driven learning) in that it does not rely on a constraint hierarchy alone as the sole representation in memory of the data the learner has seen. MRCD instead stores each winner–loser pair that it constructs, building up a list of them over time. The list of winner–loser pairs is sometimes called a support, alluding to the fact that it contains precisely the comparisons that are the empirical support for the learner’s ranking hypothesis. Instead of directly modifying a hierarchy, MRCD adds a newly constructed winner–loser pair to its support, and then constructs a new constraint hierarchy by applying (some variant of) RCD to the support. MRCD rederives a complete stratified hierarchy every time a new winner–loser pair is constructed; the “multi” in MultiRecursive Constraint Demotion refers to the fact that RCD is applied repeatedly, as informative data are processed, rather than, for instance, waiting until the learner is done collecting winner–loser pairs and then applying RCD only once. For MRCD, the key representation in memory of what the learner has seen is the support; a constraint hierarchy can be derived at any time from the support. The support will typically not contain explicit representations of all the data the learner has seen; only those that were used to construct winner–loser pairs will appear in the support.

MRCD shares with EDCD and the GLA the use of error detection for deciding when to modify the grammar. By generating a particular stratified hierarchy from the support, it mimics error-driven learning, evaluating an observed form with that generated ranking hypothesis and creating a new winner–loser pair if an error is detected. MRCD uses error detection in evaluating forms, but departs from traditional error-driven learning in storing more (the support) than just a single grammar hypothesis (the stratified hierarchy).

The difference between retaining a support of winner–loser pairs and retaining only a constraint hierarchy is fundamental to the learning theory presented in this book, especially to the parts that involve output-driven maps. As
explained in Section 5.3.2, using a stratified hierarchy to represent the ranking information in a set of winner–loser pairs will involve information loss in many cases. If all a learner has is a stratified hierarchy itself, the learner has no direct way of distinguishing which of the ranking relations represented in the hierarchy were directly entailed by observed data (via constructed winner–loser pairs), and which of the ranking relations are artifacts of the stratified hierarchy construction method. The retention of winner–loser pairs is crucial to techniques involving inconsistency detection, where a possible conclusion about the grammar is rejected on the basis that it is logically inconsistent with ranking information that has already been established. Inconsistency detection, used in this way, requires that the learner retain knowledge about which ranking relations have in fact been established, as opposed to those resulting from biases imposed by stratified hierarchical form. Inconsistency detection, and its role in learning, are discussed at length in Chapter 6.

5.4.4 MRCD step by step
The use of error detection in learning requires that the learner have a hypothesis it can parse with at each point, including at the outset of learning. With MRCD, the content of the initial hypothesis is already defined: it is the result of applying RCD to an empty list of winner–loser pairs. With no losers to possibly prefer, RCD’s default “as high as possible” bias puts every constraint at the top, resulting in a monostratal hierarchy, shown in (5.34). This is hardly a “sufficient” ranking hierarchy: it contains no domination relations between constraints and thus does not resolve any conflicts that might arise between the constraints with respect to the data. But, as a stratified hierarchy, it is possible to parse with it, nevertheless.

(5.34) \{WSP, \text{Ident}[\text{stress}], \text{MainLeft}, \text{MainRight}, \text{Ident}[\text{length}], \text{NoLong}\}

The rest of this section illustrates the operation of MRCD, using a sequence of winners (the observed forms) from language L20 as data. The data used in the illustration are given in (5.35).

(5.35) The data (winners) for the illustration of MRCD (in sequence)


Each subsection shows the addition of one winner–loser pair to the support. The winner being processed is given, and then the currently “optimal” candidates,
that is, the ones that are generated for the winner’s input with respect to the constraint hierarchy currently produced by RCD. The updated support is then shown; the last row of the support table contains the winner–loser pair just added to the support. The constraint columns of the support table reflect the prior constraint hierarchy, so that the constraint conflict triggering the construction of the new winner–loser pair can be directly observed. The constraint hierarchy shown next is the result of applying RCD to the updated support. Finally, an indication is given as to whether the winner is optimal given the learner’s new constraint hierarchy. In this illustration, each winner is rendered optimal after the addition of one new winner–loser pair; in general, it is sometimes necessary to add more than one winner–loser pair for a winner before it becomes optimal.

5.4.4.1 First pair
Winner (observed): /pá-ka/ [páka]

Optimal: [páka], [paká] (unresolved conflicts: ML, MR, IDENT[stress])

Support:

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>NoLong</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>IDENT[stress]</th>
</tr>
</thead>
<tbody>
<tr>
<td>pá-ka</td>
<td>páka</td>
<td>paká</td>
<td></td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5.36) \{WSP, IDENT[stress], MAIN[LEFT], IDENT[length], NoLong\} \gg \\
\{MAIN[RIGHT]\} \\
/páka/ [páka] is now optimal.

5.4.4.2 Second pair
Winner (observed): /pa-ká/ [paká]

Optimal: [páka], [paká] (unresolved conflicts: ML, IDENT[stress])

Support:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pá-ka</td>
<td>páka</td>
<td>paká</td>
<td></td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td>L</td>
</tr>
<tr>
<td>pa-ká</td>
<td>paká</td>
<td>páka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>W</td>
</tr>
</tbody>
</table>

(5.37) \{WSP, IDENT[stress], IDENT[length], NoLong\} \gg \{MAIN[LEFT], MAIN[RIGHT]\} \\
/pa-ká/ [paká] is now optimal.

5.4.4.3 Third pair
Winner (observed): /pa-ká:/ [paká:]

Optimal: [paká:], [paká] (unresolved conflicts: IDENT[length], NoLong)
5.4 Selecting winner–loser pairs

Support:

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>NoLong</th>
<th>Ident[stress]</th>
<th>Ident[length]</th>
<th>ML</th>
<th>MR</th>
</tr>
</thead>
<tbody>
<tr>
<td>pá-ka</td>
<td>páka</td>
<td>paká</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pa-ká</td>
<td>paká</td>
<td>páka</td>
<td></td>
<td></td>
<td>W</td>
<td></td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>pa-ká:│ paká: │ paká:│ L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5.38) \(\{\text{WSP, Ident[stress], Ident[length]}\} \gg \{\text{MainLeft, MainRight, NoLong}\}\)

/pá-kaː/ [pakáː] is now optimal.

5.4.4.4 Fourth pair

Winner (observed): /pá-kaː/ [páka]

Optimal: [páka], [pákaː], [pakáː] (unresolved conflicts: WSP, Ident[stress], Ident[length])

Support:

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>Ident[stress]</th>
<th>Ident[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>pá-ka</td>
<td>páka</td>
<td>paká</td>
<td></td>
<td>W</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>pa-ká</td>
<td>paká</td>
<td>páka</td>
<td></td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>pa-ká:│ paká: │ paká:│ W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pá-ka:│ paká: │ paká: │ W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5.39) \(\{\text{WSP, Ident[stress]}\} \gg \{\text{MainLeft, MainRight, Ident[length]}\} \gg \{\text{NoLong}\}\)

/pá-kaː/ [páka] is now optimal.

5.4.4.5 Fifth pair

Winner (observed): /pá-kaː/ [páka]

Optimal: [páka], [paká] (unresolved conflicts: MainLeft, MainRight)

Support:

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>Ident[stress]</th>
<th>Ident[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>pá-ka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>pa-ká</td>
<td>paká</td>
<td>páka</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pa-ká:│ paká: │ paká:│ W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pá-ka:│ paká: │ paká: │ W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pá-ká</td>
<td>páka</td>
<td>paká</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5.40) \(\{\text{WSP, Ident[stress]}\} \gg \{\text{MainLeft, Ident[length]}\} \gg \{\text{MainRight, NoLong}\}\)

/pá-kaː/ [páka] is now optimal.
Learning phonotactics

5.4.4.6 Last pair

Winner (observed): /pá-ká:/ [páka]

Optimal: [páka], [paká:] (unresolved conflicts: MainLeft,
Ident[length])

Support:

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>Ident[stress]</th>
<th>Ident[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>pá-ka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pa-ká</td>
<td>paká</td>
<td>páka</td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pa-ká:</td>
<td>paká:</td>
<td>paká</td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pá-ká</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pá-ká:</td>
<td>paká:</td>
<td>paká</td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5.41) \{WSP, Ident[stress]\} \gg \{MainLeft\} \gg \{Ident[length],
MainRight\} \gg \{NoLong\}

/pá-ká:/ [páka] is now optimal.

The support now consists of the same set of winner–loser pairs as was given in (5.6). The derived constraint hierarchy is thus identical to the one in (5.13). Although not a total ranking, this stratified hierarchy is sufficient to determine language L20; none of the forms in the language will cause an error on this hierarchy.

5.4.5 Limitations of loser production via stratified hierarchies

Generating losers by using production-directed parsing with a stratified hierarchy has some attractive properties. It can, however, stop short, failing to produce a loser when informative losers still exist. The techniques described in Section 5.4.2 all focus on detecting candidates that conflict on constraints within a single stratum. They all will fail to produce a loser that is not optimal for the learner’s generated constraint hierarchy, but is optimal for a different constraint hierarchy that is also consistent with the learner’s support.

In general, the set of possible rankings consistent with a support cannot be represented as all and only the refinements of a single stratified hierarchy. This is not just a property of intermediate hierarchies constructed during learning: it can be true of the set of possible rankings fully determining a complete language. If it is required that constraint C1 dominate C2, and C3 can be in the same region of the overall hierarchy as C1 and C2, but C3 does not interact with either C1 or C2, then the three rankings are all equivalent (holding the rest of the hierarchy fixed).

(5.42) C3 \gg C1 \gg C2 \quad C1 \gg C3 \gg C2 \quad C1 \gg C2 \gg C3
These three rankings cannot be represented as refinements of a single stratified hierarchy; C3 cannot simultaneously be in the same stratum as C1 and in the same stratum as C2, because C1 and C2 cannot be in the same stratum as each other.

Generating a single stratified hierarchy for a support typically requires choosing some ranking relations over others, from among those that are consistent with the support. The bias of RCD puts every constraint as high as possible in the hierarchy. Generating losers using production-directed parsing with a hierarchy generated by RCD can be expected to be good at producing losers that would beat the winner on a consistent ranking similar to the hierarchy generated by RCD, but not as good at producing losers that would beat the winner only on a consistent ranking significantly different from the hierarchy generated by RCD.

Here is an example of how error detection with a single generated hierarchy can stop short, expressed using the Stress/Length system. The learner is initially given the winner /pá:ka/ $\rightarrow$ [pá:ka]. Given an empty support, RCD produces the monostratal hierarchy. Applying production-directed parsing to the input /pá:ka/ using the monostratal hierarchy will yield somewhat varying results depending upon the technique used for interpreting multi-constraint strata, but a plausibly produced candidate not identical to the winner is [pá:ka]. This learning error produces the winner–loser pair shown in the support in (5.43).

(5.43) The learner’s support after the first error

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>pá:ka</td>
<td>pá:ka</td>
<td>pá:ka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Applying RCD to (5.43) yields the hierarchy in (5.44).

(5.44) \{WSP IDENT[stress] IDENT[length] MAINLEFT MAINRIGHT\} $\gg$ \{NoLong\}

Production-directed parsing using (5.44) plausibly creates another learning error, the candidate [pa:ká]. This learning error produces a second winner–loser pair, resulting in the support shown in (5.45).

(5.45) The learner’s support after the second error

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>pá:ka</td>
<td>pá:ka</td>
<td>páka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pá:ka</td>
<td>pá:ka</td>
<td>pa:ká</td>
<td></td>
<td>W</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Applying RCD to (5.45) yields the hierarchy in (5.46).
Error detection will not produce any more losers for the winner /pá:ka/ \(\rightarrow [pá:ka]\) at this point. The winner will be judged the sole optimum, no matter which technique for evaluating multi-constraint strata is used. The winner is the optimal candidate under all refinements of (5.46), because it is the only candidate for input /pá:ka/ that does not violate any of the constraints in the top stratum.

There are other hierarchies consistent with the support in (5.45), however. One such hierarchy is the one shown in (5.47).

This hierarchy satisfies both ERCs of the support: \texttt{Ident[length]} dominates \texttt{NoLong}, and WSP dominates \texttt{MainRight}. The winner, /pá:ka/ \(\rightarrow [pá:ka]\), is not the most harmonic candidate for this ranking, losing to [paká] on \texttt{MainRight}. Candidate [paká] is an informative loser, as pairing it with the winner to form a new winner–loser pair provides information not already present in the learner’s support: \texttt{MainRight} must be dominated by either \texttt{MainLeft}, \texttt{Ident[stress]}, or \texttt{Ident[length]}. If that winner–loser pair were added to the support, then (5.47) would no longer be consistent with the support.

Obtaining all of the ranking information implicit in a winner means ending up with a support which is only consistent with rankings in which the winner wins. The support in (5.45) is consistent with rankings under which the winner is optimal, but is also consistent with rankings under which the winner is not optimal. The hierarchy in (5.47) is consistent with the support, but is significantly different from the hierarchy generated by RCD for the support. Loser selection with production-directed parsing fails to generate the informative loser [paká] in this instance, and thus stops short.

It might be tempting to consider generating a set of stratified hierarchies, the refinements of which combine to include all of the total rankings consistent with a support. Computing such a set is quite computationally demanding, however. Even a support sufficient to fully determine a language can be consistent with a number of non-identical constraint hierarchies.

One alternative, proposed by Riggle (2004), is to focus not on all of the rankings consistent with a support, but on the subset of candidates that are possible optima, or contenders. The number of actual contenders is typically a small subset of the number of possible candidates and under most circumstances
Assessing computational requirements

5.5 Assessing computational requirements

would be expected to be smaller than the set of possible rankings associated with even a fairly substantial support. Creating a separate winner–loser pair between the winner and each distinct contender ensures that all ranking information implicit in a winner has been obtained. However, the algorithm for determining the contenders for a given input can still be relatively expensive, computationally speaking. Further, the number of contenders for a winner is usually significantly larger than the key set of winner–loser pairs generated via error detection. Most significantly, this approach results in a support which grows as a function of the number of observed forms: for every winner, a separate winner–loser pair is constructed for each competing contender, even though there will be a tremendous amount of redundancy in the information contained in those winner–loser pairs. It is quite likely that further algorithmic enhancements could cut down on the actual number of retained winner–loser pairs, at the cost of yet more computational processing. For exhaustive pursuit of all ranking information implicit in a given winner, the Contenders algorithm is almost certainly superior to constructing a covering set of stratified hierarchies for a support. Nevertheless, ensuring that all possible ranking information has been obtained for a given winner may simply be too computationally expensive to insist upon, at least in the general case.

There is more to say about error detection and finding informative losers. It is possible to adjust the bias associated with RCD so as to construct a constraint hierarchy more likely to produce a learning error. Error detection becomes more complex once the learner can no longer be certain what the winner is, as has been assumed in this initial presentation. These issues will be discussed in Chapter 7.

5.5 Assessing computational requirements

5.5.1 The computational complexity of MRCD

MRCD can process winners in an on-line fashion (as they are encountered), but still make use of RCD to rederive a constraint hierarchy after each error. The learner retains a list of key winner–loser pairs that empirically support the derived hierarchy, without having to memorize everything it has encountered. Winner–loser pairs are constructed only when they will provide new ranking information.

A formal upper bound on the number of learning errors MRCD will encounter exists (Tesar 1995) and can be derived as follows. Recall the initial hierarchy employed by MRCD, (5.34). The single stratum containing all constraints can be thought of as the top stratum, and as MRCD proceeds various constraints are
demoted down to lower strata. Each learning error results in the construction of a new winner–loser pair, and each pair will result in the “demotion” of at least one constraint by at least one stratum (in the newly derived hierarchy, at least one constraint will be at least one stratum further down relative to the top stratum than it was in the prior constraint hierarchy). The furthest down a constraint can end up is the bottom stratum of a total ranking. If there are N constraints, then there will be N strata in a total ranking; the bottom constraint will be (N-1) strata below the top one. If we make the (highly unrealistic) worst-case assumption that each winner–loser pair results in the demotion of only one constraint by only one stratum, then the number of winner–loser pairs needed to derive a total ranking, where N is the number of constraints, is as derived in (5.48).

\[(N-1) + (N-2) + \ldots + 1 = \frac{N^2 - N}{2}\]

This is less than the square of the number of constraints, a computational complexity of O\((N^2)\). It has a much lower complexity than the number of possible total orderings of the constraints, which is N!. A quick comparison of the growth, with respect to the number of constraints, of the worst-case-upper bound for MRCD and the number of possible total orderings is shown in (5.49). In practice, however, the worst case for MRCD never happens and is a gross overestimate. For instance, in the illustration of MRCD above, there are six constraints, and only six winner–loser pairs are required to resolve all constraint conflicts, less than half the upper bound of fifteen.

\[(5.49)\] Comparing the complexity of worst-case MRCD with the complexity of the number of total rankings (as a function of the number of constraints).

<table>
<thead>
<tr>
<th># Constraints</th>
<th>((N^2 - N)/2)</th>
<th>N!</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>15</td>
<td>720</td>
</tr>
<tr>
<td>10</td>
<td>45</td>
<td>3,628,800</td>
</tr>
<tr>
<td>20</td>
<td>190</td>
<td>2,432,902,008,176,640,000</td>
</tr>
</tbody>
</table>

The upper bound on the number of learning errors for MRCD also serves as an upper bound on the size of a support constructed by MRCD, because a winner–loser pair is only added to the support in response to a learning error.

Another way to look at the formula for the upper bound is that it is the number of pairs of distinct constraints. The number of ways of selecting
two constraints out of N constraints (without replacement) is as shown in (5.50).

\[
\binom{N}{2} = \frac{N!}{(N-2)!(2!)} = \frac{N^2 - N}{2}
\]

Having a support this size would mean having the equivalent of a separate winner–loser pair to specify the ranking relation between each distinct pair of constraints.

5.5.2 Grammar space vs. language space

The linguistic system for stress and length described in Section 5.2 has six constraints, meaning that there are 720 possible total rankings of the constraints. Yet the typology for the system, given in Section 5.10, has only twenty-four languages. How can the two sizes be so far apart?

The space of possible total rankings is the **grammar space**, at least for present purposes (later, when we consider lexical learning, grammars will also be distinguished on the basis of lexical commitments, resulting in a much larger grammar space). The space of distinct languages that can be described by the possible grammars of the system is the **language space**. The two spaces can have different sizes precisely because two (or more) grammars can generate exactly the same language. This can happen (and quite often does) when constraints fail to interact. The failure to interact is often contingent on where the constraints are relative to other constraints. For instance, if **MainLeft** is the top-ranked constraint, then every word will have initial stress, regardless of the input. **MainRight** and **Ident[stress]** will not interact in this circumstance; no matter how they are ranked with respect to each other, they will be inactive, as **MainLeft** will always ensure that stress is initial. Under other circumstances, **MainRight** and **Ident[stress]** strongly interact, such as when **Ident[stress]** dominates **MainRight** and **MainRight** dominates **MainLeft**.

Why compare a learner’s performance to the number of possible grammars, rather than the number of possible languages? Because the space of possible grammars is the space the learner is actually searching. A learner does not represent a language extensionally, by explicitly listing all of the possible forms of the language. The fact that human languages are effectively infinite in size makes that a non-starter. A grammar is how knowledge of a language is actually represented by a learner; learning a language really means constructing a grammar for a language. Simple exhaustive search must mean exhaustive search of the grammar space.
Even if the possibilities must be represented as grammars, why not search a
space containing only one grammar for each distinct language? There are two
serious problems with this. The first problem is theoretical. Granting a learner
an explicit list of permissible grammars destroys the explanation that linguistic
theory offers for linguistic typology. Linguistic theory typically explains pat-
terns in linguistic typology by positing a space of possible grammars that arises
from the allowed combinations of basic elements of the theory. Stipulating
that a learner has an explicit list of permissible grammars takes that away: the
explanation for typology is that it is explicitly encoded into the learner. While
it might be the case that every grammar in the list must be expressible as a
combination of basic elements, there is no prediction whatsoever that every
combination is a possible language, as many combinations could simply be
arbitrarily left off of the list of grammars. The connection between the inter-
nal structure of grammars (what they are made of) and the space of possible
grammars is destroyed. Requiring the learner to compute such a list, so that
it contains exactly one grammar for each distinct language predicted by the
linguistic theory, requires a highly dubious amount of meta-analysis.

The second problem is computational. The only obvious motivation for
granting the learner a list with a single grammar for each distinct language
is to make exhaustive search less onerous. But consider the Stress/Length
system. The space of distinct languages contains twenty-four languages. While
twenty-four might be less than 720, it is still greater than the upper bound on
the number of errors required by MRCD, which is fifteen. Further, that upper
bound is a gross overestimate: the example in Section 5.4.3 actually required
six errors for learning. Six is a lot smaller than twenty-four. The space of
twenty-four languages has no obvious means of search other than exhaustively
examining all twenty-four (one grammar for each language). The space of 720
possible grammars has a sophisticated structure that allows it to be searched
much more efficiently. Typically, the upper bound on the number of errors for
MRCD will be much smaller than the number of distinct languages. While
it might seem counterintuitive, the fact is that restricting the search space
to twenty-four languages, down from the 720 possible total rankings, makes
learning harder, not easier.

The idea of restricting the learner in advance to a list of one grammar per
distinct language might look like it could be computationally beneficial, but in
fact it is harmful. There are far more than twenty-four actual human languages
in the world, and the number of possible human languages must be much larger
still. A list of all such grammars is psychologically implausible, due to the
amount of memory it would require and the amount of computation that would be needed just to process the list. The idea seems to presuppose that the learner cannot do anything more sophisticated than exhaustive search and so should go for the smallest space to search. But a learner can do much better if the space of possible grammars has a useful structure, such as with a space in which every possibility is the result of a different total order of the same constraints. While the space of possible grammars is very large, the description of the space of possible grammars is quite compact. A list of the constraints to be ranked implicitly describes the space of the possible total rankings, but requires a very small amount of memory, far less than a list of grammars with one grammar per distinct language.

In general, it is the structure of a space, not its size, that determines the efficiency with which it can be searched. If a linguistic theory provides a space of possible grammars with effective structure, then the learner needn’t be particularly concerned with how big the space is.

5.6 Restrictiveness biases

5.6.1 Restrictiveness in learning

A grammar is restrictive to the extent that it limits the inventory of possible surface forms in the language it generates. Restrictiveness is a significant issue: we expect a learner to prefer the most restrictive grammar consistent with the learning data. The most restrictive grammar consistent with the data will account for both the presence of the surface forms in the data (consistency), and the absence of as many of the surface forms not in the data as possible (restrictiveness).

Restrictiveness is particularly significant in light of the fact that language learning occurs largely on the basis of positive evidence (Brown and Hanlon 1970): children receive a robust sample of forms that are grammatical in a language, but do not receive any kind of comparably significant data that are directly indicated to be ungrammatical. To the extent that each language contains forms not present in the other languages, the lack of negative evidence needn’t be a great obstacle (so long as the distinguishing forms are sufficiently observable, i.e., occur frequently enough to be reliably included in a learner’s dataset). However, natural languages aren’t generally so cooperative: there are language patterns where one language is distinguished from a second solely by lacking some surface forms present in the second language, without containing any surface forms that the second language lacks. This phenomenon, with its
implications for learning, is sometimes called the **subset problem** (Angluin 1980, Baker 1979).

Subset relations among languages are easily illustrated with the Stress/Length system. Throughout this discussion (and the preceding literature on the topic), the term “language” is understood to narrowly refer to the set of surface forms generated by a grammar (each is the surface form for at least one grammatical representation). Consider the three grammars given in (5.51), (5.52), and (5.53), which generate the languages L24, L22, and L21, respectively.

\[(5.51) \{\text{MAINL}\} \gg \{\text{IDENT[length]}\} \gg \{\text{WSP}\} \gg \{\text{NoLONG}\} \gg \{\text{MAINR}\} \gg \{\text{IDENT[stress]}\}\]

L24 Surface forms: páka páká páká: páká:

\[(5.52) \{\text{MAINL}\} \gg \{\text{WSP}\} \gg \{\text{IDENT[length]}\} \gg \{\text{NoLONG}\} \gg \{\text{MAINR}\} \gg \{\text{IDENT[stress]}\}\]

L22 Surface forms: páka páká

\[(5.53) \{\text{MAINL}\} \gg \{\text{WSP}\} \gg \{\text{NoLONG}\} \gg \{\text{IDENT[length]}\} \gg \{\text{MAINR}\} \gg \{\text{IDENT[stress]}\}\]

L21 Surface forms: páká

All three languages restrict stress to the initial syllable. Language L24 permits long vowels to appear in both stressed and unstressed syllables. L22 permits long vowels to appear only in stressed syllables. L21 does not permit long vowels on the surface in any position. The three languages stand in subset relations, with L22 a subset of L24, and L21 a subset of both L22 and L24. The grammar for L21 is more restrictive than the grammar for L22; L21 contains only a subset of the surface forms in L22. Similarly, the grammar for L22 is more restrictive than the grammar for L24; L22 contains only a subset of the surface forms in L24.

Suppose the learner is faced with a dataset consisting of \{páká\}. This form is in the surface inventory of all three languages; it could have been generated by any of the three grammars. There is no negative evidence of the form *páká:, explicitly indicating that long vowels are not permitted in unstressed position. The only indication in the data distinguishing the three grammars is the lack of other forms predicted by the other grammars. To reliably distinguish the three languages, the learner needs to address, in some fashion, what could be generated but is absent. By selecting the most restrictive grammar consistent with the data, a learner will implicitly account for what is absent as well as what is present. Out of these three grammars, the most restrictive one consistent with the data is the grammar for L21.
5.6 Restrictiveness biases

Suppose instead that the learner is faced with a dataset consisting of \{páka pá:ka\}. Language L21 is inconsistent with the data; the data include the surface form pá:ka, which is not part of L21. While L21 is more restrictive, it is inadequate for what is present. Languages L24 and L22 are consistent with the data. The learner’s preference should thus be for the grammar for L22, as it is the more restrictive of the two.

If the learner could be certain that their data set contained every possible grammatical form, the issue would be less significant: the lack of even one form in the data predicted by a grammar could be taken as an indication of inconsistency between that grammar and the data. But no such certainty is forthcoming; even adult speakers admit as grammatical some surface forms that they have never previously encountered. Language learners select grammars that generate some unobserved forms but reject others. Clearly, the dataset itself does not provide an overt distinction between some forms it lacks and other forms it lacks. The challenge to the learner is to make the right distinctions, to generalize correctly.

In selecting the most restrictive grammar consistent with the data, generalization is determined by the space of possible grammars under consideration by the learner. Suppose that our learner is faced with the dataset \{pá:ka pá:ka\}. Languages L21 and L22 are inconsistent with this dataset: neither of them contains the form pá:ka:. Thus, the learner will select the grammar for L24. This entails a commitment to the grammaticality of forms pá:ka and pá:ka:, not because they are present in the dataset (they are not), but because both are also generated by the grammar for L24. There is no grammar generating only \{pá:ka pá:ka\}, nor one generating only \{pá:ka pá:ka pá:ka\}. The grammaticality of pá:ka: entails the grammaticality of pá:ka:, a consequence of the space of grammars considered by the learner.

5.6.2 Phonotactic learning

In Optimality Theory, the inventory of surface forms generated by a grammar consists of the set of surface forms that are generated for at least one possible input. The principle of Richness of the Base holds that the set of possible inputs is universal, and thus is not subject to language-specific manipulation. Cross-linguistic variation in grammatical surface forms is due solely to cross-linguistic variation in constraint rankings.

Surface form restrictiveness in Optimality Theory can be illustrated with the three grammars given in (5.51), (5.52), and (5.53). The maps defined by these three grammars are given in (5.54), (5.55), and (5.56). A row of a table shows all of the inputs that map to a particular output.
The map for L24, defined by the grammar in (5.51)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>/paka/ /páka/ /paká/</td>
<td>[páka]</td>
</tr>
<tr>
<td>/paka:/ /páka:/ /paká:/</td>
<td>[páka:]</td>
</tr>
</tbody>
</table>

The map for L22, defined by the grammar in (5.52)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
</table>

The map for L21, defined by the grammar in (5.53)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>all sixteen possible inputs</td>
<td>[páka]</td>
</tr>
</tbody>
</table>

The maps for all three languages consist of sixteen input–output pairs, one for each possible input. Neutralization, in which multiple inputs are mapped to the same output, occurs in each of the three languages. Restrictiveness concerns the set of outputs in each map. L21 is more restrictive than L22 because the set of outputs for L21 is a subset of the set of outputs for L22.

If a learner had access to the entire map for a language, or even to entire selected input–output pairs, the issue of restrictiveness would not be nearly so onerous. The observation of a grammatical candidate like /pá:ka/ → [pá:ka] would be sufficient to indicate to the learner that it was dealing with language L21, rather than L24 or L22, where /pá:ka/ maps to [pá:ka]. In other words, life would be much easier if the learner could directly observe the input and the precise disparities resulting in a surface form. However, neither of these things is directly available to the learner. The learner observes the surface forms; the input and the input–output correspondence must be inferred as part of learning.

The full problem of determining inputs and maps involves the observation of morphemes in different morphological contexts and the determination of underlying forms for morphemes. However, it has been suggested that the inference of inputs can be finessed early in learning, during what has been labeled the phonotactic learning stage (Hayes 2004, Prince and Tesar 2004). The idea is that, prior to the acquisition of more sophisticated morphological knowledge, the learner can nevertheless make progress by assuming an input
that is “identical” to the observed surface form. While that may not be the correct input for the actual observed word, the learner should nevertheless be able to learn something legitimate about the ranking from that mapping. Underlying this is a presumption of map idempotency: for each grammatical surface form, there is a set of inputs that maps to it, and one of those inputs should be the input that is identical to the surface form.

The languages L24, L22, and L21 above are all idempotent. In all three languages, the surface form [páka] is mapped to by the identical input /páka/ (along with some other, non-identical inputs). In L24 and L22, the surface form [pá:ka] is mapped to by the identical input /pá:ka/. If presented with the two surface forms [páka] and [pá:ka], a learner in the phonotactic learning stage would create the mappings /páka/ → [páka] and /pá:ka/ → [pá:ka] and try to find the most restrictive grammar consistent with those mappings. If it were the case that the specific word that was observed as the surface form [páka] in fact had an input representation of /páka:/, with an underlyingly long second syllable, the learner would have to later draw upon knowledge of morpheme identity and alternation-based observations to infer that the second syllable should be long underlyingly, but that would not invalidate the information about the grammar that resulted from the mapping /páka/ → [páka], provided that the target map is idempotent.

The phonotactic learning problem can be stated as follows. Given a set of grammatical surface forms, find the grammar that allows all of the mappings constructed by mapping to each surface form from its identical input (data consistency), and that generates the fewest other surface forms (most restrictive). It should be noted that the phonotactic learning problem does not always have a unique solution. One reason is that different entire languages can have identical inventories of outputs; the languages are distinguished by which inputs map to which of the same outputs. See Section 5.8 for further discussion.

5.6.3 Language subsets vs. grammar subsets

When discussing restrictiveness and subset relations, it is important to be clear about precisely which objects are said to enter into such relations. In the example of the previous section, the objects literally standing in subset relations are sets of surface forms: the set of surface forms for the grammar in (5.52) is a subset of the set of surface forms for the grammar in (5.51). In relation to this, it is natural to speak of the grammar in (5.52) as being more restrictive than the grammar in (5.51).

That is a very different proposition from a claim to the effect that one grammar is literally a subset of another grammar. Such a claim requires, among other things, that grammars be interpretable in some sense as sets, such that it
is possible for one grammar to be a subset of another. Such a thing is certainly possible. For instance, if a linguistic theory posited that the possible surface segments are listed extensionally in a flat set in the grammar, then it could literally be the case that the segment inventory portion of one grammar could be the subset of the segment inventory portion of another grammar. This is a separate matter from subset relations among sets of surface forms. If the grammar with the subset segmental inventory nevertheless has a less restrictive syllable form inventory, then neither grammar’s set of surface forms would be a subset of the other’s.

The discussion in the rest of this book focuses largely on Optimality Theory and presumes the principle of Richness of the Base. In a theory of this sort, subset relations are only found between sets of surface forms. There is no obvious sense in which one grammar is the subset of another. Distinct grammars have distinct rankings of exactly the same constraints. Further, it is not the case that the set of grammatical candidates for one grammar can be a subset of the grammatical candidates of another. Given Richness of the Base, every grammar generates a grammatical candidate\(^5\) for each input, and the space of inputs is universal. For any given input, either two grammars generate the same grammatical candidate containing that input, or else each generates a grammatical candidate not generated by the other. The restrictiveness is only expressed as subset relations when candidates with identical overt forms are collapsed together.

The different kinds of objects available for comparison can be a source of confusion. Hale and Reiss (2008: 30–31) state a position in opposition to what they call the standard view of the subset principle. The description they give of the standard view is one of comparison of sets of surface forms. However, the arguments they provide in support of their opposition are arguments against subset relations between grammars. The latter is intentional: “We are assuming the I-language approach of Chomsky (1986), which considers a grammar to be internal, individual and intensional. The grammar is the language under this view, and thus the language is a computational system, not a set of sentences” (emphasis is original). With respect to learning, this confuses the target concept of learning with the data used by learners. The target concept is a grammar, but the data are not grammars. Language learners do not directly observe computational systems, they observe utterances, regardless of how one chooses to define the term “language.” Learners interpret the utterances they hear as formal objects (surface forms) that are generated by grammars, not as grammars themselves.

\(^5\) Or candidates, if ties occur between candidates with identical constraint violation profiles.
A focus of the arguments of Hale and Reiss are theories of acquisition in which learners begin with grammars containing few if any primitive elements (such as segmental inventories, or distinctive feature inventories) and then over time add additional elements to the grammars themselves, with a resulting expansion in the set of surface forms that can be generated. Without endorsing or rejecting their position with respect to such theories of acquisition, it should be pointed out that subset relations among sets of surface forms do not require subset relations between the grammars that generate those sets, and thus arguments against subset relations between grammars do not constitute complete arguments against viewing restrictiveness in terms of subset relations between sets of surface forms.

In the linguistic theory and the learning theory pursued in this chapter and the next three, there are no literal subset relations between grammars themselves. Every grammar is presumed to have the identical inventory of representational primitives (the same constraints, the same inventory of phonological features, the same \texttt{Gen} function). Subset relations only hold between sets of surface forms generated by grammars. The relative restrictiveness of grammars results from the rankings of the constraints in each, not from differing inventories of representational primitives.

### 5.6.4 Biased Constraint Demotion

#### 5.6.4.1 RCD and restrictiveness

A bias towards the most restrictive grammar consistent with the data is a desired consequence, not a proposal for how to achieve that consequence. It is unrealistic to assume that a learner has prior knowledge of all pairwise restrictiveness relations between grammars, at least in any simple extensional sense: a list of pairs of grammars standing in subset relations might reasonably be expected to have a size on the order of the square of the number of possible grammars, and for realistic theories the number of possible grammars itself is already too large for all grammars to be listed simultaneously. One is forced to the conclusion that the learner is able to calculate, either exactly or to some degree of estimation, the restrictiveness relations among grammars, as a part of achieving a bias towards greater restrictiveness.

RCD’s bias towards “every constraint as high as possible” does not in general achieve greater restrictiveness. The reason lies in the differences between markedness and faithfulness constraints. Markedness constraints by their nature prefer some outputs to others, regardless of the input. When active, they tend to restrict output inventories, in favor of the outputs they prefer. Conventional faithfulness constraints by their nature prefer the preservation of underlying
distinctions in outputs. When active, they tend to expand output inventories, by preventing neutralization of the input distinctions they target. Ranking markedness constraints as high as possible can be expected to loosely correlate with greater restrictiveness, while ranking faithfulness constraints as high as possible can be expected to loosely correlate with lesser restrictiveness.

The inputs used in phonotactic learning exacerbate the situation. By adopting input forms that are fully faithful to the outputs for purposes of phonotactic learning, the learner ensures that the winners it constructs will never be dispreferred by faithfulness constraints. The fully faithful winners won’t violate any faithfulness constraints,6 so no competitor will ever do better on any of the faithfulness constraints. Because none of the faithfulness constraints will ever prefer a loser in phonotactic learning, RCD will rank all of them at the top of its stratified hierarchy, which will tend toward the least restrictive hypothesis, rather than the most restrictive one.

An intuitive response to this observation is to suggest a change in bias. Instead of ranking all constraints as high as possible, rank markedness constraints as high as possible while ranking faithfulness constraints as low as possible. The connection between markedness dominating faithfulness and restrictiveness has been made repeatedly (Bernhardt and Stemberger 1998, Demuth 1995, Gnanadesikan 2004, Leveît 1995, Sherer 1994, Smolensky 1996a, Smolensky 1996b, van Oostendorp 1995). The connection can be illustrated with the same grammars used above, repeated here.

\[(5.51) \{\text{MainL}\} \gg \{\text{Ident[length]}\} \gg \{\text{WSP}\} \gg \{\text{NoLong}\} \gg \{\text{MainR}\} \gg \{\text{Ident[stress]}\}\]

L24 Surface forms: páka pá:ka pá:ka: pá:ka:

\[(5.52) \{\text{MainL}\} \gg \{\text{WSP}\} \gg \{\text{Ident[length]}\} \gg \{\text{NoLong}\} \gg \{\text{MainR}\} \gg \{\text{Ident[stress]}\}\]

L22 Surface forms: páka pá:ka

\[(5.53) \{\text{MainL}\} \gg \{\text{WSP}\} \gg \{\text{NoLong}\} \gg \{\text{Ident[length]}\} \gg \{\text{MainR}\} \gg \{\text{Ident[stress]}\}\]

L21 Surface forms: páka

The key faithfulness constraint to focus on is \text{Ident[length]}. In the least restrictive grammar, the grammar for L24, \text{Ident[length]} dominates both \text{WSP} and \text{NoLong}. In the grammar for L22, \text{Ident[length]} dominates \text{NoLong}

6 Or will have minimal violation of a faithfulness constraint, in the case of a faithfulness constraint that cannot be completely satisfied.
but is dominated by WSP. In the grammar for L21, the most restrictive grammar, \texttt{Ident[length]} is dominated by both WSP and \texttt{NoLong}.

5.6.4.2 Estimating the restrictiveness of grammars: the r-measure

When considering the ranking of one faithfulness constraint within a fixed hierarchy of markedness constraints, as above, the notion of greater or lesser markedness dominating faithfulness can seem straightforward. In the general case, things aren’t so obvious. Consider two hierarchies in (5.57) and (5.58), where the M# constraints are markedness constraints, and the F# constraints are faithfulness constraints. Which hierarchy has a greater degree of markedness dominating faithfulness?

(5.57) \( M1 \gg F1 \gg M2 \gg M3 \gg F2 \gg F3 \)

(5.58) \( M1 \gg M2 \gg F1 \gg F2 \gg F3 \gg M3 \)

The r-measure ("r" for "restrictiveness") was proposed by Prince and Tesar (2004) as a way of quantifying the degree of markedness dominating faithfulness.\footnote{Tesar sometimes refers to this as “mark-over-faith-edness.”} The r-measure of a stratified hierarchy is the sum of the number of markedness constraints dominating each faithfulness constraint. The hierarchy in (5.57) has an r-measure of \( 1 + 3 + 3 = 7 \). The hierarchy in (5.58) has an r-measure of \( 2 + 2 + 2 = 6 \). If we adopt the r-measure as an estimate of restrictiveness, then the hierarchy in (5.57) would be expected to be more restrictive than the hierarchy in (5.58).

A significant property of the r-measure is that it is a property of grammars, and one that is easily computed. This stands in stark contrast to the futility of attempting to directly compute the relative restrictiveness of the languages generated by two grammars, at least in the general case. The grammars of real interest generate infinite languages; it is not computationally feasible to explicitly list all of the grammatical forms in one such language, let alone compare two such languages on a form-by-form basis. To address restrictiveness at all, a learner must be able to contend with it via properties of grammars. The r-measure is such a property.

The r-measure is not a perfect characterization of restrictiveness. It is not difficult to construct cases where grammars with differing restrictiveness have the same r-measure, and it is possible to construct cases in which a more restrictive grammar has a lower r-measure than a less restrictive grammar consistent with the same data (Prince and Tesar 2004). However, it does correlate reasonably...
well with restrictiveness, and does so on the basis of a concept that is fundamental to Optimality Theory, the interaction of markedness and faithfulness constraints.

5.6.4.3 A restrictiveness bias for RCD

The ability to estimate the restrictiveness of different grammars does not by itself provide a tractable solution to phonotactic learning. During phonotactic learning, there will likely be many different grammars consistent with the data the learner has observed to that point. Trying to list out all of the consistent grammars, and compute and compare the r-measures of all of them, will be computationally infeasible. Even if only a few grammars are consistent with the data, identifying all of them can be computationally expensive. RCD is fast because it does not attempt to identify all grammars consistent with a set of winner–loser pairs; it only comes up with a particular one, the one in which each constraint is ranked as high as possible.

The Biased Constraint Demotion algorithm (Prince and Tesar 2004), or BCD, is a modified version of RCD that changes the bias toward selecting the grammar with the highest r-measure (see also Hayes (2004), who independently proposed a very similar algorithm). Instead of constructing the grammar with every constraint ranked as high as possible, it tries to construct the grammar with the markedness constraints ranked as high as possible, and the faithfulness constraints ranked as low as possible. The key modification is in the decision of which constraints to place into the hierarchy at a given step. RCD, at each step, puts all available constraints (those preferring no losers) into the next stratum of the hierarchy. BCD, by contrast, puts only available markedness constraints into the stratum; if no markedness constraints are available, then it will select the faithfulness constraint that allows the most markedness constraints to become available.

The differences between RCD and BCD can be illustrated using phonotactic data from language L22, described in (5.52). The two winner–loser pairs are shown in (5.59). As this is phonotactic learning, the input is identical to the output for each pair.

(5.59) Phonotactic learning winner–loser pairs for language L22

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td></td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>p:áka</td>
<td>p:áka</td>
<td>páka</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>L</td>
</tr>
</tbody>
</table>
When RCD is applied to this list of winner–loser pairs, it places every constraint that does not prefer a loser into the top stratum. That is every constraint except \textsc{MainRight} and \textsc{NoLong}. Both winner–loser pairs are then accounted for by constraints that have just been ranked: the first pair by \textsc{Ident}[stress] or \textsc{MainLeft}, and the second pair by \textsc{Ident}[length]. The resulting constraint hierarchy is shown in (5.60).

\begin{equation}
\{\text{WSP} \textsc{MainLeft} \textsc{Ident}\langle\text{stress}\rangle \textsc{Ident}\langle\text{length}\rangle\} \gg \{\textsc{MainRight} \textsc{NoLong}\}
\end{equation}

This hierarchy is consistent with the winner–loser pairs, but it is not the most restrictive. It isn’t even a complete grammar; it doesn’t resolve all conflicts. Neither of \textsc{Ident}[length] and WSP dominates the other, so the grammaticality of long vowels in unstressed position is unresolved. Further, \textsc{Ident}[stress] dominates \textsc{MainRight} and has no domination relation with \textsc{MainLeft}, leaving the prospect of variable main stress unresolved. The lack of restrictiveness is reflected in the r-measure of this hierarchy, which is 0. Both faithfulness constraints are in the top stratum, and aren’t dominated by any markedness constraints. The input matches the output for both winners, so they will have no faithfulness violations, and it is not possible for faithfulness constraints to prefer any losers. For phonotactic learning, RCD will always rank all faithfulness constraints at the top, nearly the opposite of enforcing restrictiveness.

When BCD is applied to the same list of winner–loser pairs, it recognizes that every constraint except \textsc{MainRight} and \textsc{NoLong} is available to be ranked. Among the available constraints, it determines that WSP and \textsc{MainLeft} are markedness constraints, while \textsc{Ident}[stress] and \textsc{Ident}[length] are faithfulness constraints. Because at least one markedness constraint is available, BCD ranks the available markedness constraints, but does not (yet) rank any of the faithfulness constraints. The partial hierarchy at the end of the first pass is shown in (5.61).

\begin{equation}
\{\text{WSP} \textsc{MainLeft}\}
\end{equation}

The ranking of \textsc{MainLeft} in the top stratum accounts for the first winner–loser pair, because \textsc{MainLeft} prefers the winner in that pair. That leaves the second pair to be accounted for, shown in (5.62) (only columns for constraints that have not yet been ranked are shown).

\begin{equation}
\text{Remaining winner–loser pairs after the first pass of BCD}
\end{equation}

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>\textsc{Ident}[stress]</th>
<th>\textsc{Ident}[length]</th>
<th>MR</th>
<th>\textsc{NoLong}</th>
</tr>
</thead>
<tbody>
<tr>
<td>pą:ka</td>
<td>pą:ka</td>
<td>pąka</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The postponement of placing faithfulness constraints in the hierarchy has already, on the first pass, ensured a higher r-measure: both faithfulness constraints are guaranteed to be dominated by at least two markedness constraints, WSP and MAINLEFT, in the resulting hierarchy.

On the second pass of BCD, the constraints available to be ranked are IDENT[stress], IDENT[length], and MAINRIGHT. Of these three, MAINRIGHT is a markedness constraint, so it is placed next into the hierarchy. However, MAINRIGHT does not prefer the winner in the remaining pair, and thus cannot account for it; the remaining winner–loser pair remains after the second pass. The hierarchy after the second pass is given in (5.63).

(5.63)  \{WSP MAINLEFT\} \gg \{MAINRIGHT\}

On the third pass of BCD, the constraints available to be ranked are IDENT[stress] and IDENT[length]. Neither of these is a markedness constraint, so now the learner is faced with the task of deciding which faithfulness constraint to rank. Maximizing the r-measure suggests ranking as few faithfulness constraints as necessary to make another markedness constraint available for ranking. In this case, IDENT[length] prefers the winner in the remaining winner–loser pair, while IDENT[stress] does not; IDENT[length] is said to be active in this case, while IDENT[stress] is inactive. Ranking IDENT[length] accounts for the winner–loser pair and makes NOLONG available for ranking; IDENT[length] “frees up” NOLONG. BCD chooses to place IDENT[length] alone into the hierarchy next, because it is sufficient to free up one markedness constraint, while IDENT[stress] alone frees up no markedness constraints.

The hierarchy after the third pass of BCD is given in (5.64).

(5.64)  \{WSP MAINLEFT\} \gg \{MAINRIGHT\} \gg \{IDENT[length]\}

At this point, no winner–loser pairs remain unaccounted for, and both remaining constraints are available for ranking. Because NOLONG is a markedness constraint while IDENT[stress] is a faithfulness constraint, NOLONG is ranked first, consistent with the goal of maximizing the r-measure. The final constructed hierarchy, after the fifth pass of BCD, is given in (5.65).

(5.65)  \{WSP MAINLEFT\} \gg \{MAINRIGHT\} \gg \{IDENT[length]\} \gg
\{NOLONG\} \gg \{IDENT[stress]\}

This hierarchy has an r-measure of $2 + 4 = 6$, much higher than the 0 r-measure of (5.60) that RCD produced. It is also more restrictive, generating the language L22 described in (5.52). MAINLEFT dominates IDENT[stress] and MAINRIGHT, enforcing the phonotactic pattern (implicit in the data) that
main stress will not appear on non-initial short vowels. WSP and MainLeft dominate Ident[length], enforcing the phonotactic patterns that long vowels will not appear in unstressed position, and that main stress will not appear on non-initial long vowels. Ident[length] does dominate NoLong, so stressed vowels can be long or short, as required by the data. This is the most restrictive grammar consistent with the data.

The r-measure of the hierarchy constructed by BCD, (5.65), is higher than the r-measure of the hierarchy that would have been constructed if Ident[stress] had been placed in the hierarchy on the third pass. Because Ident[stress] is inactive, it would not free up any markedness constraints. Thus, on the next pass, the learner would be obligated to place the other faithfulness constraint, Ident[length], into the hierarchy, ultimately resulting in the hierarchy in (5.66), with an r-measure of 4. Because Ident[stress] does not prefer any winners, there is no empirical motivation to rank it anywhere other than the bottom; ranking it higher simply creates the potential for it to preserve input contrasts that aren’t attested in the data, resulting in a larger output inventory and a less restrictive grammar.

\[(5.66) \{WSP \text{ MainLeft}\} \gg \{\text{MainRight}\} \gg \{\text{Ident[stress]}\} \gg \{\text{Ident[length]}\} \gg \{\text{NoLong}\}\]

BCD can construct more restrictive grammars than RCD, but it is not perfect, in two ways. First, as discussed above, the r-measure is not a perfect reflection of restrictiveness. Second, BCD is not guaranteed to return the hierarchy with the highest possible r-measure consistent with the data. For detailed discussion of this, see Prince and Tesar 2004.

Further, exhaustive pursuit of the highest possible r-measure can get computationally expensive, somewhat blunting one of the most attractive properties of RCD. The culprit is the need to decide which faithfulness constraints to rank. RCD is fast because, at each ranking step, it ranks all available constraints. BCD is similarly efficient so long as markedness constraints are available; it ranks all available markedness constraints. But when forced to rank a faithfulness constraint, optimizing the r-measure drives the learner to rank as few of the available faithfulness constraints as possible. This involves additional computation, as the learner checks each faithfulness constraint to see how many markedness constraints it would free up if it were ranked next.

Both RCD and BCD return a constraint hierarchy consistent with a list of winner–loser pairs. RCD always returns the hierarchy with every constraint ranked as high as possible. BCD attempts to find a hierarchy with every markedness constraint ranked as high as possible, and every faithfulness constraint
ranked as low as possible. The restrictiveness bias of BCD makes it useful when restrictiveness of the constructed grammar is a concern.

5.6.5 Enforcing restrictiveness in phonotactic learning

5.6.5.1 Enforcing restrictiveness with BCD

BCD can be used in place of RCD to add a restrictiveness bias to MRCD. The phonotactic winner–loser pairs for L22 in (5.59) can be obtained using MRCD with BCD. L22 has two surface forms, [páka] and [pá:ka].

The initial hierarchy for learning will be the hierarchy derived from an empty list of winner–loser pairs. Because BCD is being employed, the initial hierarchy will have all of the markedness constraints in the top stratum, and all of the faithfulness constraints in the stratum below, as shown in (5.67).

(5.67) \{WSP\ NoLong MainLeft MainRight\} \gg \{Ident[stress] Ident[length]\}

If the first form processed by the learner is [páka], the learner will adopt /páka/ as the input. If the CTie criterion is used, then the learner will generate [páka] and [paká], because the two conflict on the top stratum, one preferred by MainLeft and the other preferred by MainRight. The first winner–loser pair is constructed by adopting the generated candidate that does not match the winner. The resulting support is shown in (2.68).

(5.68) Support after the first winner–loser pair

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>NoLong</th>
<th>ML</th>
<th>MR</th>
<th>Ident[length]</th>
<th>Ident[stress]</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>paká</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The learner now applies BCD to the support, which constructs the hierarchy shown in (5.69).

(5.69) \{WSP\ NoLong MainLeft\} \gg \{MainRight\} \gg \{Ident[stress] Ident[length]\}

The support in (5.68) is the same as the one in Section 5.4.4.1, where the first winner was also /páka/[páka]. The hierarchy generated from the support here, in (5.69), is different from the hierarchy generated in Section 5.4.4.1, (5.36), because the earlier illustration used RCD, while the present one is using BCD. In (5.36), both faithfulness constraints are in the top stratum, while in (5.69) both are in the bottom stratum (below all of the other constraints). The hierarchy in (5.69) makes candidate /páka/[páka] the sole generated candidate; it is the sole optimum on that hierarchy. The learner then considers the
second data form, [pá:ka]. Constructing the input /pá:ka/ and generating with respect to the hierarchy in (5.69) yields one candidate, [páka]. The length on the initial vowel is not faithfully realized, because \( \text{NoLong} \gg \text{Ident[length]} \). This candidate does not match the data form, so another winner–loser pair is constructed, using the generated [páka] as the informative loser. Adding this pair results in the support in (5.70).

(5.70) Support after the second winner–loser pair

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>NoLong</th>
<th>ML</th>
<th>MR</th>
<th>Ident[length]</th>
<th>Ident[stress]</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>p:ka</td>
<td>p:ka</td>
<td>p:ka</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>W</td>
</tr>
</tbody>
</table>

This is the same support as was considered in (5.59), so applying BCD to it will produce the hierarchy in (5.65). At this point, the identity candidates for both data forms are the sole optima for their respective inputs, so phonotactic learning will not produce any more learning errors. For L22, phonotactic learning is done at this point.

5.6.5.2 Restrictiveness with “hierarchy only” learning

As discussed in Section 5.4.3, Error-Driven Constraint Demotion and the Gradual Learning Algorithm both function by directly manipulating a constraint hierarchy (EDCD) or ranking values that provide the basis for a hierarchy (GLA). BCD cannot be used with these approaches, because they are not based on a support of winner–loser pairs. These “hierarchy only” approaches rely on the constraint hierarchy in use at that point as the only reflection of data previously seen, and update the hierarchy based upon the currently observed form and the prior state of the hierarchy.

An older approach to restrictiveness involves a particular choice of initial hierarchy (Smolensky 1996a, 1996b), and is sometimes called ranking conservatism, in light of its connection to a more general proposal of that name by Itô and Mester (1999). Restrictiveness is imposed on the initial hierarchy of the learner (the one adopted before any data have been processed), such that all of the markedness constraints are above all of the faithfulness constraints. For EDCD, this would typically mean a two-stratum initial hierarchy, with the markedness constraints in the first stratum, and the faithfulness constraints in the second stratum. For the GLA, some initial difference in ranking value is selected, such that the initial ranking values of the markedness constraints are that much higher than the initial ranking values of the faithfulness constraints.
The idea is that if the learner only alters rankings/ranking values to the extent necessitated by the data (the learner is “conservative” in altering the hierarchy), then the initial restrictiveness should be preserved to the extent allowable. Ranking conservatism has been criticized for being insufficient; see Prince and Tesar 2004 for further discussion.

More recently, Magri (Magri 2009, 2012) has developed an alternative update rule for approaches based on ranking values (such as the GLA) that uses not only demotion of constraints preferring losers, but also calibrated promotion of constraints preferring winners. The constraint promotion approach is guaranteed to converge, and is more successful at enforcing restrictiveness than ranking conservatism.

For present purposes, the problem with the constraint-promotion approach to enforcing restrictiveness is not its capacity for enforcing restrictiveness, but its dependence on a “hierarchy only” approach to learning. Because no support is maintained, such approaches are not capable of inconsistency detection. Inconsistency detection is an essential element of the approach to learning developed in this book, especially with respect to the learning of underlying forms.

5.6.6 Implicit representation of phonotactic restrictions

Given the label “phonotactic learning,” it would be natural to expect that the information about the grammar that is encoded most directly by the learner, in the support, expresses the phonotactic patterns that must be enforced. But phonotactic learning using winner–loser pairs with MRCD and BCD, as described above, actually does something like the reverse: it explicitly records, in the support, patterns that cannot be restricted by the grammar. Phonotactic learning makes explicit note of certain forms in the data that must be admitted in the language and implicitly (via the ranking bias in BCD) tries to restrict as many forms as possible that are not entailed by the information in the support.

None of the phonotactic ERCs derived for L22, as shown above in (5.70), contain any information about the ranking of WSP. This despite the fact that one of the strongest phonotactic restrictions reflected in the data is that long vowels only appear in stressed syllables, and the expression of that generalization in the grammar is the domination of other constraints by WSP. In this regard, WSP is in some sense a victim of its own success. In L22, WSP is never violated in grammatical surface forms. This means that winners, which use only observed grammatical surface forms, will never be dispreferred by WSP, which is correct and desirable. Because WSP is a markedness constraint that will never prefer a loser, BCD will always rank it in the top stratum, and WSP
will always dominate all of the faithfulness constraints. For WSP to prefer a winner in a constructed winner–loser pair, the loser must violate WSP. To obtain such a loser, production-directed parsing has to generate a candidate that violates WSP, meaning that it has to be working with a hierarchy that would select as optimal a candidate that violates WSP. In this particular case, that can’t happen: any candidate which violates WSP fares no better on any of the markedness constraints than a related candidate which does not violate WSP. Because the faithfulness constraints will always be dominated by at least one markedness constraint (WSP) in any hierarchy generated by BCD, the top stratum will always eliminate any candidate violating WSP as suboptimal. Because WSP starts at the top of the hierarchy, and is so successful at imposing itself on competitions, its dominance doesn’t get explicitly reflected in the support generated by phonotactic learning.

Here is the heart of the argument in more detail. WSP is violated when an unstressed syllable has a long vowel. For any candidate with a long vowel in an unstressed syllable, the candidate that has a short vowel in that unstressed syllable, but is otherwise identical, fares better on WSP and \textsc{NoLong}, and equally well on \textsc{MainLeft} and \textsc{MainRight}. There is no markedness constraint in the Stress/Length system that prefers a long vowel in an unstressed syllable, under any conditions. For the observed form \([pá:ka]\), the input /pá:ka/ is used, with \([pá:ka]\) the identified winner. Candidate \([pa:ká]\) violates WSP and could form a winner–loser pair in which WSP prefers the winner. But candidate \([pa:ká]\) won’t be generated as optimal for input /pá:ka/ for any hierarchy generated by BCD, because it will always be less harmonic than candidate \([paká]\). It is not necessary that \([paká]\) be the optimal candidate: any candidate that is more harmonic than \([paká]\) will necessarily be more harmonic than \([pa:ká]\) also. Note that \([pa:ká]\) is a possible optimum; it is not harmonically bounded. In particular, it can be optimal if \textsc{Ident}[length] dominates WSP (and \textsc{NoLong}), such as in language L5 (see the appendix in Section 5.10). But such a hierarchy will never be generated by BCD, because WSP is a markedness constraint, and it does not prefer any losers for this language, so it will always be ranked above \textsc{Ident}[length] in any hierarchy generated by BCD.

The lack of reference to WSP in the support constructed by phonotactic learning is a counterintuitive state of affairs. WSP expresses a phonotactic restriction that is strongly enforced, in fact never violated, yet “phonotactic learning” produces no winner–loser pairs that directly express that fact. Phonotactic learning only realizes the phonotactic restriction when it employs a markedness-over-faithfulness bias to produce a constraint hierarchy; then WSP is ranked at the top, by virtue of being a markedness constraint that prefers no losers.
The implicit representation of phonotactic restrictions is partly a consequence of approaching learning incrementally. An incremental learner responds to data as the data are received, making at least tentative judgments about the grammar along the way. The learner is then cautious about premature commitments: just because an unstressed long vowel hasn’t yet been observed doesn’t mean one won’t be encountered in the future. MRCD with BCD is extremely cautious in this sense: what is encoded in the winner–loser pairs is based directly on observed surface forms, that is, on what has to be included in the language. The restrictiveness component, what is to be excluded from the language, is encoded in the constructed constraint hierarchy, which is defeasible; a new hierarchy is constructed every time a new winner–loser pair is added. This approach allows the learner to freely posit hypothesized grammars at any point during learning (even before any data have been observed), with the consequence that phonotactic restrictions are tentatively conjectured, not committed to.

5.7 Phonotactic contrast

5.7.1 Contrast and the nature of phonotactic learning

Consider what is encoded most directly in phonotactic learning: the stored winner–loser pairs. For phonotactic learning, the input always matches the output of the winner, so faithfulness constraints never prefer losers. A constraint which does not prefer a loser in a set of pairs is not directly required to be dominated by any other constraint, with respect to those pairs. Thus the only constraints that are explicitly required to be dominated are markedness constraints. When markedness constraints are dominated by faithfulness constraints, the effects are typically the loosening of restrictiveness, admitting into the language marked forms that might otherwise be banned.

Recall the two phonotactic learning winner–loser pairs in (5.59). In the second pair, the loser is preferred by NoLong, while the winner is preferred by Ident[length]. The fact that NoLong prefers the loser and must be dominated means that under some circumstances, at the very least for the input /pá:ka/, the marked structure long vowel must be admitted in the language. The fact that the only potential dominator for this winner–loser pair is Ident[length], a faithfulness constraint, means that under some circumstances, at the very least for the input /pá:ka/, a contrast between underlingly short and underlingly long must be admitted in the language. Thus, this winner–loser pair is not so much an expression of what the phonotactics are as of one instance of what the phonotactics cannot be.
In the first pair of (5.59), the loser is preferred by **MainRight**, the winner is preferred by both **Ident[stress]** and **MainLeft**. The fact that **MainRight** prefers the loser and must be dominated means that under some circumstances, at the very least for the input /páka/, the marked structure of a word with non-final stress must be admitted in the language. The fact that there are two potential dominators for this winner–loser pair, one a markedness constraint and one a faithfulness constraint, means that the learner cannot tell from this winner–loser pair alone if actual contrast is involved. If the actual key domination is **Ident[stress] ≫ MainRight**, then we expect a contrast in the language (under at least some circumstances) between final and non-final stress. If the actual key domination is **MainLeft ≫ MainRight**, then the observation may be a matter of a grammar-specific preference for one marked structure over another. An obvious such case would be a language in which stress is always initial; stress is still predictable, just in a different way from a language in which (predictable) stress is always final. Such non-contrastive preferences result from the domination of one markedness constraint by another. Again, the winner–loser pair is less an expression of what the phonotactics are, and more an expression of what the phonotactics cannot be: they cannot prevent non-final stress for the input /páka/.

The phonotactic restrictiveness part of phonotactic learning comes in the hierarchy constructed in response to the winner–loser pairs. The mark-over-faith-edness bias of BCD tries to restrict the presence of contrasts in the language. The requirement that the constraint hierarchy be consistent with the winner–loser pairs means that the observed marked forms must be admitted by the grammar. To the extent that the marked forms can only be admitted by the activity of one or more faithfulness constraints, contrasts are introduced into the language. For the second pair of (5.59), the only winner-preferring constraint is the faithfulness constraint **Ident[stress]**. Thus, admitting the marked form [pá:ka] into the language brings with it the consequence that the contrasting form [páka] is also in the language. The accommodation of the marked form forces a learner to accept the presence of a contrast.

For the first pair of (5.59), the learner has a choice between two dominators, one a markedness constraint and one a faithfulness constraint. Absent the effects of other winner–loser pairs, the mark-over-faith-edness bias of BCD will choose the markedness explanation, preferring a more restrictive “choice between marked forms” to a less restrictive contrast. On the other hand, if there were another winner–loser pair in which the markedness constraint preferred the loser, then it would not be available to be placed in the hierarchy, and the
learner would have no choice but to select the faithfulness constraint and the coattendant contrast.

5.7.2 A canonical form for phonotactic contrast

The way in which winner–loser pairs collectively capture contrast information can be seen more clearly via a canonical form for phonotactic information that can express much of what can be learned during phonotactic learning. The fundamental unit of contrast is the output; for current purposes, that means the phonological word. Phonological contrast exists when two words with phonologically distinct surface realizations are both admitted in the language. Recall the surface forms of L20, repeated here in (5.71).

(5.71) páka pá:ka paká paká:

Any pair of distinct surface words entails a contrast of some sort. Just what sort, in terms of the grammar, can be seen by constructing a particular pair of winner–loser pairs and examining some joint entailments of those two pairs. For purposes of illustration, consider the two words páka and paká from L20. In keeping with phonotactic learning, we can adopt, for each surface word, an input that is identical to the surface form. We can then construct a competitor for each word consisting of the same input but with an output identical to the surface form of the other, contrasting word. These winner–loser pairs, their constraint violation profiles, and the resulting ERCs are shown in (5.72) and (5.73).

(5.72) Violation tableau and ERC for the phonotactic winner–loser pair páka ~ paká

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>WSP</th>
<th>IDENT [stress]</th>
<th>IDENT [length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>páka</td>
<td>paká</td>
<td></td>
<td>**</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>páka ~ paká</td>
<td>e</td>
<td>W</td>
<td>e</td>
<td>W L e</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(5.73) Violation tableau and ERC for the phonotactic winner–loser pair paká ~ páka

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>WSP</th>
<th>IDENT [stress]</th>
<th>IDENT [length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>paká</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>páka</td>
<td></td>
<td>**</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká ~ páka</td>
<td>e</td>
<td>W</td>
<td>e</td>
<td>L W e</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In each winner–loser pair, the winner is preferred by the faithfulness constraint $\text{Ident}[\text{stress}]$ and one of the markedness constraints on stress alignment. Each pair on its own poses both a markedness and a faithfulness solution, with restrictiveness preferring the markedness solution. We can see the consequence of combining the two winner–loser pairs by examining the fusion of the two ERCs for the winner–loser pairs (see Section 3.1.3).

(5.74) The fusion of the two contrast ERCs: all active markedness constraints come out L

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>$\text{Ident}[\text{stress}]$</th>
<th>$\text{Ident}[\text{length}]$</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>páká</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fusion</strong></td>
<td></td>
<td></td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Recall that the fusion of two ERCs is necessarily jointly entailed by the two ERCs, so if both original ERCs are true, then the fusion must be true. The fusion here mandates that both markedness constraints, $\text{MainLeft}$ and $\text{MainRight}$, be dominated by the faithfulness constraint $\text{Ident}[\text{stress}]$. The pair of surface forms together indicate that stress needs to be contrastive in at least some environments.

This kind of outcome is inevitable for any pair of ERCs constructed in this fashion. Because the input is always identical to the surface form of the winner, all faithfulness constraints will be either indifferent (not violated by the loser) or prefer the winner (violated by the loser). This will be true of both ERCs, so in the fusion of the two ERCs every faithfulness constraint which prefers the winner in one of the ERCs will receive a W in the fusion. Markedness constraints, on the other hand, could prefer the winner, prefer the loser, or be indifferent in a phonotactic winner–loser pair. Because markedness constraints only evaluate outputs, their evaluation of the surface forms will not change between the two winner–loser pairs; they will be blind to the differences in the input. Thus, any markedness constraint that prefers the winner to the loser in one pair will necessarily prefer the loser to the winner in the other pair (and vice-versa), because each pair involves the same two surface forms, just with their roles as winner and loser reversed. Thus every markedness constraint that actively distinguishes the two surface forms prefers the loser for exactly one of the two ERCs and thus will receive an L in the fusion. The end result is the conclusion that at least one of the faithfulness constraints sensitive to a distinction between the contrasting surface forms must dominate all of the
markedness constraints sensitive to distinctions between the contrasting surface forms. Seen in this way, the presence of two contrasting surface forms cannot be fully accounted for by neutralizing markedness constraints; at least one faithfulness constraint must play an active role in the explanation. Taking the fusion of the two constructed ERCs gets the markedness constraints out of the way and reveals which faithfulness constraints could possibly be responsible for the surface contrast.

In some cases, only one winner–loser pair is necessary to establish the same information. This occurs when the contrast concerns surface forms which differ with respect to values at different places along a markedness scale, provided no other markedness constraints provide conflicting evaluations. A markedness scale expresses entailment relations among forms, such that the presence in a surface form of an element that is more marked on the scale entails the grammaticality of an element that is less marked on the scale in the same environment. In our illustration system, the constraint NoLong is the grammatical realization of a markedness scale in which long vowels are more marked than short vowels. Because there are no interfering markedness constraints, if a long vowel appears in a given environment in a grammatical word, then a short vowel in the same environment will also be grammatical. In such a circumstance, the observation of the more marked element in a surface form is by itself sufficient to establish the contrast, as it entails the grammaticality of a corresponding surface form with a less marked element appropriately substituted. In L20, this can be illustrated with the word pá:ka, as in (5.75).

(5.75) Violation tableau and ERC for the phonotactic winner–loser pair pá:ka ~ páka

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>pá:ka</td>
<td>pá:ka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pá:ka</td>
<td>páka</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>pá:ka ~ páka</td>
<td>e  e</td>
<td>W</td>
<td>e  e</td>
<td>e  L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The markedness constraint distinguishing the two words, NoLong, is violated by the more marked long vowel, and thus prefers the loser when the

---

8 WSP is a markedness constraint sensitive to vowel length, but in the same direction as NoLong. WSP can only be violated by long vowels, specifically long vowels that aren’t stressed. What is “marked” by WSP is also “marked” by NoLong. In this system, WSP will not run counter to the preferences of NoLong.
more marked word is taken as the winner. The relevant faithfulness constraint prefers the winner (as always), and the winner–loser pair entails that the faithfulness constraint must dominate the markedness constraint, expressing the contrastiveness of vowel length in this environment.

There is no harm in constructing the other phonotactic winner–loser pair, with the roles of the two words reversed, but it contributes no additional information. As shown in (5.76), the word with the more marked element is now the loser, so the markedness constraint $\text{NoLong}$ prefers the winner, as does the faithfulness constraint $\text{Ident}[\text{length}]$. The resulting ERC is an instance of harmonic bounding: two constraints prefer the winner, while none of the constraints prefers the loser. There is no constraint conflict here; the winner will always be more harmonic than the loser, no matter what constraint hierarchy is employed.

(5.76) Violation tableau and ERC for the phonotactic winner–loser pair páka ~ pá:ka

Taking the fusion of the two ERCs simply returns the very same informative first ERC.

(5.77) The fusion of the two ERCs is the same as the first fusand

In the preceding two examples, the contrasting pairs of surface forms were minimal pairs (single-disparity pairs) or near-minimal pairs. páka and pä:ka only differed in a single feature, making it clear that the differing feature (and only that) was responsible for the contrast between the words. pāka and paká differ in two features, the stress features of both syllables, but at least one of those features must differ underlingly between the two forms to account for
the contrast, and the same constraint, $\text{IDENT}[\text{stress}]$, is implicated whether the underlying contrast is in the first syllable, the second syllable, or both.

While those “minimal” contrasts are easy to interpret, the canonical form for phonotactic contrasts does not require that the contrasting surface forms be anything like minimal pairs. Consider the contrasting words páka and paká, which differ in almost every feature (the only feature value in common is the length feature on the first syllable of each word). This is shown in (5.78).

(5.78) The phonotactic contrast information for páka and paká:

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>$\text{IDENT}[\text{stress}]$</th>
<th>$\text{IDENT}[\text{length}]$</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>paká:</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>páka</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td></td>
</tr>
</tbody>
</table>

Fusion       | W       | W     | L   | L                             | L                             |

The same markedness/faithfulness pattern is (necessarily) exhibited: all distinguishing markedness constraints (3 of the 4) must be dominated by at least one of the distinguishing faithfulness constraints (both of them). On its own, this pair identifies that there must be at least one underlying contrast distinguishing the two words and identifies the possible feature types, both stress and length in this case, but does not indicate whether the actual underlying contrast(s) is one of stress, length, or both. In combination with other phonotactic contrasts, more can be determined. If we combine, in a list, the fusion ERC for the phonotactic contrast in (5.78) with the fusion ERC for the phonotactic contrast in (5.74), we get the list in (5.79).

(5.79) The fusion ERCs for two phonotactic contrasts

<table>
<thead>
<tr>
<th>Contrasting Words</th>
<th>WSP</th>
<th>$\text{IDENT}[\text{stress}]$</th>
<th>$\text{IDENT}[\text{length}]$</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka paká:</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>páka paká</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Applied to the ERCs in (5.79), BCD will place WSP in the top stratum. It will then observe that $\text{IDENT}[\text{stress}]$ frees up three markedness constraints, while $\text{IDENT}[\text{length}]$ only frees up one, and on that basis place $\text{IDENT}[\text{stress}]$ into the hierarchy next, followed by the three remaining markedness constraints, with $\text{IDENT}[\text{length}]$ at the bottom.

(5.80) $\{\text{WSP}\} \gg \{\text{IDENT}[\text{stress}]\} \gg \{\text{MAINLEFT MAINRIGHT NOLENGTH}\} \gg \{\text{IDENT}[\text{length}]\}$
Notice that both ERCs contribute information. The second one makes clear that a stress contrast of some sort is necessary. This allows BCD to (at least temporarily) use a stress contrast as an account of the first ERC as well. The first ERC, however, indicates that NoLong must be dominated by one of the faithfulness constraints, information not contained in the second ERC. The second ERC narrows the choice among the faithfulness constraints by having fewer Ws, while the first ERC expands the set of affected markedness constraints by having more Ls.

Not all phonotactic ranking information can be expressed in this canonical form, because not all phonotactic ranking information is based on overt contrast. To see this, consider the language given in (5.81), in which stress is always initial.

(5.81) páka pâ:ka

Because this language has only two words, there is only a single phonotactic contrast to be constructed. This is in fact the same phonotactic contrast as shown in (5.77), yielding the ERC repeated in (5.82).

(5.82) The fusion ERC for the phonotactic contrast between páka and pâ:ka

<table>
<thead>
<tr>
<th>Contrasting Words</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka pâ:ka</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This indicates that IDENT[length] must dominate NoLong, due to the observable contrast in length. Applying BCD to this single ERC yields the constraint hierarchy in (5.83).

(5.83) \{WSP MAINLEFT MAINRIGHT\} \gg \{IDENT[length]\} \gg \{NoLong\} 
\gg \{IDENT[stress]\}

This hierarchy does not reflect all of the phonotactically available information. Specifically, the winner /páka/ → [páka] implicitly needs to be more harmonic than its competitor /pá:ka/ → [paká], but that comparison is not resolved by the hierarchy in (5.83). MRCD will uncover this when it applies error detection to the word páka using the hierarchy in (5.83): with the input /páka/, the output forms [páka] and [paká] will conflict on the constraints MAINLEFT and MAINRIGHT in the top stratum, so [paká] will be selected as an informative loser, yielding the winner–loser pair shown in (5.84).
Learning phonotactics

(5.84) Violation tableau and ERC for the phonotactic winner–loser pair páka ~ paká

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>WSP</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NO:LONG</th>
<th>IDENT[stress]</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>páka</td>
<td>paká</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>páka ~ paká</td>
<td>e</td>
<td>W</td>
<td>L</td>
<td>e</td>
<td>e</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

This pair indicates that either MAINLEFT or IDENT[stress] must dominate MAINRIGHT. BCD will prefer domination by MAINLEFT if possible, because it is a markedness constraint, and here it is possible. The represented ranking information, effectively, is that MAINLEFT dominates MAINRIGHT. This is not an instance of faithfulness dominating markedness; it is an instance of markedness dominating markedness. The phonotactic information is not the presence of a contrast; stress is not contrastive here, it is predictably initial. The phonotactic information concerns the choice between two disjoint non-contrastive patterns: one in which stress is always initial, and one in which stress is always final. The output of the winner is a surface form in the language, while the output of the loser is not a surface form of the language.

The canonical form for phonotactic ranking information is intuitively appealing. It also raises the possibility of performing phonotactic learning simply by building appropriate pairs of surface forms. There are drawbacks to computing phonotactic learning in this fashion, however. One is the decision of which pairs to construct. Given any significant list of phonologically distinct words, there will be many pairs that can be formed. The formula for the precise number is given in (5.85), where \( N \) is the number of distinct words in the list; this is the same formula for the number of pairs for a set as was given in (5.50).

\[
(5.85) \quad \binom{N}{2} = \frac{N^2 - N}{2}
\]

However, of these pairs of words, many will contain redundant ranking information. Phonotactic learning with MRCD only constructs winner–loser pairs containing novel information, information that the learner’s list of winner–loser pairs did not already possess.

Another drawback to constructing only pairs of observed surface forms is that phonotactic information regarding markedness constraints dominating other markedness constraints will be missed. As described above, such information is represented by a single winner–loser pair, where the output of the loser is not a phonotactically valid output. Pairs of phonotactically valid words on their own will not produce such winner–loser pairs.
5.8 Phonotactic information underdetermines languages

The canonical form for phonotactic ranking information given here is not intended to outline a learning algorithm. It is intended as a means of analyzing and understanding the kind of ranking information that is available phonotactically.

5.8 Phonotactic information underdetermines languages

Non-identical languages can have the same phonotactic inventories. From a purely phonotactic point of view, the languages are indistinguishable; information about morpheme identity is required to distinguish them.

Recall language L20, repeated below: the forms are given in (5.2), and a ranking generating the language is given in (5.3).

(5.2) Language L20

<table>
<thead>
<tr>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>pá:ka</td>
<td>páka</td>
<td>pá:ka</td>
</tr>
<tr>
<td>páka</td>
<td>pá:ka</td>
<td>páka</td>
<td>pá:ka</td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>páka</td>
<td>pá:ka</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>páka</td>
<td>pá:ka</td>
</tr>
</tbody>
</table>

s1 = /-ka/

s2 = /-ka: /

s3 = /-ká/ 

s4 = /-ká: /

(5.3) WSP ⇒ Ident[stress] ⇒ MainLeft ⇒ MainRight ⇒ Ident[length] ⇒ NoLong

Compare that with another language, L14, given in (5.86) and (5.87).

(5.86) Language L14

<table>
<thead>
<tr>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>pá:ka</td>
<td>páka</td>
<td>pá:ka</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>páka</td>
<td>pá:ka</td>
</tr>
<tr>
<td>paká</td>
<td>pá:ka</td>
<td>páka</td>
<td>pá:ka</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
<td>pá:ka</td>
</tr>
</tbody>
</table>

s1 = /-ka/

s2 = /-ka:/

s3 = /-ká/

s4 = /-ká:/

(5.87) WSP ⇒ Ident[length] ⇒ NoLong ⇒ Ident[stress] ⇒ MainLeft ⇒ MainRight

The first thing to notice is that both languages have the same inventory of phonotactic forms, the one shown in (5.88).

(5.88) páka paká pá:ka paká:
The second thing to notice is that the languages are not identical; some inputs surface differently in the two languages. In fact, the distinguishable inventories of suffixes have different sizes for the two languages: L14 has four distinct suffixes, while L20 has only three, with suffixes s1 and s2 neutralizing in all environments. These distinguishing properties can only be observed with the assistance of morpheme identity, awareness of the realization of the same morpheme in different contexts. Without information about morpheme identity, the two languages are indistinguishable (their phonotactic inventories are the same).

Because the phonotactic inventories are the same, the phonotactic ranking information for the two languages is the same. The phonotactic ranking information is shown in (5.89). The first ERC is the one described above for L20 in (5.77), capturing the contrast between surface forms p´aka and p´a:ka. The second ERC is the one described above for L20 in (5.74), capturing the contrast between p´aka and pak´a.

(5.89) The phonotactic ranking information for languages L20 / L14

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
<th>ML</th>
<th>MR</th>
<th>NO LONG</th>
</tr>
</thead>
<tbody>
<tr>
<td>p´aka p´a:ka</td>
<td></td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>p´aka pak´a</td>
<td>W</td>
<td></td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The phonotactic ranking information clearly indicates that faithfulness to length has to dominate the markedness constraint against long vowels, and that faithfulness to stress has to dominate both markedness constraints on stress alignment. It does not indicate how the two sets of constraints should be ranked with respect to each other, nor, as noted earlier, does it indicate how WSP should be ranked.

The phonotactic ranking information does not indicate how the two faithfulness constraints should be ranked with respect to each other, and that is a crucial part of the distinction between L20 and L14. The two faithfulness constraints conflict when underlying stress and underlying length appear on different syllables: WSP (undominated in the grammars for both languages) won’t permit both to surface faithfully. In L20, IDENT[stress] dominates IDENT[length], and such a conflict is resolved in favor of faithful realization of the underlying stress. In L14, IDENT[length] dominates IDENT[stress], and the conflict is resolved in favor of faithful realization of the underlying length.
5.8 Phonotactic information underdetermines languages

(5.90) \( r3s2 \) in L20: /páka:/ → [páka]

(5.91) \( r3s2 \) in L14: /páka:/ → [paká:]

Phonotactic learning cannot uncover domination relations between faithfulness constraints, because phonotactic learning cannot directly uncover evidence for the domination of any faithfulness constraint: when the inputs are identical to the outputs in all winners, faithfulness constraints will not be violated in the winners and thus cannot prefer any losers.

In L20, MainLeft also crucially dominates Ident[length]. This ranking is responsible for the realization of \( r1s2 \), in which the default realization of stress initially takes precedence over the preservation of length in the suffix. In L14, in which Ident[length] dominates MainLeft, \( r1s2 \) is realized with final stress: stress is attracted to the preserved length on the suffix.

(5.92) \( r1s2 \) in L20: /paka:/ → [páka]

(5.93) \( r1s2 \) in L14: /paka:/ → [paká:]

This difference in the rankings for the two languages is responsible for the difference in the number of suffix contrasts in the languages. In L20, stress preservation takes precedence over length preservation, and default stress to initial position also takes precedence over length preservation. Thus, stress will be word initial unless the suffix is underlingly stressed; without an underlying stress to pull stress to final position, stress will end up initial, either by virtue of preserving underlying initial stress or by default initial stress. Thus, length will always be neutralized in suffixes that are underlingly unstressed.

Phonotactic learning cannot uncover this sort of information either, because it crucially involves a conflict that never manifests itself on the surface. The input for \( r1s2 \), /paka:, is not a valid surface form in either language: it has no main stress and has a long unstressed syllable. Because phonotactic learning only contemplates inputs that are identical to valid surface forms, this situation will not be directly considered by phonotactic learning.

In this case, phonotactic learning is able to explicitly represent that there is a stress contrast somewhere. It is also able to explicitly represent that there is a length contrast somewhere. What it cannot explicitly represent is the precise restriction on those contrasts: “where” the contrasts are preserved, and “where” the contrasts are neutralized. It implicitly represents (via the ranking of WSP at the top of the ranking) that long vowels must be stressed on the surface, but cannot determine if underlying stress contrasts are neutralized in the presence
of long vowels, or if underlying vowel length is neutralized in the absence of stress.

Learning these other aspects of the grammar requires more information. Specifically, what is required is paradigmatic information, the morpheme-based relationships between words. Paradigmatic information involves knowledge of morpheme identity: determining that the surface word páka consists of two morphemes (as opposed to one, or three), and that those morphemes are root r1 and suffix s1. Ultimately, human learners must infer this information from less direct observations about where and how utterances are used, but that problem will not be addressed in this book. Even assuming (for the present) that children can determine the morphological composition of the words they are analyzing, there is still a significant further task in learning the non-phonotactic ranking information and the lexicon of underlying forms, based upon morphologically segmented surface forms. Learning with paradigmatic information is the topic of the next chapter.

One brief note of clarification: when I claim that learners have determined the identity of a morpheme in a word, and label it as r1, I don’t mean that the text label “r1” is literally assigned by every learner of the language. That label is a stand-in for whatever resource speakers use to individuate morphemes in their lexicon. What matters is that the learners determine that a component of the surface representation of one word expresses the same morpheme as a (possibly non-identical) component of the surface representation of another word. The learner knows, and can represent, when it is seeing two different surface realizations of the same underlying morpheme.

5.9 The map

In Optimality Theory, ranking information determining the grammar comes from winner–loser candidate comparisons. Recursive Constraint Demotion and its variants are capable of finding a constraint hierarchy consistent with a representative set of winner–loser pairs for a language. The limitations of phonotactic learning are not the result of any deficiency in RCD. The limitations of phonotactic learning are the result of limitations on the winner–loser pairs that can be constructed on the basis of purely phonotactic information. RCD cannot make use of winner–loser pairs that it cannot see.

The limitations of purely phonotactic information indicate the need for additional information: paradigmatic information. Paradigmatic information stems from morpheme identity, indicating the surface realization of the same
5.10 Appendix: the Stress/Length typology

There are twenty-four languages in the typology of the Stress/Length system defined in Section 5.2, assuming monosyllabic roots and suffixes. They are labeled L1 through L24. In this appendix, the full map for each language is given, along with the BCD-preferred constraint hierarchy generating the language. Also given is a lexicon of underlying forms, with each underlying form given as a pair of vowel feature values /stress,length/. Morphemes which never contrast (surface identically in all contexts) are grouped together within morphological type (roots are always listed separately from suffixes), and for such groups non-contrastive features (changing the value of the feature never changes the output) have the symbol “?” in place of a feature value.

L1

<table>
<thead>
<tr>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>paká</td>
<td>paká</td>
<td>paká</td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>paká</td>
<td>paká</td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>paká</td>
<td>paká</td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>paká</td>
<td>paká</td>
</tr>
</tbody>
</table>

r1,r2,r3,r4  /?,?/

s1,s2,s3,s4  /?,?/

\{NoLong, WSP, MainRight\} \gg \{MainLeft\} \gg \{Ident[stress], Ident[length]\}
Learning phonotactics

L2

<table>
<thead>
<tr>
<th></th>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>paká</td>
<td>paká</td>
<td>páka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>s2</td>
<td>paká</td>
<td>paká</td>
<td>páka</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>s3</td>
<td>paká</td>
<td>paká</td>
<td>paká</td>
<td>s3 = /-ká/</td>
</tr>
<tr>
<td>s4</td>
<td>paká</td>
<td>paká</td>
<td>paká</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>

r1,r2 /−,?/  r3,r4 /+,?/  
s1,s2 /−,?/  s3,s4 /+,?/  

{NoLong, WSP} ⇒ {Ident[stress]} ⇒ {MainRight} ⇒ {MainLeft} ⇒ {Ident[length]}

L3

<table>
<thead>
<tr>
<th></th>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>paká</td>
<td>paká</td>
<td>paká</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>s2</td>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>s3</td>
<td>paká</td>
<td>paká</td>
<td>paká</td>
<td>s3 = /-ká/</td>
</tr>
<tr>
<td>s4</td>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>

r1,r2,r3,r4 /?/?  
s1,s3 /?,-/  s2,s4 /?,+/  

{WSP, MainRight} ⇒ {MainLeft} ⇒ {Ident[length]} ⇒ {NoLong} ⇒ {Ident[stress]}

L4

<table>
<thead>
<tr>
<th></th>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>paká</td>
<td>paká</td>
<td>páka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>s2</td>
<td>paká:</td>
<td>paká:</td>
<td>páka:</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>s3</td>
<td>paká</td>
<td>paká</td>
<td>paká</td>
<td>s3 = /-ká/</td>
</tr>
<tr>
<td>s4</td>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>

r1,r2 /−,?/  r3 /+,−/  r4 /+,+/  
s1 /−,−/  s2 /−,+/  s3 /+,−/  s4 /+,+/  

{WSP} ⇒ {Ident[stress]} ⇒ {MainRight} ⇒ {MainLeft} ⇒ {Ident[length]} ⇒ {NoLong}
### 5.10 Appendix: the Stress/Length typology

<table>
<thead>
<tr>
<th></th>
<th>L5</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>/pa/</td>
<td>r2</td>
<td>/pa:/</td>
<td>r3</td>
</tr>
<tr>
<td>paká</td>
<td>pa:ká</td>
<td>paká</td>
<td>pa:ká</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>paká:</td>
<td>pa:ká:</td>
<td>paká:</td>
<td>pa:ká:</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>paká</td>
<td>pa:ká</td>
<td>paká</td>
<td>pa:ká</td>
<td>s3 = /-ká/</td>
</tr>
</tbody>
</table>

\[ r1,r3 \ /?,−/ \ r2,r4 \ /?,+/ \]
\[ s1,s3 \ /?,−/ \ s2,s4 \ /?,+/ \]

\{MainRight\} \Rightarrow \{MainLeft\} \Rightarrow \{Ident[length]\} \Rightarrow \{NoLong,WSP\} \Rightarrow \{Ident[stress]\}

### L6

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>/pa/</td>
<td>r2</td>
<td>/pa:/</td>
<td>r3</td>
</tr>
<tr>
<td>paká</td>
<td>pa:ká</td>
<td>páka</td>
<td>pá:ka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>paká:</td>
<td>pa:ká:</td>
<td>páka:</td>
<td>pá:ka:</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>paká</td>
<td>pa:ká</td>
<td>paká</td>
<td>pa:ká</td>
<td>s3 = /-ká/</td>
</tr>
</tbody>
</table>

\[ r1 \ /−,−/ \ r2 \ /−,+/ \ r3 \ /+,−/ \ r4 \ /+,+/ \]
\[ s1 \ /−,−/ \ s2 \ /−,+/ \ s3 \ /+,−/ \ s4 \ /+,+/ \]

\{Ident[stress]\} \Rightarrow \{MainRight\} \Rightarrow \{MainLeft\} \Rightarrow \{Ident[length]\} \Rightarrow \{NoLong,WSP\}

### L7

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>/pa/</td>
<td>r2</td>
<td>/pa:/</td>
<td>r3</td>
</tr>
<tr>
<td>paká</td>
<td>pá:ka</td>
<td>páka</td>
<td>pá:ka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>páka</td>
<td>pá:ka</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>paká</td>
<td>pá:ka</td>
<td>s3 = /-ká/</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>

\[ r1 \ /−,−/ \ r2 \ /−,+/ \ r3 \ /+,−/ \ r4 \ /+,+/ \]
\[ s1 \ /−,−/ \ s2 \ /−,+/ \ s3 \ /+,−/ \ s4 \ /+,+/ \]

\{WSP\} \Rightarrow \{Ident[stress]\} \Rightarrow \{Ident[length]\} \Rightarrow \{NoLong,MainRight\} \Rightarrow \{MainLeft\}
Learning phonotactics

L8

<table>
<thead>
<tr>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>pák:a</td>
<td>paká</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>paká:</td>
<td>pák:a</td>
<td>paká:</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>paká</td>
<td>pák:a</td>
<td>paká</td>
<td>s3 = /-ká/</td>
</tr>
<tr>
<td>paká:</td>
<td>pák:a</td>
<td>paká:</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>

r1,r3 /?,−/ r2,r4 /?,+/ s1,s3 /?,−/ s2,s4 /?,+/

{WSP} ⇒ {Ident[length]} ⇒ {NoLong,MainRight} ⇒ {MainLeft} ⇒ {Ident[stress]}

L9

<table>
<thead>
<tr>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>pák:a</td>
<td>pák:ka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>paká:</td>
<td>pák:a</td>
<td>pák:ka</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>paká</td>
<td>pák:a</td>
<td>pák:ka</td>
<td>s3 = /-ká/</td>
</tr>
<tr>
<td>paká:</td>
<td>pák:a</td>
<td>pák:ka</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>

r1 /−,−/ r2 /−,+/ r3 /+,−/ r4 /+,+/
s1 /−,−/ s2 /−,+/ s3 /+,−/ s4 /+,+/

{WSP} ⇒ {Ident[length]} ⇒ {NoLong} ⇒ {Ident[stress]} ⇒ {MainRight} ⇒ {MainLeft}

L10

<table>
<thead>
<tr>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>pák:a</td>
<td>pák:ka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>paká:</td>
<td>pák:a</td>
<td>pák:ka</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>paká</td>
<td>pák:a</td>
<td>pák:ka</td>
<td>s3 = /-ká/</td>
</tr>
<tr>
<td>paká:</td>
<td>pák:a</td>
<td>pák:ka</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>

r1 /−,−/ r2 /−,+/ r3 /+,−/ r4 /+,+/
s1 /−,−/ s2 /−,+/ s3 /+,−/ s4 /+,+/

{Ident[length]} ⇒ {NoLong} ⇒ {Ident[stress]} ⇒ {WSP} ⇒ {MainRight} ⇒ {MainLeft}
5.10 Appendix: the Stress/Length typology

<table>
<thead>
<tr>
<th>L11</th>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>pá:ka</td>
<td>paká</td>
<td>pá:ka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>paká:</td>
<td>pa:ká:</td>
<td>paká:</td>
<td>pa:ká:</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>paká</td>
<td>pá:ka</td>
<td>paká</td>
<td>pá:ka</td>
<td>s3 = /-ká:/</td>
</tr>
</tbody>
</table>

r1,r3 /?,–/ r2,r4 /?,+/
s1,s3 /?,–/ s2,s4 /?,+/

\{Ident[length]\} \Rightarrow \{NoLong,WSP\} \Rightarrow \{MainRight\} \Rightarrow 
\{MainLeft\} \Rightarrow \{Ident[stress]\}

<table>
<thead>
<tr>
<th>L12</th>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>pá:ka</td>
<td>páka</td>
<td>pá:ka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>paká:</td>
<td>pa:ká:</td>
<td>paká:</td>
<td>pa:ká:</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>paká</td>
<td>pá:ka</td>
<td>paká</td>
<td>pá:ka</td>
<td>s3 = /-ká:/</td>
</tr>
</tbody>
</table>

r1 /−,−/ r2 /−,+/ r3 /+,−/ r4 /+,+/
s1 /−,−/ s2 /−,+/ s3 /+,−/ s4 /+,+/

\{Ident[length]\} \Rightarrow \{NoLong,WSP\} \Rightarrow \{Ident[stress]\} \Rightarrow 
\{MainRight\} \Rightarrow \{MainLeft\}

<table>
<thead>
<tr>
<th>L13</th>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>pá:ka</td>
<td>páka</td>
<td>pá:ka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>pá:ka</td>
<td>pa:ka</td>
<td>páka</td>
<td>pa:ka</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>paká</td>
<td>páka</td>
<td>páka</td>
<td>pá:ka</td>
<td>s3 = /-ká:/</td>
</tr>
<tr>
<td>paká:</td>
<td>pa:ka</td>
<td>paká:</td>
<td>pa:ka</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>

r1 /−,−/ r2 /−,+/ r3 /+,−/ r4 /+,+/
s1 /−,−/ s2 /−,+/ s3 /+,−/ s4 /+,+/

\{WSP\} \Rightarrow \{Ident[stress]\} \Rightarrow \{Ident[length]\} \Rightarrow \{NoLong, 
MainLeft\} \Rightarrow \{MainRight\}
### L14

<table>
<thead>
<tr>
<th></th>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>pák a</td>
<td>pá:ka</td>
<td>pák a</td>
<td>pá:ka</td>
<td>s1 = -ka/</td>
</tr>
<tr>
<td>páká:</td>
<td>pák a</td>
<td>pá:ka</td>
<td>pá:ka</td>
<td>s2 = -ka:/</td>
</tr>
<tr>
<td>páká:</td>
<td>pák a</td>
<td>pák a</td>
<td>pá:ka</td>
<td>s3 = -ká/</td>
</tr>
<tr>
<td>páká:</td>
<td>pák a</td>
<td>pák a</td>
<td>pák a</td>
<td>s4 = -ká:/</td>
</tr>
</tbody>
</table>

r1 /-,/- r2 /-,+/ r3 /+,-/ r4 /+,+/
s1 /-,/- s2 /-,+/ s3 /+,-/ s4 /+,+/

{WSP} ⇒ {Ident[length]} ⇒ {NoLong} ⇒ {Ident[stress]} ⇒ {MainLeft} ⇒ {MainRight}

### L15

<table>
<thead>
<tr>
<th></th>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>pák a</td>
<td>pá:ka</td>
<td>pák a</td>
<td>pá:ka</td>
<td>s1 = -ka/</td>
</tr>
<tr>
<td>páká:</td>
<td>pák a</td>
<td>pák a</td>
<td>pá:ka</td>
<td>s2 = -ka:/</td>
</tr>
<tr>
<td>páká:</td>
<td>pák a</td>
<td>pák a</td>
<td>pá:ka</td>
<td>s3 = -ká/</td>
</tr>
<tr>
<td>páká:</td>
<td>pák a</td>
<td>pák a</td>
<td>pák a</td>
<td>s4 = -ká:/</td>
</tr>
</tbody>
</table>

r1 /-,/- r2 /-,+/ r3 /+,-/ r4 /+,+/
s1 /-,/- s2 /-,+/ s3 /+,-/ s4 /+,+/

{Ident[length]} ⇒ {NoLong} ⇒ {Ident[stress]} ⇒ {WSP} ⇒ {MainLeft} ⇒ {MainRight}

### L16

<table>
<thead>
<tr>
<th></th>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>pák a</td>
<td>pá:ka</td>
<td>pák a</td>
<td>pá:ka</td>
<td>s1 = -ka/</td>
</tr>
<tr>
<td>páká:</td>
<td>pák a</td>
<td>pák a</td>
<td>pá:ka</td>
<td>s2 = -ka:/</td>
</tr>
<tr>
<td>páká:</td>
<td>pák a</td>
<td>pák a</td>
<td>pá:ka</td>
<td>s3 = -ká/</td>
</tr>
<tr>
<td>páká:</td>
<td>pák a</td>
<td>pák a</td>
<td>pák a</td>
<td>s4 = -ká:/</td>
</tr>
</tbody>
</table>

r1 /-,/- r2 /-,+/ r3 /+,-/ r4 /+,+/
s1 /-,/- s2 /-,+/ s3 /+,-/ s4 /+,+/

{Ident[length]} ⇒ {NoLong,WSP} ⇒ {Ident[stress]} ⇒ {MainLeft} ⇒ {MainRight}
### 5.10 Appendix: the Stress/Length typology

<table>
<thead>
<tr>
<th></th>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>pák:ka</td>
<td>páka</td>
<td>pák:ka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>paká:</td>
<td>pák:ka</td>
<td>paká:</td>
<td>pák:ka</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>páka</td>
<td>pák:ka</td>
<td>pák:ka</td>
<td>pák:ka</td>
<td>s3 = /-ká/</td>
</tr>
<tr>
<td>paká:</td>
<td>pák:ka</td>
<td>pák:ka</td>
<td>pák:ka</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>

\[ r1, r3 /?,-/ \quad r2, r4 /?,+/
\]
\[ s1, s3 /?,-/ \quad s2, s4 /?,+/
\]

\[ \{\text{WSP}\} \gg \{\text{Ident[length]}\} \gg \{\text{NoLong, MainLeft}\} \gg \{\text{MainRight}\} \gg \{\text{Ident[stress]}\} \]

### L18

<table>
<thead>
<tr>
<th></th>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>pák:ka</td>
<td>páka</td>
<td>pák:ka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>paká:</td>
<td>pák:ka:</td>
<td>paká:</td>
<td>pák:ka:</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>páka</td>
<td>pák:ka</td>
<td>pák:ka</td>
<td>pák:ka</td>
<td>s3 = /-ká/</td>
</tr>
</tbody>
</table>

\[ r1, r3 /?,-/ \quad r2, r4 /?,+/
\]
\[ s1, s3 /?,-/ \quad s2, s4 /?,+/
\]

\[ \{\text{Ident[length]}\} \gg \{\text{NoLong, WSP}\} \gg \{\text{MainLeft}\} \gg \{\text{MainRight}\} \gg \{\text{Ident[stress]}\} \]

### L19

<table>
<thead>
<tr>
<th></th>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>pák:ka</td>
<td>pák:ka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>pák:</td>
<td>pák:</td>
<td>pák:</td>
<td>pák:</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>paká</td>
<td>pák</td>
<td>pák</td>
<td>pák</td>
<td>s3 = /-ká/</td>
</tr>
<tr>
<td>pkák</td>
<td>pkák</td>
<td>pkák</td>
<td>pkák</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>

\[ r1, r2 /?,-/ \quad r3, r4 /?,+/
\]
\[ s1, s2 /?,-/ \quad s3, s4 /?,+/
\]

\[ \{\text{NoLong, WSP}\} \gg \{\text{Ident[stress]}\} \gg \{\text{MainLeft}\} \gg \{\text{MainRight}\} \gg \{\text{Ident[length]}\} \]
Learning phonotactics

L20

<table>
<thead>
<tr>
<th></th>
<th>r1 = /pa/</th>
<th>r2 = /paː/</th>
<th>r3 = /pá/</th>
<th>r4 = /páː/</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>páka</td>
<td>s1 = /-ka/</td>
<td></td>
</tr>
<tr>
<td>páka</td>
<td>páka</td>
<td>páka</td>
<td>s2 = /-kaː/</td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>páka</td>
<td>s3 = /-ká/</td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>pása</td>
<td>pása</td>
<td>s4 = /-káː/</td>
<td></td>
</tr>
</tbody>
</table>

r1, r2, r3, r4 /?, ?, ?/  
s1, s2, s3, s4 /?, ?, ?, ?/

{WSP} ⇒ {IDENT[stress]} ⇒ {MAINLEFT} ⇒ {MAINRIGHT} ⇒ {IDENT[length]} ⇒ {NOLONG}

L21

<table>
<thead>
<tr>
<th></th>
<th>r1 = /pa/</th>
<th>r2 = /paː/</th>
<th>r3 = /pá/</th>
<th>r4 = /páː/</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>pása</td>
<td>pása</td>
<td>s1 = /-ka/</td>
<td></td>
</tr>
<tr>
<td>páka</td>
<td>pása</td>
<td>pása</td>
<td>s2 = /-kaː/</td>
<td></td>
</tr>
<tr>
<td>pása</td>
<td>pása</td>
<td>pása</td>
<td>s3 = /-ká/</td>
<td></td>
</tr>
<tr>
<td>pása</td>
<td>pása</td>
<td>pása</td>
<td>s4 = /-káː/</td>
<td></td>
</tr>
</tbody>
</table>

r1, r2, r3, r4 /?, ?, ?, ?/  
s1, s2, s3, s4 /?, ?, ?, ?/

{NOLONG, WSP, MAINLEFT} ⇒ {MAINRIGHT} ⇒ {IDENT[stress], IDENT[length]}

L22

<table>
<thead>
<tr>
<th></th>
<th>r1 = /pa/</th>
<th>r2 = /paː/</th>
<th>r3 = /pá/</th>
<th>r4 = /páː/</th>
</tr>
</thead>
<tbody>
<tr>
<td>pása</td>
<td>pása</td>
<td>pása</td>
<td>s1 = /-ka/</td>
<td></td>
</tr>
<tr>
<td>pása</td>
<td>pása</td>
<td>pása</td>
<td>s2 = /-kaː/</td>
<td></td>
</tr>
<tr>
<td>pása</td>
<td>pása</td>
<td>pása</td>
<td>s3 = /-ká/</td>
<td></td>
</tr>
<tr>
<td>pása</td>
<td>pása</td>
<td>pása</td>
<td>s4 = /-káː/</td>
<td></td>
</tr>
</tbody>
</table>

r1, r3 /?, ?, ?/  
s1, s2, s3, s4 /?, ?, ?, ?/

{WSP, MAINLEFT} ⇒ {MAINRIGHT} ⇒ {IDENT[length]} ⇒ {NOLONG} ⇒ {IDENT[stress]}
### 5.10 Appendix: the Stress/Length typology

#### L23

<table>
<thead>
<tr>
<th></th>
<th>(r_1)</th>
<th>(r_2)</th>
<th>(r_3)</th>
<th>(r_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>(\text{pá:}ka)</td>
<td>(\text{pá:}ka)</td>
<td>(\text{pá:}ka)</td>
<td>(s_1 = -ka/)</td>
</tr>
<tr>
<td>páka:</td>
<td>(\text{pá:}ka:)</td>
<td>(\text{pá:}ka:)</td>
<td>(\text{pá:}ka:)</td>
<td>(s_2 = -ka:/)</td>
</tr>
<tr>
<td>paká</td>
<td>(\text{pa:ká})</td>
<td>(\text{pá:}ka)</td>
<td>(\text{pá:}ka)</td>
<td>(s_3 = -ká/)</td>
</tr>
<tr>
<td>paká:</td>
<td>(\text{pa:ká}:)</td>
<td>(\text{pá:}ka:)</td>
<td>(\text{pá:}ka:)</td>
<td>(s_4 = -ká:/)</td>
</tr>
</tbody>
</table>

\(r_1\) \(-,−/r_2\) \(-,+/r_3\) \(+,−/r_4\) \(+,+/r\)

\(s_1\) \(-,−/s_2\) \(-,+/s_3\) \(+,−/s_4\) \(+,+/s\)

\{\textbf{Ident[stress]}\} \gg \{\textbf{MainLeft}\} \gg \{\textbf{MainRight}\} \gg \{\textbf{Ident[length]}\} \gg \{\textbf{NoLong, WSP}\}

#### L24

<table>
<thead>
<tr>
<th></th>
<th>(r_1)</th>
<th>(r_2)</th>
<th>(r_3)</th>
<th>(r_4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>(\text{pá:}ka)</td>
<td>(\text{pá:}ka)</td>
<td>(\text{pá:}ka)</td>
<td>(s_1 = -ka/)</td>
</tr>
<tr>
<td>páka:</td>
<td>(\text{pá:}ka:)</td>
<td>(\text{pá:}ka:)</td>
<td>(\text{pá:}ka:)</td>
<td>(s_2 = -ka:/)</td>
</tr>
<tr>
<td>paká</td>
<td>(\text{pa:ká})</td>
<td>(\text{pá:}ka)</td>
<td>(\text{pá:}ka)</td>
<td>(s_3 = -ká/)</td>
</tr>
<tr>
<td>paká:</td>
<td>(\text{pa:ká}:)</td>
<td>(\text{pá:}ka:)</td>
<td>(\text{pá:}ka:)</td>
<td>(s_4 = -ká:/)</td>
</tr>
</tbody>
</table>

\(r_1,r_3\) \(?,-/r_2,r_4\) \(?,+/

\(s_1,s_3\) \(?,-/s_2,s_4\) \(?,+/

\{\textbf{MainLeft}\} \gg \{\textbf{MainRight}\} \gg \{\textbf{Ident[length]}\} \gg \{\textbf{NoLong, WSP}\} \gg \{\textbf{Ident[stress]}\}
This chapter provides an overview of some of the issues involved with learning non-phonotactic aspects of the grammar, specifically phonological underlying forms and non-phonotactic ranking information. Such learning is based on the use of paradigmatic information, the subject of Section 6.1. Section 6.2 lays out the main computational challenge posed by the learning of non-phonotactic aspects of a grammar and includes a very brief review of some selected prior work on the learning of underlying forms and non-phonotactic ranking information. Section 6.3 discusses some issues relating the present work to the overall study of language learning and acquisition.

The benefits of output-driven maps for learning depend on the use of inconsistency detection in learning. A learner can conclude that an underlying feature for a morpheme has a certain value if it can determine that any other possible value for the feature is inconsistent with what the learner already knows (a kind of process of elimination). Section 6.4 examines inconsistency detection in detail: what it means for information to be mutually inconsistent in Optimality Theory, and how the computation of inconsistency emerges as a natural consequence of Recursive Constraint Demotion. Section 6.5 demonstrates in detail the use of inconsistency detection in the learning of underlying feature values.

Non-phonotactic ranking information is the subject of Section 6.6. An algorithm for obtaining such information, Merchant’s join operation, is given. The two kinds of non-phonotactic information, underlying forms and non-phonotactic ranking information, are ultimately interdependent, and Section 6.7 presents an overview of a proposal that combines the learning of both into a single algorithm, the Contrast Pairs and Ranking (CPR) algorithm.

CPR is a significant advance in the learning of non-phonotactic information. It can be computationally expensive, however, as is discussed in Section 6.8. That sets the stage for Chapter 7, which will examine the contributions that
output-drivenness can make to learning, and describe a learner that combines the use of output-drivenness with elements of CPR.

6.1 Paradigmatic information

Phonotactic ranking information is obtained by temporarily adopting, for each word, an input that is segmentally identical to the output form. Thus, phonotactic ranking information consists of ranking information obtainable from fully faithful mappings. As shown in Section 5.8, obtaining non-phonotactic ranking information requires the learner to posit unfaithful mappings. That, in turn, requires paradigmatic information: information about the appearance of different morphemes in the same context, and about the appearance of the same morpheme in different contexts. The setting of an underlying feature for a morpheme must be motivated by how the morpheme surfaces in at least one context, and that set feature value can then be the basis for an unfaithful mapping if the morpheme, in a different context, appears with a different value for the feature on the surface.

I will distinguish two kinds of paradigmatic information. Morphemic contrast refers to the surface realizations of different morphemes in the same morphological context. Morphemic alternation refers to the surface realizations of the same morpheme in different morphological contexts. The two kinds of paradigmatic information play different roles in learning, but both are essential.

For language L20, consider the words r2s2, which surfaces as pā:ka, and r2s3, which surfaces as paká. This pair provides an example of morphemic contrast between the suffixes s2 and s3, in the morphological environment of root r2. Suffix s2 surfaces unstressed in this environment, while s3 surfaces as stressed. Because the morphological environment, r1, is the same morpheme in both words, the underlying form for the environment is the same in both words. Therefore, the only differences between the inputs can be differences between the underlying forms for s2 and s3, and given that the words surface non-identically, there must be at least one difference between the underlying forms of s2 and s3, because the difference(s) are responsible for the different surface realizations of the two words.

The words r2s2 and r2s3 also provide an example of morphemic alternation for root r2, in the two environments provided by s2 and s3. Root r2 surfaces as stressed and long in the environment of s2, but surfaces as unstressed and short in the environment of s3. Since r2 can have only one underlying form, at least one of the two words involves an unfaithful mapping for r2.
6.2 The explosive growth of lexical hypothesis spaces

6.2.1 Now that’s big
A phonological learner must simultaneously learn the ranking and lexicon (Hale and Reiss 1997, Tesar and Smolensky 2000). This poses a far greater computational challenge than learning the constraint ranking alone, in part because of the explosive combinatorial growth in the number of possible lexica. Exhaustively evaluating all possible lexicon-ranking combinations (Hale and Reiss 1997) is hopelessly intractable. Even modest assumptions lead to large numbers. In a system where all segments possess five binary features and each morpheme has an underlying form with three segments, a morpheme has $2^{15} \approx 32,000$ possible underlying forms. For a lexicon of only fifty morphemes, that yields $(2^{15})^{50} \approx 10^{200}$ possible lexica alone.

To put this space of $10^{200}$ possible lexica in perspective, consider that the universe is estimated to contain a total of about $10^{80}$ atoms. Very little is known about how the human brain conducts computations, but it is not hard to justify the claim that exhaustive search of the space of possible lexica is not computationally plausible for language learners. If one could command the entire universe as one giant computer, and each atom in the universe could on its own evaluate a trillion ($10^{12}$) lexica per second, it would still take $10^{108}$ seconds, or $10^{100}$ years, to evaluate all of the lexica.1 This is what “hopelessly intractable” means in the previous paragraph. The combinatorics of linguistics are such that even arguments as crude as this carry real force.

6.2.2 The basic alternant constraint
One alternative view about the nature of phonological underlying forms is that the underlying form for a morpheme should be identical to one of its surface allomorphs. There have been a variety of proposals for restricting underlying forms in this way. Kenstowicz and Kisseberth (1979: 196–204) review several such proposals and group them under the label “the basic alternant.” While some proposals impose further restrictions on the basic alternant, the weaker and more general basic alternant constraint is stated by Kenstowicz and Kisseberth as follows: “All of the segments appearing in the UR [underlying representation – BBT] must occur together in at least one phonetic alternant – the basic alternant” (Kenstowicz and Kisseberth 1979: 202). They cite McCawley (1967) as an example of work adopting this view; a more recent example is work by Albright (2002). From the point of view of learning, the intuitive appeal of this idea is

---

1 The number $10^{100}$ is commonly known as a googol.
6.2 The explosive growth of lexical hypothesis spaces

that it might greatly restrict the number of candidate underlying forms for a morpheme: instead of a combinatorially large space of possibilities, the learner is limited to only those variations that have actually been observed.

The main problem with the basic alternant view is theoretical: it is inconsistent with numerous straightforward analyses of phonological phenomena. One well-known example is vowel reduction in Palauan (Flora 1974, Schane 1974). Another example can be found in the analysis of Pāli by de Lacy (2002, chap. 8), and several other examples are discussed by Kenstowicz and Kisseberth (1979: 196–204). The theme of these examples is that different parts of an underlying form are each well motivated by how the form surfaces in different environments, but no one environment simultaneously exhibits all of the parts on the surface at once.

The Palauan example involves vowel reduction in unstressed position. Unstressed vowels reduce to schwa. Stress appears on the penultimate syllable, except when a suffix is present, in which case stress appears on the final syllable. Thus, stress can shift with respect to a root depending on what morphemic environment the root is in, so a root vowel may reduce in some environments and not in others. This is illustrated in (6.1) with the verb root /daŋob/ ‘cover opening’; the data are from Flora 1974.

(6.1) Verb root in three different morphemic environments

<table>
<thead>
<tr>
<th></th>
<th>present middle</th>
<th>future participle (conservative)</th>
<th>future participle (innovative)</th>
</tr>
</thead>
<tbody>
<tr>
<td>mə-dāŋob</td>
<td>present middle</td>
<td>future participle (conservative)</td>
<td>future participle (innovative)</td>
</tr>
<tr>
<td>daŋob-ll</td>
<td></td>
<td>future participle (conservative)</td>
<td>future participle (innovative)</td>
</tr>
<tr>
<td>daŋob-áll</td>
<td></td>
<td>future participle (innovative)</td>
<td></td>
</tr>
</tbody>
</table>

In the present middle form, there is no suffix, and stress appears on the first vowel of the root. In the future participle (conservative) form, there is a suffix, but the suffix has no vowel, so the vowel of the final syllable is the second vowel of the root. In the future participle (innovative) form, there is a suffix with a vowel, so the vowel of the final syllable is the suffix vowel. In these three forms, the verb root is stressed on either the first vowel, the second vowel, or neither vowel. Palauan prosody prevents the root from ever surfacing with both vowels simultaneously stressed. However, full vowel quality is not predictable, and must be specified underlyingly: the first vowel of the root in (6.2) is /a/, while the second vowel of the root is /o/. To account for all of the surface forms, the underlying form for the root should have both full vowels. Somewhat perversely for the basic alternant condition, that is the only combination that does not appear as a single surface alternant; the only combination of full and
reduced vowels eliminated from consideration by the basic alternant condition
is the correct one.

Section 5.5.2 argued that, for learning rankings, the structure of the space of
grammars is more important than the size, and that learnability considerations
actually favor the use of the space of possible grammars (constraint hierarchies)
over the space of distinct languages (one grammar in the space for each distinct
generated language), even though the former is typically much larger. A similar
case can be made for the learning of underlying forms: the basic alternant
condition, in addition to being theoretically problematic, is undesirable from
the point of view of learnability. For a learner to employ the basic alternant
constraint, the learner may need to wait until it is convinced it has seen all
surface forms of a morpheme before drawing conclusions about its underlying
form (or any other conclusions about the grammar on the basis of that
morpheme’s underlying form). If the word containing the basic alternant of a
morpheme is particularly infrequent, but other words containing the morpheme
are much more frequent, this could be a significant problem. Even if the more
frequent words, taken together, unambiguously indicated the correct under-
lying form for the morpheme, the learner would prevent itself from making
use of that information until it observed the word in which the basic alternant
surfaced.

The MRCD approach to learning constraint rankings treats (the ranking part
of) a grammar as a set of pieces of information, the winner–loser pairs, rather
than as an atomic, indivisible hypothesis. It can reason directly from data to the
grammatical principles most relevant to that data. Different winner–loser pairs
can come from different words, and the same ranking information can typically
be represented in a winner–loser pair from any of a variety of words. The
approach to learning underlying forms taken by the CPR algorithm has the same
spirit: an underlying form is seen as a structured collection of feature instances,
rather than as an atomic, indivisible form. The values for different feature
instances can come from different words in which the morpheme appears, and
the value for a feature instance can possibly be determined by any of several
words. This approach will be explained throughout the rest of this chapter.

6.2.3 Selected prior work
The computational challenges posed by the combinatorics of the lexicon are
addressed in some recent work on phonological learnability. While some
approaches manage to avoid exhaustive search of all possible lexica, all of
the approaches face the prospect of enumerating most or all of the possible
underlying forms for individual morphemes, along with at least some
combinatorial interaction between the possible underlying forms of different morphemes.

Jarosz (2006) has investigated an approach based on likelihood maximization, called Maximum Likelihood Learning of Lexicons and Grammars, or MLG. MLG simultaneously learns about both constraint rankings and underlying forms, can display a bias towards more restrictive grammars, and can display robustness and sensitivity to frequency. However, to accomplish this it must define probability distributions over the space of possible rankings and, separately, the space of possible underlying forms for each morpheme. The work is intended to demonstrate the ability of likelihood maximization to contend with several learning issues, not to account for the computational limitations of human learners, so the simulations described by Jarosz use exhaustive evaluation of all possible rankings and all possible lexica (repeatedly). For this kind of approach to be ultimately successful, some way will have to be found to optimize likelihoods and estimate the hidden variables in candidates (key computational components of MLG) that completely escapes the massive growth rates for the sizes of these spaces.

Apoussidou (2007) has investigated an approach based on the Gradual Learning Algorithm and lexical constraints against possible underlying forms (Boersma 2001). The Gradual Learning Algorithm, or GLA, is error driven, and uses production-directed parsing to construct winner–loser pairs, which are then used to adjust the learner’s ranking information. Lexical entries are represented by constraints penalizing the use of the different possible underlying forms for a morpheme. The choice of underlying form for a given morpheme is then largely determined by the relative ranking of all of the lexical constraints for that morpheme.

Representing lexical entries in this way allows the GLA ranking algorithm to also be used in the learning of the underlying forms. However, in the process it greatly transforms the computational nature of parsing and the learning of rankings. In this approach, the set of constraints contains not only the markedness and faithfulness constraints, but also a separate constraint for every possible underlying form, and a separate set of such constraints for every morpheme. This will explode the number of constraints for systems with even relatively modest assumptions about the number of possible underlying forms. Thus, simply parsing a form requires evaluating candidates with respect to a huge number of constraints. The explosive combinatorics of the lexicon aren’t avoided, they

---

2 The impact is compounded by the fact that each lexical constraint penalizes a possible underlying form, so that the optimal underlying form for a morpheme will commonly be determined by the lowest-ranked lexical constraint for that morpheme.
are shifted into the constraint space: the number of possible rankings is now greater than the number of possible lexica. The GLA ranking algorithm does not do anything like exhaustively evaluate all possible rankings, and therefore will not exhaustively evaluate all possible lexica, but its computational requirements are still a function of the number of constraints. Even if one had a ranking algorithm that was linear in the number of constraints, it would be linear with respect to the hugely expanded number of constraints, a number reflecting the possible underlying forms for morphemes. For this kind of approach to be ultimately successful, some way will have to be found to compute optima and learn rankings from words that manages to avoid explicit evaluation of the vast majority of lexical constraints, even for the morphemes included in the word being processed at any given time.

Merchant (Merchant 2008, Merchant and Tesar 2008) proposes evaluating local lexica for a small morpheme set, in an approach that determines underlying forms by setting one feature at a time (rather than treating underlying forms monolithically). A local lexicon is a possible assignment of feature values to unset underlying features, so the number of possible local lexica goes down as more features are set. This is better than exhaustive search of all possible underlying forms, but the number of local lexica is still exponential in the number of unset features. For this approach to be ultimately successful, some way will have to be found to perform feature setting for a morpheme set without having to exhaustively generate and evaluate all local lexica.

Each of these recent lines of research has advanced the field in various ways, but computationally the techniques are still implausibly slow. Processing all underlying forms for even a modest number of morphemes gets expensive very quickly. The claim motivating the present work is that faster (and more cognitively plausible) learning will require additional posited structure in the space of possible grammars, structure that can be exploited by a learner to effectively search the space without exhaustively evaluating all (or even most) of the possibilities in the space. The concept of output-driven maps is here proposed as that additional structure.

### 6.2.4 Combinatorics of the Stress/Length linguistic system

Recall the Stress/Length linguistic system. For each input with \( v \) vowels, there are \( v^2 \) possible candidates,\(^3\) one for each output form. For each output with \( v \)

\( ^3 \) Exactly one output syllable has stress, with \( v \) to choose from, combined with all possible assignments of length to all of the vowels, with \( 2^v \) possibilities. Combining the two choices yields \( v2^v \) possible outputs.
6.2 The explosive growth of lexical hypothesis spaces

vowels, there are $4^v$ possible candidates,\footnote{A completely independent choice of values for stress and length are possible for each vowel, with two values for each feature, means a total of $(2*2)^v = 4^v$ inputs.} one for each input form. Thus, the number of possible inputs for a word (and, correspondingly, the number of possible underlying forms for a morpheme) grows exponentially in the number of vowels. Increasing the inventory of possible vowels to greater than the four considered here, by adding more features to the vowels, will further expand the space of possibilities, in general exponentially with respect to the number of features.

For present purposes, the illustrations are restricted to two-syllable words (two morphemes, one syllable each), meaning each word has two vowels. Each morpheme consists of a single syllable, and there are four possible syllables, so there are four possible underlying forms for each morpheme. For each type of morpheme (root, suffix), there are four potentially phonologically distinguishable morphemes, one for each possible syllable, giving a total of eight morphemes, four roots and four suffixes. Thus, there are $4^8 = 65,536$ possible lexica. The six constraints admit a total of $6! = 720$ possible total rankings. The total number of possible grammatical hypotheses is thus 47,185,920.

Again, the alert reader may wonder about the discrepancy between the number of grammatical hypotheses and the number of distinct languages in the typology, twenty-four. This was already discussed with respect to possible rankings in Section 5.5.2. Once we include the lexicon in grammars, the divergence in size between the grammar space and the language space becomes much greater. In addition to having multiple total rankings defining the same map, there is the possibility (depending on the map) of multiple underlying forms neutralizing in all environments; grammars differing only in which of the neutralizing underlying forms they assign to a morpheme will generate the same language.

The number of languages in the typology, as presented in Section 5.10, is very misleading with respect to the learning of the lexicon. The languages listed in the typology simply enumerate the possible underlying forms and their surface behaviors. No morpheme identity holds across different grammars, because that is irrelevant to the purely phonological behavior of the system. But the learner is attempting to learn not only what the possible morpheme behaviors are, but which behaviors go with which morphemes.

We could model this by providing pre-determined meanings for morphemes, to be used in every language. For example, we could define four root morpheme meanings, “dog,” “cat,” “pig,” and “bat,” and we could define
four suffix morpheme meanings, “nominative,” “accusative,” “genitive,” and “ablative.” This would provide the missing morpheme identity that holds across grammars and would result in a language typology of greatly expanded size. To use language L20 as an example, L20 has four distinct root behaviors and three distinct suffix behaviors. Assuming that all four root behaviors are represented, there are twenty-four ways of assigning four meanings to four root behaviors. Assuming, for the suffixes, that two of the meanings are assigned to the neutralizing suffix behavior, and one meaning is assigned to each of the others, there are twelve ways of assigning meanings to the suffix behaviors. The number of distinct versions of L20, differing in the surface form assigned to at least one word meaning, would then be $24 \times 12 = 288$. Thus, language L20 in the original typology would be replaced with 288 grammars in the expanded typology. A completely neutralizing language like L21, with only a single root behavior and a single suffix behavior, would be replaced by only one language in the expanded typology. The sum of each such appropriate replacement count, one for each of the twenty-four distinct maps defined by the rankings, would constitute the size of the expanded typology and would more accurately reflect the range of empirical possibilities faced by the learner. While far less than 47,185,920, the total would nevertheless be far greater than twenty-four.

The points made about the distinction between grammar space and language space in Section 5.5.2 apply with even greater force once we consider the lexicon along with the ranking. While the language space may be much smaller than the grammar space, it is still non-trivial in size. Furthermore, the description of the grammar space is much smaller than the language space: a list of the constraints to be ranked, along with a list of the feature instances to be set for the underlying forms, is quite compact. It is the structure of the grammar space, not its size, that is most relevant for learning.

6.3 An aside on methodology

Phonotactic learning, as described in Section 5.6.2, takes as data the phonological output forms of individual words. As a model of early language acquisition, that is clearly an idealization, on several levels. Child learners early on need to learn about the language-specific phonetic details of phonological

---

5 Such a language has only one surface form; for L21, that single surface form is [páka]. All word meanings in the language have no choice but to be realized as that same single surface form.
representations. They also need to segment the speech stream into words in order to know what the words are, and it is quite likely that learning phonotactic regularities plays a role in this: the learning of phonotactic regularities and word segmentation will interact and happen in tandem. Child learners are also engaged in learning phonology at levels above the word, including phrasal phonology. The idealization of phonotactic learning has proven to be very useful: a lot has been learned about the role of phonological theory in phonotactic learning, and the ability to efficiently determine phonotactic regularities from phonological output forms is almost unavoidably an important part of early phonological learning.

The work described in this chapter and the next assumes that the learner has access to phonological outputs that are morphologically segmented, providing the learner with both the identity of each morpheme in the word and the morphological affiliation of each part of the output. This information on morpheme identity is crucial, as it informs the learner when two different surface realizations are in fact realizations of the same morpheme, as well as when a single surface realization in different words are in fact separate, possibly neutralized realizations of different morphemes.

Providing the learner with access to morpheme identity information is clearly another idealization. Child learners, in addition to word segmentation, need to perform morpheme segmentation on their own. Further, it is likely that morpheme segmentation and identification interacts in complex ways with both phonological learning and lexical semantic learning. The idealization of providing the learner with morpheme identity information is also quite useful. It avoids (for the time being) the many messy complexities inherent in lexical semantic learning, morphosyntactic learning, and morphological segmentation. This allows us to focus on the role of phonological theory in the learning of non-phonotactic ranking information and phonological underlying forms. A better understanding of these specific issues will provide the basis for progress on the more general issues in learning.

The claim that work on these idealized problems will provide the basis for further progress has precedent behind it, in many areas of scientific inquiry and specifically in past work on phonological learning. When constraint demotion was first proposed (Tesar and Smolensky 1994), the work presumed that the learner was given complete winner–loser pairs. At the time, the claim was made that the result under this idealization was important, and that it would be the basis for work on the problem of actually selecting informative competitors. Subsequent work made good on this claim with the development of Error-Driven Constraint Demotion, which used the intermediate rankings derived
by constraint demotion to direct parsing toward informative competitors (Tesar 1995, 1998b). The idealization had been relaxed, but still assumed that the learner was provided with the phonological inputs and outputs, including hidden structure. Subsequent work addressed the issue of hidden structure in the output by building on constraint demotion and error-driven competitor selection (Tesar 1998a, 2004, Tesar and Smolensky 2000). More recent work has pursued the learning of underlying forms, further relaxing earlier idealizations (Merchant and Tesar 2008, Tesar 2006a, 2006b, Tesar and Prince 2007). This work on the learning of underlying forms is built directly on prior work on Recursive Constraint Demotion, error-driven competitor selection, inconsistency detection, and restrictiveness biases, all of which were successfully developed under stronger idealizations.

The sequence of research just described is but one coherent thread in a larger literature of work on phonological learning. Other work has both influenced and been influenced by this thread, while working with the same or similar idealizations. All told, this provides strong support for the methodological approach taken in this book, which prefers explicit, up-front statements of the idealizations in use.

6.4 Inconsistency detection

A list of winner–loser pairs is inconsistent when there does not exist a total ordering of the constraints that simultaneously satisfies all of the pairs. If one pair requires that ConstraintA dominate ConstraintB, and another pair requires that ConstraintB dominate ConstraintA, the two pairs are logically inconsistent with each other: they cannot simultaneously be true of the same ranking.

One particularly interesting and useful property of Recursive Constraint Demotion and its variants is that, when given a set of winner–loser pairs that is inconsistent, the algorithm rapidly determines this fact. Inconsistency detection happens as an automatic consequence of the normal operation of the algorithm. This can be illustrated with the inconsistent set of winner–loser pairs shown in (6.2). Note that the first winner shortens an underlyingly long vowel in stressed position, while the second winner permits an underlyingly long vowel to surface long in stressed position, and the third winner permits an underlyingly long vowel to surface as long in unstressed position.

6 The issue of hidden structure in the output is beyond the scope of this book, but see Section 9.4.3 for a brief discussion of the issue with respect to learning with output-driven maps.
6.4 Inconsistency detection

(6.2) An inconsistent set of winner–loser pairs

<table>
<thead>
<tr>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>/pá:ka/ páká ~ pá:ka</td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>/pa:ka/ páká ~ paká</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>/paka:/ páká: ~ páká</td>
<td>L</td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
</tr>
</tbody>
</table>

If we apply RCD to this set of winner–loser pairs, it will first rank MainLeft and IDENT[stress] at the top, as they are the only constraints initially preferring no losers. IDENT[stress] does not prefer any winners, but MainLeft prefers the winner in the second pair, so that pair can be removed.

(6.3) \{MainLeft IDENT[stress]\}

(6.4) The remaining winner–loser pairs after the first pass of RCD

<table>
<thead>
<tr>
<th>WSP</th>
<th>IDENT[length]</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>/pá:ka/ páká ~ pá:ka</td>
<td>L</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>/paka:/ páká: ~ páká</td>
<td>L</td>
<td>W</td>
<td>L</td>
</tr>
</tbody>
</table>

On the next pass, the only constraint available to be ranked is MainRight, so it is placed into the second stratum of the hierarchy. However, MainRight does not prefer any winners, so no winner–loser pairs are removed.

(6.5) \{MainLeft IDENT[stress]\} \gg \{MainRight\}

(6.6) The remaining winner–loser pairs after the second pass of RCD

<table>
<thead>
<tr>
<th>WSP</th>
<th>IDENT[length]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>/pá:ka/ páká ~ pá:ka</td>
<td>L</td>
<td>W</td>
</tr>
<tr>
<td>/paka:/ páká: ~ páká</td>
<td>L</td>
<td>W</td>
</tr>
<tr>
<td>Fusion</td>
<td>L</td>
<td>L</td>
</tr>
</tbody>
</table>

Now the learner observes that none of the remaining constraints are available for ranking; each one of them prefers at least one loser. The algorithm cannot continue: there are constraints remaining to be ranked, but none can be ranked. At this point, RCD halts, returning an indication that inconsistency has been detected. Note that the fusion of the winner–loser pairs in (6.6) is a trivially false ERC: all three remaining constraints fuse to L. In fact, the same pairs in (6.4) also fuse to a trivially false ERC, one with three Ls and one e. In general, an inconsistent set of winner–loser pairs will not necessarily fuse to a trivially
false ERC, but RCD will always reduce the list to a subset that does fuse to a trivially false ERC.

The illustration just given used RCD with the “all constraints as high as possible” bias, but choice of bias is irrelevant to inconsistency detection. The bias determines which hierarchy is constructed from among those that are consistent with the winner–loser pairs; if the pairs are inconsistent, then there are no consistent hierarchies to choose from. If BCD, with the “faithfulness as low as possible” bias, were applied to (6.2), it would construct the partial hierarchy in (6.7) before terminating. This hierarchy differs from that in (6.5) only in that the (inactive) faithfulness constraint IDENT [stress] is ranked at the bottom instead of the top. BCD still winds up with the pairs in (6.6) and on that basis detects inconsistency in the winner–loser pairs.

\[(6.7) \quad \{ \text{MainLeft} \} \gg \{ \text{MainRight} \} \gg \{ \text{IDENT[stress]} \} \]


The key benefit of inconsistency detection for learning is the ability to detect when multiple analytic commitments cannot be simultaneously part of the same grammar, even though each commitment is possible on its own. Looking back to the winner–loser pairs in (6.2), the first winner–loser pair requires that underlyingly long vowels surface as short, even in stressed position. The second and third winner–loser pairs require that an underlyingly long vowel surface as long, in stressed and unstressed positions, respectively. Each of these mappings on its own is possible; for each such pair, there are grammars that admit that mapping. However, no grammar admits all of them simultaneously: in this system, there is no way for a single grammar to neutralize vowel length to short in surface-stressed vowels (winner–loser pair 1), but preserve underlyingly length in surface-unstressed vowels (winner–loser pair 3). Inconsistency detection enables the learner to test analytic hypotheses for consistency with other information about distinct but interacting parts of the language.

### 6.5 Setting underlying features via inconsistency detection

Inconsistency detection can be used to evaluate features in the underlying forms for morphemes. At the most general level, a hypothesized set of
underlying forms can be tested for consistency. If no available ranking yields the correct outputs for the inputs formable from the underlying forms, then the hypothesized set is inconsistent; something must be wrong with at least one of the underlying forms of the hypothesis. More narrowly, the logic of determining the value of an underlying feature is to determine that every other possible value for the feature leads to inconsistency. For binary features, if one value of a feature can be demonstrated to be inconsistent with things that are already known, then the feature must have the opposite value.

A learner sets a feature when it permanently marks a value for that feature in its lexicon. It would be more technically correct to refer to “setting the value of an underlying feature instance”: when a learner sets a feature, it is permanently fixing the value of a feature for a particular segment in a particular underlying form. The phrase “setting the value of an underlying feature instance” is, however, too cumbersome to be practical, so I will generally use the shorter “setting a feature” in its place.

6.5.1 Feature setting

The learners contemplated in much of this chapter and the next approach the lexicon by selectively setting features of underlying forms. This means that, when a learner initially constructs an underlying form for a morpheme, the presence of segments in the underlying form is indicated, but each of the features for each of the segments is unset (has not been set). In general the features of a segment will not be set all at once (let alone all of the features for all of the segments of an underlying form). Different information, possibly encountered or constructed at different times, will indicate to the learner the correct values for different features.

Setting a feature in a lexicon is permanent; the learner will only set a feature once it has concluded for certain what the value of that feature is. Frequently

7 Kager (1999: 333–336) proposed using a heuristic for inconsistency to tell the learner when to try altering underlying form hypotheses. He discussed this in the context of an older approach to learning in Optimality Theory, Robust Interpretive Parsing / Constraint Demotion (Tesar 1998a, Tesar and Smolensky 1998). Tesar et al. 2003 proposed using MRCD to directly detect inconsistency in a set of stored winner–loser pairs, and alter underlying forms to resolve that inconsistency, with the set of underlying forms to possibly alter restricted to those actually appearing in the inconsistent winner–loser pairs. The approach described in this book is somewhat different, using inconsistency detection via MRCD with respect to a feature already targeted by the learner as possibly settable. The present approach attempts inconsistency detection after it has decided to try setting an underlying feature, while the prior approaches attempted to alter underlying features after detecting inconsistency.
during learning the learner will test certain values of unset features in constructed inputs; in such circumstances, the feature has been assigned a value in a particular input. Assigning a feature value is a temporary action and applies to features in inputs, while setting a feature value is a permanent action and applies to features in underlying forms in the lexicon.

This approach works with the representational theory: it treats features as fundamental constructs of linguistic theory, not merely as a taxonomic scheme for unitary segments. It will be occasionally convenient to denote forms (underlying forms, input, and outputs) in terms of the sets of feature values embodied (in more traditional terms, as feature matrices). For the Stress/Length system, each vowel has two features. When forms are denoted as feature values, the feature values for each vowel will be presented in the order (stress, length). An underlying form that would have been denoted /pá/ in the previous chapter could also be denoted /+,–/, the feature values +stress and –long. The output denoted [pá:ka] can also be denoted [(+,+)(–,–)]; each notation will be used where convenient. A feature in the lexicon that has not yet been set will be designated with “?” Unset features will only be present in underlying forms in the present work: complete inputs and outputs always have values assigned to all features. An underlying form with the feature value –stress and an unset length feature would be denoted as /–,?/.  

6.5.2 Setting a single unset feature
In some instances, inconsistency detection can be almost immediate. In the Stress/Length system, vowels can only surface long if they are underlyingly long; there are no constraints in this system that can compel a short vowel to lengthen. Thus, if a learner is evaluating the length feature for a vowel with respect to a word in which the vowel surfaces as long, an underlying short value for the length feature will immediately lead to inconsistency.

This is shown in (6.8). The situation concerns a word like r2s1 in L20, which surfaces [pá:ka]. Suppose the suffix vowel has had both of its underlying features set, and the suffix has a lexical entry of /–,–/. The root vowel has had its stress feature set to –stress, but its length feature has not been set, so the root has a lexical entry of /–,?/. The learner can test the possible values of the root vowel’s length feature, using the word. Testing the –long value of the feature involves constructing a hypothesized input for the word with the root vowel’s length feature set to –long; this yields the input /(–,–)(–,–)/, or /paka/. This input is then combined with the observed output, forming a hypothesized winner. If we pair that winner with a competing candidate, one with output [páka], we get the winner–loser pair shown in (6.8).
6.5 Setting underlying features via inconsistency detection

(6.8) Setting the root vowel to short results in inconsistency

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>/paka/</td>
<td>pacute:ka ~ pacuteka</td>
<td>L</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This pair has two constraints that prefer the loser, and no constraints that prefer the winner, a situation unsatisfiable by any constraint hierarchy. RCD will quickly discover this, ranking all of the constraints with no preference, and then being unable to rank either of the loser-preferring constraints. The –long value results in inconsistency, entailing that the feature must have the value +long in the lexicon. The related winner–loser pair with the root vowel length set to +long is shown in (6.9). This winner–loser pair is satisfied by any hierarchy in which IDENT[length] dominates NoLong.

(6.9) Setting the root vowel to long is consistent

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>/pa:ka/</td>
<td>pacute:ka ~ pacuteka</td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Many feature instances, however, cannot be set solely on the basis of a single word in isolation. Ranking information from other words is often crucial to detecting inconsistency. Consider the word r1s3 in L20, where the word surfaces as [paká], in which s3 has a lexical entry of /?,-/ with the stress feature unset, and r1 has a lexical entry of /–,–/. If s3 is assigned the value –stress in the input for r1s3, yielding input /paka/, the result is consistent, even when the phonotactic ranking information for L20 is included. There are grammars consistent with the phonotactic ranking information which admit the mapping /paka/ → paká. The tableau in (6.10) shows the phonotactic ERCs for L20 in their canonical form as given in (5.89), along with the key winner–loser pair for r1s3.

(6.10) s3 assigned –stress is consistent (at this point)

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>pacuteka</td>
<td>pacute:ka</td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pacuteka</td>
<td>paká</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1s3 /paka/</td>
<td>pacuteká ~ pacuteka</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If we consider the value +stress for s3, the result is also consistent. This is no surprise, as the resulting mapping, /paká/ → paká, is an identity candidate and was already included during phonotactic learning. The tableau in (6.11)
shows the phonotactic ERCs for L20, along with the key winner–loser pair for rls3 with s3 assigned +stress. Given only this information about r1s3 and the phonotactic ranking information, then, the stress feature of s3 cannot be set by inconsistency detection; both values of the feature are consistent.

(6.11) s3 assigned +stress is consistent (at this point)

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td></td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1s3</td>
<td>/paká/</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The situation is different, however, if the learner has some knowledge about other words of the language. Suppose the learner has also observed the surface form [paká] for r1s1, and has determined that the underlying form for r1 is /pa/, or /–,–/, and the underlying form for s1 is /-ka/, or /–,–/. In other words, both are set underlingly to –stress and –long. This knowledge justifies an additional winner–loser pair, shown in (6.12) along with the winner–loser pairs from (6.10).

(6.12) s3 assigned –stress is now inconsistent

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td></td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1s3</td>
<td>/paká/</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1s1</td>
<td>/paká/</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Now, inconsistency can be detected; the third and fourth ERCs directly contradict each other. It isn’t hard to see why. The third and fourth ERCs, winner–loser pairs for r1s3 and r1s1, both involve the same input form, /paká/, but are trying to map that input to different outputs. Because r1s3 and r1s1 have different outputs, there must be something different about their inputs to give rise to the difference in the outputs. This hints at the important role of contrast in the setting of underlying forms, a point that will be discussed in greater detail in Chapter 7. Because r1s3 and r1s1 contrast on the surface, they must contrast underlingly. Furthermore, because the root is the same in both words, the
underlying contrast has to be between the underlying form of s1 and the under-
lying form of s3. Suffix s3 has to be set to +stress, so that it contrasts with s1,
which has already been set to –stress.

In light of the computed inconsistency of the feature value –stress for s3,
the stress feature of s3 can be set to +stress. Using that value avoids the
inconsistency, as shown in (6.13): IDENT [stress] now prefers the winner in the
winner–loser pair for r1s3 (the third ERC), and by accounting for that ERC it
frees up MAIN LEFT to account for the fourth ERC.

(6.13) s3 assigned +stress is still consistent

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>påka</td>
<td>paká</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1s3 /paká/</td>
<td>paká ~ påka</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1s1 /paka/</td>
<td>påka ~ paká</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Inconsistency detection can use information about the grammar (both ranking
and underlying forms) obtained from other words to set underlying features.
This is a straightforward use of the same algorithm, MRCD, used to learn
ranking information. Technically, it evaluates an underlying feature value in a
word by checking the ranking implications of that feature value for consistency
with other known ranking conditions. In the phonological theory, it is the
constraint ranking that determines which surface forms are grammatical for
which underlying forms. In the learning theory, it is information about the
constraint ranking that is the basis for the reasoning between the observed
surface forms and underlying feature values.

### 6.5.3 Multiple unset features

In the examples of the previous section, the learner had already set all but one
feature and used inconsistency detection to set the final one. Attempts to set
a feature typically occur in the context of uncertainty about what some of the
other feature values are. Because feature values can interact, the evaluation of
the possible values for one underlying feature can be affected by what values
are chosen for other underlying features.

Inconsistency detection can be applied in such circumstances by considering
the possible combinations of values for unset features. Recall the phonotactic
ranking information for L20, repeated here.
The phonotactic ranking information for language L20

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
<th>ML</th>
<th>MR</th>
<th>NoLONG</th>
</tr>
</thead>
<tbody>
<tr>
<td>pąka</td>
<td>pą:ka</td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
<td>L</td>
</tr>
<tr>
<td>pąka</td>
<td>paká</td>
<td>W</td>
<td></td>
<td>L</td>
<td>L</td>
<td></td>
</tr>
</tbody>
</table>

Now we want to reconsider the word r2s1 surfacing as [pą:ka] in L20, assuming all underlying features for both r2 and s1 are unset. The learner can test the possible inputs for this word for consistency. One input is guaranteed to be consistent: the fully faithful input, the one used in phonotactic learning. But others could be consistent as well. In terms of feature values, the output is [(+,+)(-,–)]: the root syllable is stressed and long, while the suffix syllable is unstressed and short. There are a total of four underlying features, each with two possible values, there are $2^4 = 16$ possible combinations of values for the underlying features of the word. The consistency of each of the sixteen possibilities is shown in (6.14).

Output [pą:ka] for r2s1; underlying feature values are listed in order (stress, long)

<table>
<thead>
<tr>
<th>r2 UF values</th>
<th>s1 UF values</th>
<th>Consistent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>– – /pa/</td>
<td>– – /-ka/</td>
<td>no</td>
</tr>
<tr>
<td>– + /pa:/</td>
<td>– – /-ka/</td>
<td>yes</td>
</tr>
<tr>
<td>+ – /pá/</td>
<td>– – /-ka/</td>
<td>no</td>
</tr>
<tr>
<td>– + /pá:/</td>
<td>– – /-ka/</td>
<td>yes</td>
</tr>
<tr>
<td>– – /pa/</td>
<td>– + /-ka:/</td>
<td>no</td>
</tr>
<tr>
<td>– + /pa:/</td>
<td>– + /-ka:/</td>
<td>yes</td>
</tr>
<tr>
<td>+ – /pá/</td>
<td>– + /-ka:/</td>
<td>no</td>
</tr>
<tr>
<td>+ + /pá:/</td>
<td>– + /-ka:/</td>
<td>yes</td>
</tr>
<tr>
<td>– – /pa/</td>
<td>+ + /-ká:/</td>
<td>no</td>
</tr>
<tr>
<td>– + /pa:/</td>
<td>+ + /-ká:/</td>
<td>yes</td>
</tr>
<tr>
<td>+ – /pá/</td>
<td>+ + /-ká:/</td>
<td>no</td>
</tr>
<tr>
<td>+ + /pá:/</td>
<td>+ + /-ká:/</td>
<td>yes</td>
</tr>
<tr>
<td>– – /pa/</td>
<td>– + /-ká:/</td>
<td>no</td>
</tr>
<tr>
<td>– + /pa:/</td>
<td>– + /-ká:/</td>
<td>yes</td>
</tr>
<tr>
<td>+ – /pá/</td>
<td>– + /-ká:/</td>
<td>no</td>
</tr>
<tr>
<td>+ + /pá:/</td>
<td>– + /-ká:/</td>
<td>yes</td>
</tr>
</tbody>
</table>
The consistency of each local lexicon is determined using MRCD, with a support that already contains the phonotactic ranking information. The consistent inputs constitute (at this point of learning) the viable inputs, and are listed themselves in (6.15).

(6.15) Consistent inputs for r2s1 (given phonotactic ranking information)

<table>
<thead>
<tr>
<th>r2 UF values</th>
<th>s1 UF values</th>
<th>Consistent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>– +</td>
<td>/pa:/</td>
<td>– – /-ka/</td>
</tr>
<tr>
<td>+ +</td>
<td>/pá:/</td>
<td>– – /-ka/</td>
</tr>
<tr>
<td>– +</td>
<td>/pa:/</td>
<td>– + /-ka:/</td>
</tr>
<tr>
<td>+ +</td>
<td>/pá:/</td>
<td>– + /-ka:/</td>
</tr>
<tr>
<td>– +</td>
<td>/pa:/</td>
<td>+ – /-ká/</td>
</tr>
<tr>
<td>+ +</td>
<td>/pá:/</td>
<td>+ – /-ká/</td>
</tr>
<tr>
<td>+ +</td>
<td>/pá:/</td>
<td>+ + /-ká:/</td>
</tr>
</tbody>
</table>

The correct input for the word must be one of these possibilities. Therefore, anything that is true of all of these possibilities must be true of the correct input. The learner can observe the feature values for each feature and see if any of the features have the same value in all of the possibilities. In this case, the length feature for r2 is set to +long in every consistent input. Based upon this, the learner can set the underlying length feature for r2 to be +long.

Notice that the reasoning is not based on considering every possible input in which r2 has the value +long. In fact, one of the possible inputs for r2s1, /(- +)(+ +)/, or /pa:ká:/, is inconsistent. The inconsistency of this particular input does no harm to the learner’s reasoning: the learner only needs to be concerned with the consistent inputs. What is relevant is that in every consistent input, the root r2 is set to +long; that there are also one or more inconsistent inputs with r2 set to +long is hardly surprising (they could be inconsistent for reasons having nothing to do with r2 at all).

Inconsistency detection at this point allows the underlying form for r2 to be set to +long. It does not provide a basis for the learner to set the stress feature of r2, or either of the features for s1. If these features are to be set, the learner must either consider the morphemes in other contexts, or else obtain more ranking information and then reconsider the word r2s1. More ranking information increases the potential for detecting inconsistency.

The learner is not retaining all of the information gathered from these computations. What the learner retains from the processing of r2s1 above is the set value of r2’s length feature to +long. The other three features remain unset at this point. The learner is not retaining the fact that the input /(- +)(+ +)/ is
inconsistent, even though that fact rules out a combination of values of the unset features. This learner uses only feature setting as its memory for information it has gleaned about the lexicon; conditional relationships among the possible values of unset features are not retained.

6.5.4 Multiple words and local lexica

Section 6.5.2 showed the stress feature for s3 being set via consideration of the word r1s3, given some ranking information from the word r1s1. In that example, only the stress feature for s3 was unset; in particular, all of the other underlying features for r1 and s1 had already been set, so the full input for r1s1 was known. Section 6.5.3 showed that the learner is capable of setting a feature for a word with more than one unset feature. An example is now given where the learner considers r1s3 and r1s1 simultaneously, where both words have unset features in the underlying forms for their morphemes. The lexicon at the start of the example is given in (6.16). Root r1 has been set to –long, and suffix s3 has been set to –long; for L20, these can be determined via inconsistency detection on individual forms, along the lines illustrated in Section 6.5.3. The stress feature of r1 and stress feature of s3 and both features of s1 are as yet unset.

\[(6.16) \text{r1 /?,–/ s1 /?,?/ s3 /?,–/}\]

To use inconsistency detection, we wish to consider the possible inputs for the outputs. This means considering the different combinations of values for the unset features. This must be done across the morphemes included in all of the words under simultaneous consideration. In this case, the learner is simultaneously considering r1s3 and r1s1, so the learner must attend to the unset features of r1, s1, and s3.

A set of underlying forms for just the morphemes appearing in a particular set of words, with assigned values for all of the features, is called a local lexicon. The lexicon is local to the set of words in question. If the learner is simultaneously considering r1s3 and r1s1, then a relevant local lexicon is one containing fully specified underlying forms for morphemes r1, s3, and s1. A local lexicon for a single word is simply a set of underlying forms for the morphemes of the word. The concept of local lexicon reifies the focus of the learner to only those parts of the overall lexicon that are relevant to the words currently under consideration.

For the example involving r1s3 and r1s1, there is a separate local lexicon for each combination of possible values for the unset features. There are currently four unset features among the three morphemes, so there are a total of sixteen
local lexica. All sixteen are shown in (6.17), along with an indication of whether or not each local lexicon is consistent with the surface forms for the two words and the phonotactic ranking information. Note that the two features that have already been set, the length features for r1 and s3, only appear with their set value in the local lexica.

(6.17)  Local lexica for r1, s3, and s1, evaluated with r1s3 [paká] and r1s1 [páka]

<table>
<thead>
<tr>
<th>r1 UF values</th>
<th>s3 UF values</th>
<th>s1 UF values</th>
<th>Consistent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>/pa/</td>
<td>--</td>
<td>/-ka/</td>
</tr>
<tr>
<td>--</td>
<td>/pa/</td>
<td>--</td>
<td>/-ka:/</td>
</tr>
<tr>
<td>--</td>
<td>/pa/</td>
<td>--</td>
<td>/-ka/</td>
</tr>
<tr>
<td>--</td>
<td>/pa/</td>
<td>+</td>
<td>/-ka:/</td>
</tr>
<tr>
<td>--</td>
<td>/pa/</td>
<td>+</td>
<td>/-ka/</td>
</tr>
<tr>
<td>+</td>
<td>/pa/</td>
<td>--</td>
<td>/-ka:/</td>
</tr>
<tr>
<td>+</td>
<td>/pa/</td>
<td>+</td>
<td>/-ka:/</td>
</tr>
<tr>
<td>+</td>
<td>/pa/</td>
<td>+</td>
<td>/-ka/</td>
</tr>
<tr>
<td>+</td>
<td>/pa/</td>
<td>+</td>
<td>/-ka:/</td>
</tr>
</tbody>
</table>

The consistent local lexica only are shown in (6.18), for ease of inspection.

(6.18)  Consistent local lexica for r1s3 and r1s1

<table>
<thead>
<tr>
<th>r1 UF values</th>
<th>s3 UF values</th>
<th>s1 UF values</th>
<th>Consistent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>--</td>
<td>/pa/</td>
<td>+</td>
<td>/-ka:/</td>
</tr>
<tr>
<td>--</td>
<td>/pa/</td>
<td>--</td>
<td>/-ka:/</td>
</tr>
<tr>
<td>+</td>
<td>/pa/</td>
<td>--</td>
<td>/-ka:/</td>
</tr>
<tr>
<td>+</td>
<td>/pa/</td>
<td>--</td>
<td>/-ka:/</td>
</tr>
</tbody>
</table>

Of the four unset features, two of them take on a single value in all of the consistent local lexica: the stress features of the suffixes (the length features
for r1 and s3 have a single value because they have already been set). Suffix
s3 is +stress in every consistent local lexicon, and suffix s1 is –stress in every
consistent local lexicon. Therefore, the learner may set both of these features
in its actual lexicon.

Notice that the learner is unable to set the stress feature for r1 on the basis
of this pair of words. Yet the learner is able to determine the values of the
stress features for s3 and s1, without knowing what the correct value of the
stress feature for r1 is. The presence of r1 in both of the words is important,
despite the uncertainty about its underlying form, because whatever underlying
form is adopted for a particular local lexicon, it must be the same underlying
form for r1 in both words r1s3 and r1s1. This is the source of the additional
power the learner gains by evaluating both words simultaneously, as opposed
to separately (one at a time): whatever ranking commitments have to be made
in order for a particular underlying form for r1 to work with r1s3, the same
underlying form for r1 and the same ranking commitments also have to work
for r1s1, in order for joint consistency to be sustained.

The uncertainty about the underlying stress feature of r1 reflects uncertainty
about the ranking; there are two different analyses lurking behind the consistent
local lexica. Consider first the possibility of r1 being –stress underlingly. The
ranking consequences in the case where s1 is underlyingly –long are shown
in (6.19) in the form of ERCs, along with the phonotactic ERCs. For r1s3,
s3 is +stress while r1 is –stress, so faithfulness to stress decides in favor of
stress on the suffix (phonotactic learning ensures that IDENT[stress] dominates
both MAINLEFT and MAINRIGHT). For r1s1, both r1 and s1 are –stress
underlyingly, so faithfulness is indifferent to stress position (both are equally
unfaithful), and the decision passes to the alignment constraints, mandating that
MAINLEFT dominate MAINRIGHT. The corresponding constraint hierarchy
is reflected in the order of the constraint columns in (6.19).

(6.19) r1 as –stress is consistent when MAINLEFT \gg MAINRIGHT

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NO LONG</th>
</tr>
</thead>
<tbody>
<tr>
<td>pāka</td>
<td>pā:ka</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pāka</td>
<td></td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1s3 /paká/</td>
<td>paká ~ pāka</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1s1 /paka/</td>
<td>pāka ~ paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Consider next the possibility of r1 being +stress underlingly. The ranking
consequences in the case where s1 is –long are shown in (6.20), again in
the form of ERCs along with the phonotactic ERCs. For r1s3, both r1 and s3 are +stress underlyingly, so faithfulness is indifferent to stress position (both are equally unfaithful), and the decision passes to the alignment constraints, mandating that MainRight dominate MainLeft. For r1s1, r1 is +stress while s1 is –stress, so faithfulness to stress decides in favor of stress on the root. The corresponding constraint hierarchy is reflected in the order of the constraint columns in (6.20).

(6.20) r1 as +stress is consistent when MainRight \(\gg\) MainLeft

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>MR</th>
<th>ML</th>
<th>IDENT[length]</th>
<th>NoLONG</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>pá:ka</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1s3</td>
<td>/páká/</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1s1</td>
<td>/páká/</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In both of the analyses, one of the two words is decided by faithfulness to underlying stress, and the other is decided by default stress alignment. The two analyses differ in their default stress position (determined by the relative ranking of MainRight and MainLeft), and in the underlying stress feature value for r1. They share an underlying contrast in stress in the suffixes, one which must be expressed with s3 as +stress and s1 as –stress.

The reason that there are four consistent local lexica, rather than two, is that suffix s1 also has an unset length feature. Suffix s1 only appears in one of the words, r1s1. In r1s1, the suffix is unstressed on the surface, and the existing ranking information permits WSP to be ranked at the top, allowing the suffix to surface as short when it surfaces as unstressed, regardless of what its underlying length feature value is. Either value of the length feature for s1 is consistent, independent of the value of r1’s stress feature.

The stress feature for r1 could be set if the learner obtained further ranking information, the ranking relation between MainRight and MainLeft. The next section explains how to obtain such information from words for which some underlying features remain unset.

6.6 Non-phonotactic ranking information

6.6.1 Ranking information in local lexica

We’ve already seen how a learner can use MRCD to obtain ranking information on the basis of a fully specified winner. For phonotactic ranking information,
the learner is able to translate an observed output into a fully specified winner by adopting an input identical to the output. For non-phonotactic ranking information, however, things are more complicated: the inputs must be consistent with the underlying forms for the morphemes of the word. The learner could wait until all underlying features for all the morphemes of a word have been set before trying to gain further ranking information from that word. But that will significantly delay the learning of non-phonotactic ranking information. Even worse, there are some features that can only be set once certain non-phonotactic ranking information has been obtained: the additional ranking information is needed in order to force inconsistency for the incorrect values of those features.

Merchant (2008) proposed an alternative approach, inspired by the local lexicon approach to setting underlying features. Just as underlying features can be set when they have the same value in all consistent local lexica, Merchant proposed that non-phonotactic ranking information can be adopted when it holds for all consistent local lexica. Further, this can be done partly in combination with the setting of underlying forms. Given a pair of words to learn from, the learner determines which local lexica are consistent. The consistency of a local lexicon is determined using MRCD, which may construct and accumulate winner–loser pairs in the process of demonstrating consistency. For each local lexicon, those accumulated winner–loser pairs are the additional non-phonotactic ranking information entailed by that local lexicon; they are already calculated in the process of determining consistency. The learner determines what underlying features can be set by examining the consistent local lexica themselves, to see what features have the same value in each. The learner can also examine the additional ranking information entailed by each local lexicon and extract the ranking information that is common to all.

Extracting the ranking information in common to the ERCs for each local lexicon is more complex than extracting the underlying feature values that are in common for each local lexicon. It is not a simple matter of seeing if a given ERC is present in the added ERCs for each local lexicon. The full details of Merchant’s procedure for extracting non-phonotactic ranking information are beyond the scope of this book, but a sketch is given here.

### 6.6.2 The join operation

Given a selection of ERCs, one from each ERC set, in which a given constraint prefers the loser, the ranking information in common to that selection of ERCs can be computed using the lattice-theoretic join operation on the ERC
entailment lattice (Merchant 2008). Recall that ERC-A is ordered before ERC-B if and only if ERC-A entails ERC-B. Given a set of ERCs, the join of those ERCs will be the most informative ERC that is separately entailed by each of the factor ERCs. It is the equivalent of taking the logical OR of each of the ERCs in the set.

The join of a selection of ERCs may be computed component wise, separately for each constraint (it is like the fusion operation in this respect). For each constraint, the join of the values is derived from the order defining entailment: L < e < W. The join of any set of values is the highest value in the set with respect to this order. This is illustrated in (6.21).

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERC1</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>e</td>
<td>e</td>
<td>L</td>
</tr>
<tr>
<td>ERC2</td>
<td>W</td>
<td>e</td>
<td>L</td>
<td>e</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>ERC1 ⊕ ERC2</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>e</td>
<td>e</td>
<td>L</td>
</tr>
</tbody>
</table>

If a constraint prefers the loser in every ERC, then it also prefers the loser in the join. For each other constraint, if it prefers the winner in at least one of the ERCs in the selection, then it prefers the winner in the join; otherwise, it is indifferent (e). The join, therefore, concludes that the constraints preferring the loser in all of the ERCs must be dominated by at least one of the constraints that it conflicts with in at least one of the selected ERCs.

The join of a set of ERCs is separately entailed by each ERC in the set. This can be seen by observing that the join can be reached from any of the factor ERCs by a combination of L-retraction and W-extension. In (6.21), constraint C5 has a join value of e, which is reached from the L of C5 in ERC2 by L-retraction. Constraint C2 has a join value of W, which is reached from the e of C2 in ERC2 by W-extension. Constraint C3 has a join value of W, which is reached from the L of C3 in ERC2 by a combination of L-retraction and W-extension.

The join operation can be understood in logical terms. In (6.22), ERC A2 states that C2 must dominate C3. ERC B1 states that C1 must dominate C3. If these two ERCs come from the two possible local lexica, then at least one of them must be true. That is expressed with a logical OR, as shown in (6.23). The entailed form shown in (6.23) is precisely the content of the join of the two, with C1 and C2 preferring the winner, C3 preferring the loser, and C4 having no preference.
The join of two ERCs

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2</td>
<td>e</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>B1</td>
<td>W</td>
<td>e</td>
<td>L</td>
</tr>
<tr>
<td>A2 ⊙ B1</td>
<td>W</td>
<td>W</td>
<td>L</td>
</tr>
</tbody>
</table>

(C2 ≫ C3) OR (C1 ≫ C3) which entails (C2 OR C1) ≫ C3

The join operation may appear similar to the fusion operation, but the two are quite different. A comparison of the two is shown in (6.24), with the differences in bold. The join is W-dominant (any value joined with W is W), while fusion is L-dominant. The join can be defined in terms of the entailment order on the three values for a constraint: L < e < W. Fusion, by contrast, is determined by the order e < W < L, which is neither the entailment order nor its reverse.

Comparing join and fusion

<table>
<thead>
<tr>
<th>ERC1</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
<th>C5</th>
<th>C6</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERC2</td>
<td>W</td>
<td>e</td>
<td>L</td>
<td>e</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>ERC1 ⊙ ERC2</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>e</td>
<td>e</td>
<td>L</td>
</tr>
<tr>
<td>ERC1 ∘ ERC2</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td>e</td>
<td>L</td>
<td>L</td>
</tr>
</tbody>
</table>

Extracting shared ranking information

The following example illustrates the extraction of shared ranking information. Suppose a learner has two consistent local lexica, and the two local lexica yield the two sets of ERCs listed in (6.25) and (6.26).

ERC set A

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>W</td>
<td>L</td>
<td>e</td>
</tr>
<tr>
<td>A2</td>
<td>e</td>
<td>W</td>
<td>L</td>
</tr>
</tbody>
</table>

ERC set B

<table>
<thead>
<tr>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>B1</td>
<td>W</td>
<td>e</td>
<td>L</td>
</tr>
</tbody>
</table>
C1 and C4 prefer no losers in any of the ERCs. ERC set A has an ERC where C2 prefers the loser, but ERC set B does not, so there is no shared ranking information concerning the domination of C2. Each ERC set has an ERC in which C3 prefers a loser, A2 and B1, so those two ERCs can be selected to find the shared ranking information concerning the domination of C3. The join of A2 and B1 is shown in (6.27).

(6.27) The join of A2 and B1: (C1 OR C2) \geq C3

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2</td>
<td>e</td>
<td>W</td>
<td>L</td>
<td>e</td>
</tr>
<tr>
<td>B1</td>
<td>W</td>
<td>e</td>
<td>L</td>
<td>e</td>
</tr>
<tr>
<td>A2 \oplus B1</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td>e</td>
</tr>
</tbody>
</table>

Notice that the shared ranking information extracted here is not identical to any of the ERCs in either ERC set A or ERC set B. Simply looking for identical ERCs in each of the ERC sets is inadequate; there are no ERCs in common between set A and set B. This algorithm uses the logic of Optimality Theory to find that ranking information which is separately entailed by each ERC set for each local lexicon.

The learner could apply the join to A1 and B1, although it won’t accomplish much. This is shown in (6.28). The join is a trivial ERC, trivially true, because it doesn’t require that anything be dominated. None of the constraints assigns an L to both ERCs, so none of the constraints has a value of L in the join. If a join is to be non-trivial, there must be at least one constraint with a value of L for every ERC in the set. Merchant proposed a procedure that capitalizes on this observation to intelligently select sets of ERCs to join. Because the learner wants information that is jointly entailed by all of the local lexica, each ERC set should contain one ERC from the list for each local lexicon. The learner should only bother applying the join to such a set if there is at least one constraint that all of the ERCs assign L to.

(6.28) The join of A1 and B1 is trivially true

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td>W</td>
<td>L</td>
<td>e</td>
<td>e</td>
</tr>
<tr>
<td>B1</td>
<td>W</td>
<td>e</td>
<td>L</td>
<td>e</td>
</tr>
<tr>
<td>A1 \oplus B1</td>
<td>W</td>
<td>e</td>
<td>e</td>
<td>e</td>
</tr>
</tbody>
</table>
6.7 The Contrast Pair and Ranking (CPR) algorithm

The Contrast Pair and Ranking algorithm, or CPR, uses a combination of phonotactic learning, inconsistency detection, and extraction of shared ranking information to learn phonologies from morphologically segmented outputs (Merchant 2008). This section outlines how CPR puts those pieces together and discusses some of the computational properties of the algorithm.

CPR first performs phonotactic learning, by applying MRCD, using the mark-over-faith-edness bias of BCD, to the words individually, to obtain phonotactic ranking information. Once phonotactic learning is complete, CPR evaluates all available words using morpheme identity information. For each morpheme, it examines the surface forms and notes any features that do not alternate across contexts. Non-alternating features are set underlyingly at this stage with values identical to their (single) surface value. This stage is known as initial lexicon construction. The primary motivation for initial lexicon construction is to reduce the number of unset features prior to the evaluation of local lexica. Since the number of local lexica for a given word or set of words grows exponentially in the number of unset features, setting non-alternating features beforehand can result in a huge reduction in the amount of computation required by CPR if some of the features in the data are non-alternating. This points to a significant idealization assumed by CPR: prior to the initiation of non-phonotactic learning, the learner assumes it has stored and morphologically analyzed all relevant data (this is discussed further in Section 6.8).

The rest of the algorithm relates to the data in terms of contrast pairs (Tesar 2006a, 2006b). A contrast pair is a pair of words that differ in only one morpheme. The two different words feature two distinct morphemes in the same morphological environment (all the other morphemes are the same for both words). Examples are two different roots, each with the same suffixes, or two words with different suffixes attached to the same stem.

The concept of contrast pair originated in work attempting to use contrast relations to set underlying features. The starting idea is that if two words surface non-identically, then there must be some (at least one) differences between their inputs that are responsible for the differences on the surface. Focusing on the surface differences between the words could provide some clues as to how they differ underlyingly. Choosing a pair of words that differ in only a single morpheme restricts the possible differences in the inputs to just the differing morphemes. If a morpheme is shared between two words, then it must have the same underlying form in both words, even if it surfaces differently in the two words. A contrast pair allows the learner to focus only on the distinct
morphemes for the underlying differences that distinguish the surface forms of
the two words.

Recall the pair of words evaluated together in Section 6.5.4: r1s3 and r1s1. The words surface as shown in (6.29).

(6.29) r1s3 [paká]  r1s1 [páka]

The two surface forms differ in stress on both the first and second syllables, that is, on both the root and the suffix. However, both words have the same morphological root, r1. Therefore, the underlying difference(s) responsible for the surface differences in stress must be with the underlying forms of the suffixes s3 and s1.

CPR adapts the contrast pair as the domain over which local lexica are constructed and evaluated. The local lexicon space for a contrast pair won’t be much larger than that for a single word, as only one additional morpheme is involved. The local lexicon space for the word r1s1 involves the possible underlying forms for r1 and s1; the local lexicon space for the contrast pair r1s3 and r1s1 involves the possible underlying forms for r1, s1, and s3.

CPR works by repeatedly constructing contrast pairs, one at a time. As each contrast pair is constructed, it is evaluated. First, inconsistency detection is used to set any settable underlying features for the morphemes of the contrast pair. Then, ranking extraction is performed over the consistent local lexica. Once that is finished, CPR moves on to the next contrast pair.

CPR’s method of choosing which contrast pair to evaluate next is computationally motivated. The number of local lexica for a contrast pair grows exponentially in the number of unset features. CPR examines the available contrast pairs and determines which one has the fewest unset features, selecting it for evaluation. It then works through the contrast pairs in order by the number of unset features, adjusting whenever a feature is set. The idea is to focus early on pairs that require less computational effort to evaluate, because they have fewer local lexica. If those contrast pairs allow some underlying features to be set, then other contrast pairs containing the morphemes with the newly set features will have fewer local lexica to evaluate.

CPR continues until no more underlying features can be set by any constructible contrast pairs. It then assigns “default” values to any underlying features that have not yet been set, where the default value is the unmarked feature value for the type of feature.

CPR can be effective, but it is not guaranteed to succeed for all OT systems that meet its basic conditions. While the precise conditions necessary
Learning with paradigmatic information

to ensure the success of CPR are not known, the problematic cases that have been uncovered to date involve paradigmatic subsets, which are discussed further in Chapter 8. See Merchant (2008) for further discussion of the scope of CPR. Further discussion of the formal properties of contrast pairs, and their significance for learning, is given in Chapter 7.

6.8 Computational issues for CPR

Computationally, CPR is a huge improvement over the baseline of evaluating all possible grammars described in Section 6.2. The number of possible lexica alone is exponential in the number of features in all of the morphemes of the language. CPR, by contrast, evaluates local lexica, and the size of any particular set of local lexica is exponential only in the number of features in a contrast pair, which is the number of features in a word plus one additional morpheme. The number of contrast pairs will be a function of the number of morphemes of various morphological classes, not the number of features in each morpheme, and will be modest in comparison.

The difference is noticeable even in a very small system, like the Stress/Length system. As explained in Section 6.2.4, the number of possible lexica alone is 65,536, and the number of grammatical hypotheses is 47,185,920. There are “4 choose 2,” or six, distinct pairs of roots that can be contrasted, and for each pair there are four suffixes that could be the morphological environment, yielding $6 \times 4 = 24$ root-contrasting contrast pairs. Identical reasoning yields twenty-four suffix-contrasting contrast pairs, for a total of $24 + 24 = 48$ possible contrast pairs. Each contrast pair has three morphemes, each with two features, for a total of six features, and therefore $2^6 = 64$ local lexica in the worst case where no features have been set. If we add up the number of local lexica for each possible contrast pair, not allowing for the effects of any actual setting of feature values, we get $48 \times 64 = 3,072$ local lexica to possibly evaluate.

Despite the simplifications that make this figure a significant overestimate of the number of local lexica CPR will evaluate, 3,072 local lexica is many times less than 65,536 possible lexica. Further, a local lexicon will take much less time to evaluate than a complete lexicon for the language (two words for a contrast pair vs. sixteen words for an entire lexicon). The difference between the two approaches will increase vastly when applied to larger, more complex linguistic systems.

While it is the case that with CPR the number of local lexica is “only” exponential over the number of unset features in a contrast pair, the fact remains
that it is still exponential over the number of unset features in a contrast pair. Both underlying feature setting and ranking extraction compute over all local lexica for a given contrast pair. This gets computationally expensive when scaling up to more complex linguistic systems than our simple Stress/Length system. For a single word containing five segments, with each segment having six unset features, the number of local lexica is $2^{(5*6)} = 2^{30} = 103,737,411,824$. The cognitive plausibility of searching all local lexica for such a word (which is still quite modest relative to actual human languages) must be questioned.

CPR does attempt to reduce the number of unset features up-front with initial lexicon construction (setting features that do not alternate). But that has its own problems. Most notably, initial lexicon construction assumes that the learner has already observed and remembered each morpheme in a sufficiently representative range of morphological environments that it can reliably determine which features ever alternate. Given the variable frequency at which different paradigm members can occur, the learner might have to wait quite a while before it could have much confidence that it had seen a sufficient number of forms. This is made worse by the fact that initial lexicon construction has to occur initially in order to have any computational impact, so getting initial lexicon construction to significantly cut down on the number of local lexica to be evaluated comes at the cost of significantly delaying non-phonotactic learning until long after the learner has begun identifying morphemes.

6.9 The map

The advent of CPR constitutes significant progress in the learning of non-phonotactic information. However, there is room for improvement, in particular with respect to the amount of computation required. The next chapter will show how a learner can capitalize on output drivenness to greatly improve the computational efficiency of learning, while building on key elements of CPR. Specifically, output drivenness can convert the process of setting underlying features for a word from exponential in the number of unset features to linear in the number of unset features for individual words, and tremendously shrink the potential for exponential growth for contrast pairs. The exponentially expensive process for extracting shared ranking information can be eliminated outright, with non-phonotactic ranking information obtainable through the evaluation of a small number of easily determined candidates. The selection of contrast pairs can be done much more efficiently, avoiding spending a lot of effort on

8 There always is.
some kinds of contrast pairs that will provide no new information. Also, initial lexicon construction can be dispensed with completely, allowing the learner to make incremental progress as more morphologically segmented data become available. All of these benefits to learning will be shown to follow from output drivenness.
Chapter 5 and Chapter 6 provided essential background concepts on phonological learning. This chapter demonstrates the impact of output-driven maps on phonological learning.

A key concept in phonological learning is contrast. While a generic notion of contrast has been a part of phonological theory throughout its history, the specific characterization of contrast used here is different in some respects from a more traditional characterization. This alternative characterization of contrast is motivated largely by Richness of the Base. When the maps of an OT linguistic system are all output-driven, there are additional consequences for how contrast is realized in grammar. The alternative characterization of contrast is the subject of Section 7.1.

Section 7.2 introduces relative similarity lattices, which are the link between the relative similarity relation of output-driven maps and the structured spaces of possible underlying forms for morphemes and words. Section 7.3 shows how the structure imposed by relative similarity lattices on local lexica allows a learner to search the space of local lexica with far greater efficiency than the local exhaustive search employed by the CPR algorithm discussed in Chapter 6.

Sections 7.4 and 7.5 explain how output drivenness can be exploited in the learning of underlying feature values and the learning of non-phonotactic ranking information. The conception of contrast introduced in Section 7.1 is essential to an understanding of both. Section 7.6 then explores the processing of contrast pairs, where the learner processes a related pair of words simultaneously. Many of the languages in the Stress/Length system require the learner to process a contrast pair in order to successfully learn the language, and this involves simultaneous reasoning over both underlying feature values and non-phonotactic ranking information.

Error-driven learning processes words (in general, grammatical outputs) strictly one at a time, reasoning only between a hypothesized grammar and a single word at any given time. The use of contrast pairs in learning, like
MRCD’s storage of a support of winner–loser pairs, is a way in which the architecture of learning developed in this book departs from traditional error-driven learning. The significance of these departures is discussed in Section 7.7.

The ideas concerning the exploitation of output drivenness in learning are pulled together in a learning algorithm called the Output-Driven Learner (ODL). Section 7.8 gives a compact statement of the basic ODL, and Section 7.9 gives a detailed illustration of the ODL learning language L20 of the Stress/Length system.

The ODL described in Section 7.8 successfully learns all but two of the languages of the Stress/Length system. The two languages for which it fails exhibit a phenomenon that is here labeled paradigmatic subsets. Chapter 8 discusses the nature of paradigmatic subsets, why they cause difficulty for the basic ODL, and provides an extension of the ODL which can overcome the difficulties posed by paradigmatic subsets. The final version of the ODL proposed in Chapter 8 successfully learns all of the languages in the Stress/Length system.

7.1 Contrast with Richness of the Base

7.1.1 Contrastive for an input

The principle of Richness of the Base has significant consequences for the notion of contrast in phonology. Because the space of inputs is universal, systematic language-specific patterns of contrast cannot be accounted for by language-specific restrictions on the possible inputs. The core of phonological contrast must reside in the input–output map. Contrastive properties emerge from the way representations pattern in the map, rather than being inherent properties of specific representational units, such as a (language-specific) inventory of phones, or of distinctive features. This leads almost inevitably to a basic notion of contrast as a relation between entire inputs. Non-identical inputs are contrastive if and only if they yield non-identical outputs. The contrastive status of other (lower) units of representation derive from the basic one between entire inputs.

This view of contrast is quite different from traditional ones which focus on language-specific inventories of distinctive elements. The traditional model, with roots extending well back into structuralist phonology, posits contrast as a property of elements of the inputs independent of the input–output map of the

1 For a history of contrast in phonology, see Dresher 2009.
phonology. Differences between inputs are contrastive by definition, but the contrast can be obscured by the input–output map via neutralizing processes. The explanation for the systematic properties of a language is split between conditions on allowable inputs and the input–output map.

Richness of the Base forces the entire explanation for systematic properties into the map. Contrastive properties that vary cross-linguistically, at least, cannot be inherent in the inputs themselves; they must be determined by the map. Two inputs are contrastive in a language not if the inputs are non-identical, but if their corresponding outputs are non-identical. Contrasts aren’t obscured by the phonological map, they are determined by the phonological map.

I here propose an alternative view of the contrastive status of representational units in the input, one that is more compatible with Richness of the Base. For concreteness, the discussion of the contrastive status of features focuses for the most part on binary features (Jakobson et al. 1952/1963). For a given grammar and a given full input, a particular input feature instance is **contrastive for that input** if and only if changing the underlying value of the feature, while holding the rest of the input unchanged, results in a different output. That is, if the input is changed by changing the value of that single feature instance, and the resulting grammatical candidate for that new input has a different output from the output assigned to the original input, then the input feature instance that distinguishes the two inputs is contrastive for the original input. If the new input’s grammatical candidate has an output that is identical to the output of the grammatical candidate of the original input, then the feature instance is not contrastive for the original input.

The property of being “contrastive for an input” is a property that may hold of input feature instances. It is a separate property for each feature instance of each input; there is no prior expectation that all instances of a given type of feature in an input will be contrastive for that input. Contrastiveness is a property of feature tokens, not feature types. In a language in which obstruents may be voiced or voiceless word-initially but must be voiceless word-finally, for the input /tat/ with grammatical candidate /tat/ → [tat], we should expect that the voicing feature of the initial input segment is contrastive for that input, but the voicing feature for the final input segment is not contrastive for that input. The fact that both input segments are /t/ is irrelevant, just as the fact that both segments surface as [t] is irrelevant. What matters is that the language includes the additional mappings /dat/ → [dat] and /tad/ → [tat]. It is also true that, for the input of each of these latter two mappings, the voicing feature of the initial segment is contrastive, while the voicing feature of the final segment is not contrastive.
To further appreciate the tokenness of “contrastive for an input,” consider the map for L20, given in (5.2). Recall that L20 has lexical stress, with stress on the initial syllable by default, and long vowels shorten in unstressed position. The map for L20 includes /páka/ → [páka]. Is the stress feature of the first syllable of /páka/ contrastive? No, because changing it yields the input /paka/, and /paka/ → [páka], the same output. By default, stress is initial. Is the stress feature of the second syllable of /páka/ contrastive? No, because changing it yields the input /páká/, and /páká/ → [páka], again the same output. But, if we consider the input /paka/, is the stress feature on the second syllable contrastive? Yes, because /páká/ → [paká], a different output from /paka/ → [páka]. The stress feature of the second syllable of input /páká/ is a different feature instance than the stress feature of the second syllable of input /paka/. Similarly, the stress feature of the second syllable of input /paká/ is contrastive. In general, the contrastiveness of a particular feature instance may be dependent on the values assigned to every other feature in the input.

Some theories organize features into asymmetric structures. Such theories include feature geometry (Clements 1985, Sagey 1986), Government Phonology (Kaye et al. 1985), Dependency Phonology (Anderson and Ewen 1987), Radical CV Phonology (Hulst 1995, 1996), and the Contrastive Hierarchy (Dresher 2009).2 One motivation for organizing features is to create dependencies in contrast among features. Whether a given occurrence of a feature is contrastive can depend upon the values of one or more other features in the same segment. The contrastiveness of a feature is thus dependent on context, the context of other features within the segment. The proposal above for “contrastive for an input” significantly extends the contextual domain that can condition contrast beyond a single segment, to the entire input.

7.1.2 Contrastive for a morpheme

Morpheme identity allows us to equate a feature instance for a morpheme’s underlying form with an instance of the same feature type in each input containing that morpheme. Thus, a feature instance of the underlying form of a morpheme is contrastive for that morpheme if the feature instance is contrastive for at least one input in which the morpheme’s underlying form appears. A feature instance of an underlying form for a morpheme is not contrastive for
that morpheme if changing its value alone would not change the output for any input in which the morpheme’s underlying form appears.

Relatedly, two underlying forms can be said to be contrastive if there is at least one word containing one of the underlying forms such that, if the other underlying form is substituted for the first in the input, the resulting input has a different output. Two underlying forms that are not contrastive in this sense can be said to exhibit identical phonological behavior: they behave the same phonologically in all possible phonological contexts.

7.1.3 Contrast in output-driven maps
Output-drivenness, when combined with the principle of Richness of the Base, has further consequences for the contrastive status of features. One consequence is that each feature instance which is contrastive for an input must either surface faithfully or belong to an input segment lacking an output correspondent. If an input feature is unfaithfully realized in the output, then the differing input/output values constitute a disparity in a grammatical candidate. By the definition of output-driven maps, changing the input to remove the disparity (by changing the input feature value to match its output correspondent) necessarily results in an input which maps to the same output. Therefore, the feature instance is not contrastive for that input.

Relative similarity is defined by correspondence between identical disparities across the entire input. This suggests that the representational units most suitable for expressing contrast should be the units utilized to express the inventory of disparities. With respect to an inventory of disparities like the one in (2.38), determined from a purely segmental IO correspondence with segments described by binary features, this has an interesting consequence: both features and segments are units of contrast, but in different ways. Features are the units of contrast for identity disparities, while entire segments are the units of contrast for insertions and deletions. The feature is the relevant unit when comparing inputs that differ on the values of a feature of (input–input) corresponding segments. An entire segment (not a feature) is the relevant unit when comparing inputs that differ on the presence/absence of a segment (one input has a segment with no correspondent in the other input).

Assuming that IO correspondence is only defined with respect to segments, a segment of an input is contrastive for that input if and only if the removal of

---

3 As stated earlier, this assumes binary-valued features. If a feature has three or more values, the situation is more complicated: it could be the case that two values of the feature contrast in a given environment, but one of those values does not contrast with the third value of the feature.
that entire segment from the input results in an input mapping to a different output. A segment can fail to be contrastive for an input in two ways. If, in the grammatical candidate for the original input, the input segment in question has no output correspondent, then it is clearly not contrastive for that input: removing the segment from the input results in the removal of a disparity, which by output-drivenness is guaranteed to map to the same output. If, in the original candidate, the input segment in question has an output correspondent, then the input segment is not contrastive if the input with the segment removed maps to the very same output form, with the removed segment’s former output correspondent either lacking an input correspondent (inserted by the grammar into the output) or corresponding to a different input segment.

This characterization of contrastive segments for insertion and deletion has a couple of quirks worth noting. If an input /ta/ maps to [ta], and the input /a/, with the “t” removed, maps to [a], then the segment “t” in /ta/ is contrastive: removing it results in a different output. However, there isn’t a straightforward way in which it is a contrastive segment of the contrasting input: there is no “t” in /a/, as its contrastiveness lies in its absence. This lacks the symmetry inherent in contrastive features, where the contrasting inputs both have the feature in question, just different values for the feature.

Further, it is possible for a segment in the input to be non-contrastive, because removing it from the input results in the same output, but for a feature of that input segment to be contrastive, because keeping the segment but changing the feature value results in a different output. If input /ketbu/ has output [kebu], under the generalization that obstruents delete rather than appear in codas, then the “t” in /ketbu/ is non-contrastive: input /kebu/ will have output [kebu] also. However, if coda nasals don’t delete but instead assimilate in place, then the nasal feature of the “t” in /ketbu/ is contrastive, because changing its value results in the input /kenbu/, which will have output [kembu], clearly a distinct output from [kebu]. See also the related discussion in Section 2.4.6.

### 7.2 Relative similarity lattices

Every grammar constructible within the Stress/Length system generates an output-driven map. All of the constraints are output-driven preserving, and Gen is correspondence uniform. The only type of disparity under consideration is the feature value disparity, so the candidate with the maximum number of disparities for an input will be the candidate where the output and the input disagree on the value of every feature of every vowel. With two features per vowel, the maximum number of disparities is twice the number of vowels,
four disparities for two syllable words. Recall from Section 2.2 that the relative similarity relation can be partitioned into relative similarity orders, one for each output (because two candidates can be related in the relation only if they share the same output). In fact, so long as all of the features are binary and independent, the relative similarity order for a given output forms a lattice. I will take advantage of this, and separately present and discuss the lattices for different individual outputs, referring to the relative similarity order involving candidates with \textit{out}, as the \textbf{relative similarity lattice} for \textit{out}. 

The relative similarity lattice for the output paká: is shown in Figure 7.1. Each node represents a candidate, with the text within each node indicating the input for that candidate (all candidates have output paká:). The output [paká:] has two vowels, for a total of four features, so there are $2^4 = 16$ possible inputs for this output, and thus sixteen nodes in the lattice. If one candidate is above another, it means that the higher candidate has greater internal similarity than the lower.

It can be easier to visually process the relational structure of a relative similarity lattice if feature matrices are used. Such a diagram is given in Figure 7.2, where the four features are given in the order [root-stress root-length suffix-stress suffix-length].
The top node in the lattice represents the input with the greatest similarity to the output: the identity input. The top candidate has zero disparities. The candidates immediately below the top one each have a single disparity with the output (one such candidate for each feature). This continues down the order until the bottom is reached: the candidate in which the input differs from the output on every feature of every vowel.

A learner can benefit from the knowledge that the language being learned is output-driven. Specifically, the learner can capitalize on the entailment relations between different candidates to arrive at conclusions about both underlying forms and the ranking without having to generate and evaluate all of the relevant possible underlying forms.

Output-driven maps are defined by an entailment relation: $akx \text{ entails } bmx$. If $in_a$ maps to $out_x$, then $in_b$ must map to $out_x$ also. Importantly, this can be extended from actuality to possibility: if $in_a$ possibly maps to $out_x$, then $in_b$ possibly maps to $out_x$ also. That is, if the state of the learner’s knowledge is consistent with at least one grammar in which $in_a$ maps to $out_x$, then it is also consistent with at least one grammar (the same grammar(s)) in which $in_b$ maps to $out_x$. With respect to a relative similarity lattice, if a particular candidate in the lattice is explicitly determined to be possibly optimal, then one can automatically conclude that every candidate above it in the lattice (every
7.3 Limiting lexical search in output-driven maps

The benefit of output drivenness for setting features can be illustrated by considering the length feature of suffix s4 in L20. The output of word r1s4 is [paká:]. Recall that Merchant’s CPR algorithm evaluates every local lexicon (all of the possible assignments of feature values to the unset features of the underlying forms of the morphemes). Assuming for the moment that none of the underlying features for r1 and s4 have yet been set, the word r1s4, taken on its own, has four unset features, for a total of sixteen local lexica. For those local lexica (out of the sixteen) that prove to be consistent with the learner’s current knowledge, if the underlying value of the length feature for s4 proves to have the same value in all of them, then the learner knows that the feature can be set to that value. The consistent local lexica constitute the only viable combinations of underlying forms with respect to the learner’s knowledge at that point; one of them must be the correct one, and if the length feature of

4 While CPR works exclusively with contrast pairs, the idea of evaluating all local lexica can be applied to any set of one or more words.
s4 has the same value in all of them, then it must be assigned that value no matter which of the local lexica is ultimately correct.

CPR evaluates every local lexicon because it is attempting to set as many of the unset underlying features as possible. If we wish to focus only on the length feature of s4, we can be a bit more selective. Idempotency ensures that an input which is featurally identical to the output (no disparities) forms an optimal candidate: if any input maps to the output, the zero-disparity candidate does. It follows from this that because the suffix is long in the output of r1s4, s4 is assigned the value +long in the input of the zero-disparity candidate. Thus, there exists at least one local lexicon with s4 assigned +long that is consistent. The learner will be able to set s4 to +long if it turns out that all consistent local lexica have s4 assigned +long. To determine that, it is sufficient to evaluate all of the local lexica with s4 assigned –long. If all of them prove to be inconsistent, then the learner can set s4 to +long. If at least one of them proves to be consistent, then the learner does not yet have enough information to confidently set the length feature of s4.

For the space of local lexica formed just by the morphemes r1 and s4 (with no features yet set), half of the local lexica have s4 set to –long, and half have s4 set to +long. If we are focusing just on the length feature of s4, we cut the space of local lexica to be evaluated in half; only those with s4 set to –long need to be evaluated for consistency. While this is a reduction, it doesn’t solve the problem of exponential growth. The number of local lexica grows exponentially in the number of unset features, and cutting each quantity of local lexica in half does not halt exponential growth.

The learner can do much better by exploiting output drivenness (rather than mere idempotency). Specifically, the learner can test the length feature of s4 by constructing an input with just a single disparity relative to that output, a disparity in the length feature of the suffix. That input is /paká/. The learner then constructs the candidate /paká/→[paká:], with the output of r1s4 and only a disparity in the suffix length. The relative similarity lattice for [paká:] is given in Figure 7.3. This is the same lattice as in Figure 7.1, but with a shaded sublattice consisting of all the candidates that have a disparity with the output for the suffix length feature: they all have s4 underlyingly –long. The single disparity candidate thus has a subset of the disparities of all of the candidates in the sublattice; /paká/→[paká:] has greater similarity than any candidate for this output with a disparity in the length of the suffix. The single disparity candidate is the top element in the shaded sublattice: if it proves to be inconsistent and therefore non-optimal, then all of the candidates in the sublattice are non-optimal, given that the map is output-driven.
Figure 7.3. Setting s4 to $+$long. This is the relative similarity lattice for the output of r1s4, $[paká:].$ The shaded sublattice contains all candidates with s4 underlyingly $-$long.

If the learner determines that $/paká/ \rightarrow [paká:]$ cannot be optimal, then the learner may conclude that none of the candidates in the entire sublattice can be optimal. The only remaining candidates for the output have s4 underlyingly $+$long. Thus, the learner can conclude that the underlying form for s4 is set to $+$long.

The benefit is one of computational efficiency. Even though half of the possible underlying forms have the suffix $-$long underlyingly, the learner does not need to evaluate all of them, only the one at the top of the sublattice. An entire half of the relative similarity lattice is effectively collapsed to one form for computational purposes. The same benefit applies to every other unset feature of the word. Only candidates with a single disparity need to be tested, ones with a single disparity concerning an unset feature. This converts exponential search into linear search: the number of possible underlying forms is exponential in the number of features, but the number of forms to actually be tested is linear in the number of features (precisely one per unset feature).

In an output-driven map, for a single word on its own, an underlying feature value that matches its surface value will always be consistent: there will always be at least one consistent input for the word in which the underlying value
of the feature matches its surface realization (the fully faithful input comes to mind). This remains true of unset features for restricted sets of inputs in which some underlying features have had their values set (possibly to values other than their surface realizations). An input is **viable** at this point if it does not conflict with any of the values for features that have already been set. For any underlying feature not yet set, there will be at least one consistent input in the range of viable inputs with the as-yet-unset feature assigned the value that matches its surface realization. This follows directly from the definition of output-driven map: given that there must be at least one consistent input in the space (a correct one), then in that consistent input, the underlying value of the relevant feature either matches the surface realization already, or changing it to match the surface realization (removing that identity disparity) also yields a consistent input. A consequence of this, for binary features, is stated in (7.1).

(7.1) For an output-driven map, inconsistency detection can only set the underlying value of a binary feature with respect to a word in which that value is faithfully realized on the surface.

Inconsistency detection works by determining what a feature cannot have as its value. Because a value matching the surface realization will always be consistent, inconsistency can only be detected for a value not matching the surface realization. For a binary feature, eliminating the sole non-matching value leaves the matching value as the only remaining possibility, allowing the learner to set the feature to that value.\(^5\)

A particular feature for a particular morpheme can be set only with respect to a word in which the correct underlying value matches the surface value, and the incorrect underlying value is inconsistent (yields an input which maps to a different output in the target grammar). The reverse will never happen, as it would contradict output drivenness. If a word is going to provide the basis for setting one of its features, then that feature will be set to the value of its surface realization in the word.

\(^5\) For suprabinary features, the situation is potentially more complicated. If a feature is to be set on the basis of a single word, then all values of the feature except the surface realized value must be inconsistent for that word; that is fully consistent with the spirit of (7.1). The alternative would be to eliminate different incorrect values of the feature with different words, in which case the correct underlying value of the feature would not necessarily have to be realized on the surface (for any of the words). The details of a learner making use of the alternative approach to suprabinary features will not be further pursued here.
7.4 Phonotactic contrast and underlying feature values

Recall the conception of contrastive feature introduced in Section 7.1: an input feature is contrastive for a word if changing the feature’s value results in a distinct output. If one is in possession of the correct ranking, one could determine if a given feature of a particular input is contrastive by changing its value in the input and seeing if the grammatical output for that modified input is different from the grammatical output for the original input.

A language learner does not start out in possession of the full correct ranking. However, a language learner could reasonably be expected to possess some ranking information when contemplating underlying feature values. As shown in Chapter 5, the learner can obtain phonotactic ranking information without needing to know any underlying forms. Given some ranking information, a learner can potentially determine the underlying values for some features by using inconsistency detection.

When dealing with output-driven maps, an underlying feature can be tested with a single word by evaluating the consistency of a single candidate: the candidate in which the input matches the output except for the value of the single feature being evaluated (it forms the only disparity in the candidate). If the optimality of that candidate is inconsistent with what is already known about the ranking, then the feature being tested cannot have the mismatched value; it must have an underlying value matching its surface realization in that word. Notice that the learner need not know at this point what output the target grammar would map the modified input to; for purposes of setting the underlying value of the feature, it suffices to know that the input would be mapped to some other output (not the one at hand).

When a feature is set in this way, it is because the learner’s ranking information guarantees that there is a contrasting word with an input that differs only in the relevant feature (the one being set). Phonotactic ranking information is based largely on word-level contrast; that was the moral of Section 5.7.2. The underlying features that can be set on the basis of phonotactic ranking information can be labeled phonotactically contrastive features. For such a feature, not only is only one value for the feature consistent with the language, but only one value for the feature is consistent with the phonotactic ranking information.

When we speak of phonotactically contrastive features, any such “feature” is limited in scope to a specific word. There is no way to identify a feature in one word with a feature in another word without use of paradigmatic information: it requires knowing that parts of two different words are the same morpheme.
and must share the same underlying form. Phonotactic ranking information
and phonotactically contrastive feature values are the extent of what can be
learned purely from word contrast, without use of paradigmatic information.
Frequently, these are insufficient to fully determine the language: there will
be non-phonotactic ranking information, and underlying features that are con-
trastive but not phonotactically contrastive, and paradigmatic information is
necessary to learn them.

The setting of a phonotactically contrastive feature is illustrated in (7.2). The
first two rows are the ERCs for the phonotactic ranking information for L20,
as previously given in (5.89); the ERCs have been relabeled Phonotactic1 and
Phonotactic2 to make it easier to distinguish the purely phonotactic ranking
information from other ranking information, in this and several upcoming
illustrations. The third ERC in the tableau is the test ERC for the length feature
of the second syllable of the word [paká]. The underlying value of the length
feature has the value +long, contrary to its surface realization. The result is
inconsistency: the third ERC is inconsistent with the first. The phonotactics have
determined that Ident[length] dominates NoLong. The only other constraint
that can possibly penalize long vowels, WSP, does not do so in this case because
the long vowel appears in a stressed syllable in the competing candidate, [paká:].
If the last syllable of the word is underlyingly +long, it could only surface as
–long and stressed if NoLong dominates Ident[length], directly contradicting
the phonotactic ranking information.

(7.2)  The second syllable of [paká] cannot be underlyingly +long

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
<th>Ident[stress]</th>
<th>Ident[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonotactic1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phonotactic2</td>
<td></td>
<td>L</td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>/paká:/</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>[paká] ~ [paká:]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The role of word contrast in the setting of this underlying feature value can
be seen by examining the phonotactic ranking information that led to the
inconsistency. The first ERC came about by the fact that both [páka] and [pá:ka]
are surface forms of the language. The surface contrast between these words
requires that Ident[length] dominate NoLong. That ranking relation then
applies equally to all syllables that are stressed on the surface. Because there is
no other constraint in the system that is violated by either length value in surface-
stressed syllables, all syllables stressed on the surface will faithfully reflect
their underlying length. Thus, the phonotactic ranking information requires
that [paká] have a faithful realization of length for the final vowel. The second syllable of [paká] must be underlingly –long.

In concluding that the second syllable of [paká] must be underlingly –long, the learner is implicitly recognizing that there must be a word-level contrast between the inputs /paká/ and /pakáː/. If the only phonotactic ranking information the learner had were Phonotactic1, then the learner would not know if the output of /pakáː/ was [pakáː] or not. The alternative would be an output in which the final syllable was not stressed, like [páka].

What the learner knows for certain is that there must be a word-level contrast: /pakáː/ cannot surface as [paká].

If this feature setting takes place before the learner has morphologically analyzed the word, then the learner does not yet know that this word consists of, for instance, root r1 and suffix s3; it is just a word. The learner has determined that for the word [paká], the length feature of the final vowel is contrastive and must have the value –long. However, once the learner gains access to the appropriate paradigmatic information and analyzes this word into root r1 and suffix s3, they will be able to assign the feature value –long to suffix s3, extending the feature value to other words analyzed as containing s3.

Recall the surface forms of L20: {páka pá:ka paká pakáː}. For each of these surface words, the values of the phonotactically contrastive features, based on phonotactic ranking information, are shown in (7.3). For L20, each surface form has one phonotactically contrastive feature: the length feature of the stressed syllable.

(7.3) The set features are phonotactically contrastive

<table>
<thead>
<tr>
<th>Surface word</th>
<th>Feature Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>/? – ? ?/</td>
</tr>
<tr>
<td>pá:ka</td>
<td>/? + ? ?/</td>
</tr>
<tr>
<td>paká</td>
<td>/? ? ? –/</td>
</tr>
<tr>
<td>pakáː</td>
<td>/? ? ? +/</td>
</tr>
</tbody>
</table>

Once paradigmatic information is taken into account, the learner can construct a lexicon in which the length feature is set for every underlying syllable that surfaces as stressed in at least one word. This is shown in (7.4). The length

6 In (7.2), this mapping would be inconsistent with Phonotactic2, due to preservation of the underlying stress.
features remain unset for suffixes s1 and s2; those features are not phonotactically contrastive, because s1 and s2 never surface as stressed.

(7.4) Lexicon for L20 with the phonotactically contrastive features set

<table>
<thead>
<tr>
<th>r1</th>
<th>??,?/</th>
<th>r2</th>
<th>??,+/</th>
<th>r3</th>
<th>??,?/</th>
<th>r4</th>
<th>??,+/</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>??,?/</td>
<td>s2</td>
<td>??,?/</td>
<td>s3</td>
<td>??,?/</td>
<td>s4</td>
<td>??,+/</td>
</tr>
</tbody>
</table>

It may seem odd that none of the stress features are set. After all, we have a phonotactic contrast between [páka] and [paká]; how can stress not be phonotactically contrastive? The catch is that the learner can tell from those two surface forms that stress is contrastive somewhere; that is reflected in the phonotactic ranking information (Phonotactic2 in (7.2)). But the learner cannot determine for sure where that contrast is realized in the input, given only phonotactic information.

The surface form [páka] is grammatical in L20, so the learner knows that /páka/ → [páka]. What about the input /paka/, resulting from changing the stress feature of the first syllable? The learner cannot tell from phonotactic ranking information alone what the output is. The learner can tell that the output is not [paka], because that is not a valid output, but it cannot determine if the output is [páka] or some other valid output, like [paká]. Thus, the stress feature of the first syllable of /páka/ is not phonotactically contrastive. Similarly, the learner knows that /paká/ → [paká], but again cannot tell from phonotactic ranking information alone what the output of /paka/ is. The output for /paka/ cannot be both [páka] and [paká], so in the final grammar, at least one of the first syllable of /páka/ and the second syllable of /páká/ has a stress feature that is contrastive for that input (see also the discussion in Section 7.1.1). But the learner cannot determine which on the basis of phonotactic information, so neither one is phonotactically contrastive.

7.5 Morphemic alternation and non-phonotactic ranking information

Phonotactic learning operates solely on the basis of fully faithful winning candidates. Non-phonotactic ranking information is ranking information that can be defined only with respect to winning candidates that are not fully faithful. Winners with disparities are needed to obtain such ranking information.

Getting winners with disparities requires reliable knowledge about unfaithful aspects of inputs. This is possible once a learner becomes morphologically
Morphemic alternation and non-phonotactic ranking information

Once the value of an underlying feature for a morpheme has been set, the potential for finding non-phonotactic ranking information lies in other words containing that morpheme in which the feature in question is not faithfully realized (Tesar 2006a). In other words, there is potential for learning non-phonotactic ranking information when the underlying value is set for a feature that alternates across morphological contexts. Paradigmatic information is essential here: determining that a feature alternates requires morpheme identity to establish the relationship between a feature in one word and a feature in another.

Morphemic alternation is the key to learning non-phonotactic ranking information. A language in which no morphemes alternate can easily have contrastive features (in the form of phonotactically contrastive features), but it will not have any non-phonotactic ranking information accessible via inconsistency detection.

Once the learner sets the underlying value of a feature for a morpheme, it can examine that morpheme’s other surface realizations to see if the feature surfaces unfaithfully in another word. If the learner finds such a word, it can exploit output drivenness to efficiently determine if any new non-phonotactic ranking information is entailed. The learner can do this by applying MRCD to that word, using, as the winner, the candidate with an input in which all features set in the lexicon match their lexically set values, and all features not set in the lexicon match their surface realization for the current word.

Recall the example in Section 7.3, where suffix s4 was set to +long for L20. That feature was set on the basis of word r1s4, in which s4 surfaces as +long. In r3s4 [páka], on the other hand, s4 surfaces as short. The word r3s4 now provides an opportunity for the learner to learn non-phonotactic ranking information. Specifically, the learner wants to determine if there are any further ranking conditions necessary to ensure that s4 surfaces as short in r3s4, given that it is underlingly long.

The relative similarity lattice for r3s4 is shown in Figure 7.4. Because s4 has been set to be +long, none of the inputs that have s4 as –long are viable; the nodes for nonviable inputs are marked in the figure with shaded diamonds. The viable inputs, the ones with s4 as +long, are ovals. Notice that the viable nodes form a sublattice, with a top element. The form of the top element of the sublattice is predictable: it is the candidate in which all features unset in the lexicon match the surface form of the word. In the top element of the sublattice, the only disparities are those resulting from features that have been previously set. In Figure 7.4, the top element of the viable sublattice, corresponding to the...
candidate /páka:/ is non-phonotactic, because it is not fully faithful, and so it is an opportunity for the learner to obtain non-phonotactic ranking information.

The reasoning here is parallel to the reasoning that leads to the conclusion that fully faithful mappings for grammatical outputs are always grammatical candidates in phonotactic learning. If the learner has no lexical information
(no set features), then the entire relative similarity lattice for an observed surface word is viable. One of the nodes in the lattice must be grammatical, and if any of them are grammatical, then the top node is grammatical. That’s why phonotactic ranking information can be justifiably obtained from zero-disparity candidates for observed outputs. The same reasoning over relative similarity lattices applies to any sublattice based on viability (all and only the viable candidates). Lexical information can limit the viable candidates to a sublattice of the full relative similarity lattice, but the relations among the members of the (sub)lattice remain the same.

The learner can obtain ranking information from this mapping by using MRCD (just as with phonotactic ranking information). This is summarized in (7.5). The unfaithful mapping at the top of the viable sublattice, /páka:/[páka], is adopted as the winner. An appropriate loser is selected via production-directed parsing on the winner’s input, in this case yielding the candidate with output identical to the input.7 The winner and loser are listed in (7.5), and the resulting ERC is listed in the next row down.

(7.5) Non-phonotactic ranking information from r3s4

<table>
<thead>
<tr>
<th>/páka/</th>
<th>WSP</th>
<th>MAINL</th>
<th>MAINR</th>
<th>NO LONG</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>páka:</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERC</td>
<td>W</td>
<td>W</td>
<td></td>
<td></td>
<td>L</td>
<td></td>
</tr>
</tbody>
</table>

We can get a sharper sense of what new information has been obtained by combining this new ERC with a previously obtained ERC, one obtainable from purely phonotactic information. The relevant phonotactic ERC from (7.2) is shown, along with the non-phonotactic ERC just obtained, in (7.6). The phonotactic ERC expresses the observation that long vowels sometimes appear on the surface, and in this system that requires that IDENT[length] dominate NO LONG. Taking the fusion of the two ERCs, shown in the last row, reveals some of what the learner has obtained: WSP must dominate both NO LONG and IDENT[length]. The conclusion that the faithfulness constraint IDENT[length] must be dominated relies on the unfaithful element of the unfaithful mapping, the failure to faithfully realize the underlying length on the suffix.

7 This candidate is produced by production-directed parsing when used with a constraint hierarchy other than the one constructed by BCD. The construction of this hierarchy, and its justification, are discussed in Section 7.7.2.
Exploiting output drivenness in learning

(7.6) \[ \text{WSP} \gg \{\text{NoLong, Ident[length]}\} \]

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>MainL</th>
<th>MainR</th>
<th>NoLong</th>
<th>Ident[stress]</th>
<th>Ident[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>/pāka/</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phonotactic1</td>
<td>L</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fusion</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The new ERC, along with Phonotactic1, combine to provide a partial picture of the desired ranking: \[ \text{WSP} \gg \text{Ident[length]} \gg \text{NoLong} \]. The learner was able to obtain this information from the word r3s4 despite not knowing the complete input for the word (a consequence of not knowing the complete underlying forms for r3 and s4). Whenever a feature has been newly set in the lexicon, the learner can identify possible sources of additional ranking information by finding those words that do not faithfully realize the feature. Each such word can be tested by constructing an input in which all features not set in the lexicon are taken to have the same value as their output counterparts. Output drivenness ensures that these mappings are valid in the target language, and any additional ranking information required to ensure that these mappings are optimal may be adopted by the learner.

When the learner applies BCD to the phonotactic ranking information for L20, it generates a constraint hierarchy with WSP at the top. WSP is a markedness constraint that prefers no losers and in fact is never violated on the surface in L20, so BCD’s mark-over-faith-edness bias puts WSP at the top, over all of the faithfulness constraints. The alternation in length of suffix s4 has now allowed the learner to move the dominance of WSP over \text{Ident[length]} from an implicit bias (enforcing phonotactic restrictiveness) to an explicit requirement on the ranking in the form of a winner–loser pair (ensuring that the observed alternation occurs).

Computationally, the benefits relative to the CPR algorithm are similar to those provided for the learning of underlying feature values. In CPR, every possible local lexicon is separately evaluated for consistency. The possible local lexica are those consistent with the known lexical information; they are precisely the nodes of the viable sublattice. The inconsistent local lexica are discarded, while the consistent ones are noted along with the additional ranking information, for each consistent local lexicon, necessary to make it optimal. Ranking information extraction is then performed by CPR on all of the sets of additional ranking information, to determine the informational intersection (the ranking information required by each of the consistent local lexica). In addition
to the exponential growth in the number of local lexica (each of which must be separately evaluated), the ranking information extraction algorithm itself is computationally demanding.

By contrast, the exploitation of output-driven maps described here makes it possible to learn non-phonotactic ranking information by the same basic procedure used to learn phonotactic learning information. Instead of evaluating all local lexica to see which are consistent, output drivenness tells the learner that the one at the top of the viable sublattice must be consistent and in fact must be a grammatical mapping. There is no combinatorial growth here at all; only one candidate to be processed with MRCD.

Further, the top candidate of the viable sublattice will entail precisely the ranking information shared by the candidates of all the local lexica. To see this, first consider that the top candidate corresponds to one of the local lexica: if ranking information is separately entailed by all of the local lexica, then it must be entailed by the top one. Because the top candidate is entailed by all of the others, any ranking information that is entailed by the top candidate must also be entailed by the candidates for all of the other local lexica. The shared ranking information can be obtained without the join-based extraction procedure of CPR in output-driven maps, giving a huge computational advantage.

Morphemic alternation is the key to obtaining non-phonotactic ranking information because it is the source of information about unfaithful mappings. Once the underlying value of an alternating feature has been set, the words in which the set feature surfaces unfaithfully are guaranteed to be the product of unfaithful mappings. Faithful mappings are the source of phonotactic ranking information; unfaithful mappings are the source of non-phonotactic ranking information.

7.6 Contrast pairs

7.6.1 When one word isn’t enough

The ranking information for L20 based on phonotactic information and the alternation of phonotactically contrastive features is shown in (7.7). The ERC labeled “NonPhon1” is the learner’s first piece of non-phonotactic ranking information and was obtained based on the alternation of the length feature in suffix s4, as was shown in (7.5). The length feature of suffix s4 is a phonotactically contrastive feature, able to be set based solely on phonotactic ranking information.
Exploiting output drivenness in learning

(7.7) L20 ranking information, based on phonotactics and phonotactically contrastive features

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
<th>Ident[stress]</th>
<th>Ident[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonotactic1</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phonotactic2</td>
<td>L</td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NonPhon1</td>
<td>W</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Applying BCD to this list of winner–loser pairs results in the hierarchy in (7.8).

(7.8) \( WSP \gg Ident[stress] \gg \{MainLeft, MainRight\} \gg Ident[length] \gg NoLong \)

WSP is ranked at the top, because it is the only markedness constraint preferring no losers. It frees up the faithfulness constraint \( Ident[stress] \), but none of the other markedness constraints, so one of the faithfulness constraints must be ranked next. The choice of ranking \( Ident[stress] \) as the higher faithfulness constraint is decided by the fact that \( Ident[stress] \) frees up two markedness constraints (\( MainLeft \) and \( MainRight \)), while \( Ident[length] \) only frees up one (\( NoLong \)). That is a fluke occurrence here, stemming from the fact that the constraint system happens to have two markedness constraints sensitive exclusively to stress, but only one markedness constraint sensitive exclusively to length. The undominated position of WSP suggests that length and stress may interact, but doesn’t indicate how. A hierarchy with \( Ident[length] \) in the second stratum (below WSP), followed by \( NoLong \) and then \( Ident[stress] \), would define a different kind of map, but one with exactly the same phonotactic inventory; the difference in r-measure isn’t actually indicating any difference in phonotactic restrictiveness in this case.

Because phonotactic learning always uses an input identical to the output, the phonotactic input always has exactly one accent, and in just the right position. Phonotactic learning will ensure that \( Ident[stress] \) dominates \( MainLeft \) and \( MainRight \), so for those phonotactically well-formed inputs \( Ident[stress] \) will always decide the location of stress before \( MainLeft \) and \( MainRight \) get a chance to. This is why \( MainLeft \) and \( MainRight \) are in the same stratum in the phonotactic constraint hierarchy; the phonotactic data for this language do not provide any opportunities for \( MainLeft \) and \( MainRight \) to make any decisions in competitions, so there is (as yet) no basis for distinguishing their ranking.

The lack of information about the relative ranking of \( MainLeft \) and \( MainRight \) prevents the learner from being able to use inconsistency detection to set underlying stress feature values on the basis of individual words.
Consider the word r1s1, páka. The fully faithful input assigns a value of +stress to the underlying stress feature of r1, and the fully faithful input is guaranteed to be consistent. But if we set the underlying stress feature of r1 to –stress, the resulting local lexicon is still consistent: ranking \texttt{MainLeft} over \texttt{MainRight} satisfies the relevant ERCs, as shown in (7.9).

(7.9) The stress feature for r1 cannot be set on the basis of r1s1 alone

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>\texttt{Ident[stress]}</th>
<th>ML</th>
<th>MR</th>
<th>\texttt{Ident[length]}</th>
<th>\texttt{NoLong}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonotactic1</td>
<td></td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phonotactic2</td>
<td>W</td>
<td></td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NonPhon1</td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/paka/ páka</td>
<td>W</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

It is perhaps no surprise that assigning –stress to r1 is consistent, since that is the correct underlying value for r1 in L20. But things are no better when the word r1s3, paká, is evaluated in isolation, where r1 surfaces faithfully as –stress. If we set the underlying stress feature of r1 to +stress, the resulting local lexicon is still consistent: ranking \texttt{MainRight} over \texttt{MainLeft} satisfies the relevant ERCs, as shown in (7.10).

(7.10) The stress feature for r1 cannot be set on the basis of r1s3 alone

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>\texttt{Ident[stress]}</th>
<th>MR</th>
<th>ML</th>
<th>\texttt{Ident[length]}</th>
<th>\texttt{NoLong}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonotactic1</td>
<td></td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phonotactic2</td>
<td>W</td>
<td></td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NonPhon1</td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/paká/ paká</td>
<td>W</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Returning to word r1s1, we can instead attempt to set the stress feature of s1, but that will fare no better: ranking \texttt{MainLeft} over \texttt{MainRight} satisfies the relevant ERCs, as shown in (7.11).

(7.11) The stress feature for s1 cannot be set on the basis of r1s1 alone

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>\texttt{Ident[stress]}</th>
<th>ML</th>
<th>MR</th>
<th>\texttt{Ident[length]}</th>
<th>\texttt{NoLong}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonotactic1</td>
<td></td>
<td></td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phonotactic2</td>
<td>W</td>
<td></td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NonPhon1</td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/pák/ pák</td>
<td>W</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
For L20, the same reasoning applies to the stress features of all of the morphemes. For any word of the language, altering one underlying stress feature of the word will still allow the same output to be produced, consistent with the phonotactic ranking information, because the necessary ranking relation between $\text{MainRight}$ and $\text{MainLeft}$ can be adopted to assign stress in the attested location. Using inconsistency detection to set a stress feature on the basis of a single word requires knowing the relative ranking of the two constraints.

Setting some underlying features (those that aren’t phonotactically contrastive) requires non-phonotactic ranking information, getting non-phonotactic ranking information requires unfaithful mappings, and getting unfaithful mappings requires setting underlying features. These interdependencies might appear to constitute a vicious cycle, but that isn’t the case. For one thing, phonotactically contrastive features can alternate, and be the basis for some unfaithful mappings. More relevant for the stress features in L20 is the possibility of processing more than one word simultaneously. Specifically, the possibility of processing two words that share a morpheme, and in which a feature of the shared morpheme alternates. This is the case for the words r1s1 and r1s3 taken together: the words share the morpheme r1, and stress alternates on r1 between the two words. If the two words are processed together, then no matter which underlying value for the alternating feature (stress for r1) is used, it will be unfaithful in one of the words: there will be no solution available that is “fully faithful” for both words simultaneously. Processing a pair of words in which a feature alternates is a way to force unfaithful mappings to be contemplated, without (yet) knowing which unfaithful mapping is the correct one. That is the way to contend with the interdependencies of underlying feature values, non-phonotactic ranking information, and unfaithful mappings.

### 7.6.2 A disjunction of disparities

Simply putting the words r1s1 and r1s3 next to each other won’t cause any breakthroughs, but it sets the stage for inconsistency detection to succeed. The two possible values of the stress feature for r1 pose a disjunction: the feature is either $+$stress or $-$stress underlingly. Considering two words in which that feature alternates means that with either value there is a disparity, an unfaithful mapping of the stress feature of r1. The payoff comes if the learner can identify some other feature such that one value of the other feature is inconsistent with either disparity for r1’s stress feature. Because one of the two disjuncts must be true, something that is inconsistent with both disjuncts is inconsistent, period. Looking at a pair of words in which the stress feature of r1 alternates will not
allow the learner to set the stress feature of r1. But it will set the stage for the learner to set a different feature.

Before launching into what that other feature is and how it is set, it is worth examining more closely how r1s1 and r1s3 relate to each other. Each of the bottom two ERCs in (7.12) shows a comparison between a winner with the correct output for r1s1 (including initial stress) and a loser with final stress. The two ERCs differ in their inputs. The “fully faithful” ERC assigns +stress underlingly to r1, which is faithfully realized in the winner’s output. The “r1 unfaithful” ERC assigns –stress underlingly to r1, which is not faithfully realized in the winner’s output. Note that the fully faithful ERC is entailed by Phonotactic2: the change in the preference of MainLeft from L to W is a combination of L-retraction and W-extension. This is totally unsurprising: the fully faithful mappings are precisely what phonotactic ranking information is based on. The r1 unfaithful ERC, however, is not entailed by Phonotactic2 (nor by the combination of Phonotactic1, Phonotactic2, and NonPhon1). It contains a ranking commitment beyond those entailed by the phonotactic ranking information. Note further that the r1 unfaithful ERC does entail the fully faithful ERC, by W-extension for Ident[stress]. The winner for the fully faithful ERC has greater internal similarity than the winner for the r1 unfaithful ERC, so the optimality of the latter entails the optimality of the former (in fact, the latter is one step down the relative similarity lattice for r1s1 from the former). Adding the disparity (by assigning –stress to r1 underlingly) brings in the stronger ranking requirement, by removing the W for Ident[stress].

Each of the bottom two ERCs in (7.13) shows a comparison between a winner with the correct output for r1s3 (including final stress) and a loser with initial stress. The two ERCs differ in their inputs. The “fully faithful” ERC assigns –stress underlingly to r1, which is faithfully realized in the winner’s output. The “r1 unfaithful” ERC assigns +stress underlingly to r1, which is not faithfully realized in the winner’s output. Again, the fully faithful ERC is entailed by Phonotactic2, while the r1 unfaithful ERC is not. Adding the
Disparity (by assigning +stress to r1 underlyingly) brings in the stronger ranking requirement, by removing the W for IDENT[stress].

(7.13) Assigning the unfaithful value of r1’s stress feature in r1s3 neutralizes faithfulness

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ML</th>
<th>MR</th>
<th>NoLONG</th>
<th>Id[stress]</th>
<th>Id[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonotactic2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fully faithful (r1s3) /paká/ paká ~ páka</td>
<td>L</td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1 unfaithful (r1s3) /páká/ paká ~ páka</td>
<td>L</td>
<td>W</td>
<td></td>
<td>e</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The tableau in (7.14) shows the ERCs of r1s1 and r1s3 for the disjunct in which the alternating feature, stress in r1, is underlyingly +stress, causing a disparity in r1s3. The set of ERCs (which also includes the two phonotactic ERCs as well as the non-phonotactic ERC from (7.5)) is collectively consistent. The unfaithful realization of r1’s stress in r1s3 has neutralized the faithfulness constraint IDENT[stress], so MAINRIGHT ⇒ MAINLEFT must hold in order for r1s3 to surface correctly. This means that IDENT[stress] must be the deciding constraint in the ERC for r1s1: MAINLEFT cannot dominate MAINRIGHT, so IDENT[stress] has to.

(7.14) r1 underlyingly +stress: consistent, requires MAINRIGHT ⇒ MAINLEFT

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ML</th>
<th>MR</th>
<th>NoLONG</th>
<th>Id[stress]</th>
<th>Id[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonotactic1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phonotactic2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NonPhon1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fully faithful (r1s1) /páka/ paká ~ paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1 unfaithful (r1s3) /páká/ paká ~ paká</td>
<td>L</td>
<td>W</td>
<td></td>
<td>e</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The tableau in (7.15) shows the ERCs of r1s1 and r1s3 for the disjunct in which the alternating feature, stress in r1, is underlyingly −stress, causing a disparity in r1s1. The set of ERCs (which also includes the two phonotactic ERCs as well as the non-phonotactic ERC from (7.5)) is collectively consistent. The unfaithful realization of r1’s stress in r1s1 has neutralized the faithfulness constraint IDENT[stress], so MAINLEFT ⇒ MAINRIGHT must hold in order for r1s1 to surface correctly. This means that IDENT[stress] must be the deciding constraint in the ERC for r1s3: MAINRIGHT cannot dominate MAINLEFT, so IDENT[stress] has to.
7.6 Contrast pairs

(7.15) \( r_1 \) underlyingly –stress: consistent, requires **MainLeft \( \gg \) MainRight**

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
<th>Id[stress]</th>
<th>Id[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonotactic1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phonotactic2</td>
<td></td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NonPhon1</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>( r_1 ) unfaithful (( r_1s_1 ))</td>
<td>/paka/</td>
<td>páka ~ paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fully faithful (( r_1s_3 ))</td>
<td>/paká/</td>
<td>paká ~ páka</td>
<td>L</td>
<td>W</td>
<td></td>
<td>W</td>
</tr>
</tbody>
</table>

If the two words are processed together, then no matter which value of \( r_1 \)'s underlying stress feature is adopted, additional restriction will be imposed on the ranking. Increased restriction on the ranking means increased potential for inconsistency with other things, which can be exploited to set other underlying features.

7.6.3 *Restricted rankings create inconsistencies*

Consider the stress feature for the suffix \( s_1 \). No stress feature could be set on the basis of a single word using only phonotactic ranking information, and the stress feature of \( s_1 \) is no exception: as was shown in (7.11), both values of the stress feature for \( s_1 \) are consistent for the word \( r_1s_1 \) on its own. But the story changes if the stress feature for \( s_1 \) is tested while processing \( r_1s_1 \) and \( r_1s_3 \) simultaneously. The additional ranking relations imposed by the alternation of stress on \( r_1 \) will cause the incorrect underlying value for the stress feature of \( s_1 \) to be inconsistent.

Suffix \( s_1 \) surfaces as unstressed in \( r_1s_1 \), and is not present in \( r_1s_3 \), so an underlying value of –stress for \( s_1 \) will be consistent. Testing, with \( r_1s_1 \) and \( r_1s_3 \), the underlying value of stress for \( s_1 \) involves evaluating \( r_1s_1 \) and \( r_1s_3 \) with an underlying value of +stress for \( s_1 \): if the combination is inconsistent, then the learner can set \( s_1 \) to –stress underlyingly. Because the learner does not yet know the underlying value of the stress feature for \( r_1 \), and stress for \( r_1 \) alternates on the surface in the pair of words, the learner must consider both possible values for \( r_1 \)'s underlying stress feature. If an underlying value of +stress for \( s_1 \) is inconsistent for every case in the disjunction (here, any value of \( r_1 \)'s stress feature), then the learner can set the stress feature of \( s_1 \). The disjuncts in the disjunction (\( r_1 +stress \) vs. \( r_1 –stress \)) impose different additional ranking relations (opposite ranking relations, in this example: **MainRight \( \gg \) MainLeft** vs. **MainLeft \( \gg \) MainRight**), so it is important that both disjuncts be tested for inconsistency.

There are two disjuncts, so two pairs of mappings are evaluated for consistency. Each pair of mappings contains a mapping for \( r_1s_1 \) and a mapping for...
Exploiting output drivenness in learning

r1s3, evaluated together. Each pair has an underlying value of +stress for s1 (the feature the learner is attempting to set). The two pairs of mappings differ on the value of the underlying stress feature for r1 (the alternating feature). The remaining underlying features can be set to their (single) surface realization for this pair of words, as allowed for by output drivenness. The relevant ERCs for each disjunct, along with the previously derived ERCs, are shown in (7.16) and (7.17).

(7.16) With r1 assigned +stress, assigning +stress to s1 is inconsistent

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ML</th>
<th>MR</th>
<th>NoLONG</th>
<th>Id[stress]</th>
<th>Id[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonotactic1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phonotactic2</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NonPhon1</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1 unfaithful (r1s1) /páká/ páka ~ paká</td>
<td>W</td>
<td>L</td>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1 unfaithful (r1s3) /páká/ paká ~ páká</td>
<td>L</td>
<td>W</td>
<td>e</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(7.17) With r1 assigned –stress, assigning +stress to s1 is inconsistent

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ML</th>
<th>MR</th>
<th>NoLONG</th>
<th>Id[stress]</th>
<th>Id[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonotactic1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phonotactic2</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NonPhon1</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1, s1 unfaithful (r1s1) /paká/ páka ~ paká</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>fully faithful (r1s3) /páká/ paká ~ páká</td>
<td>L</td>
<td>W</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The two pairs of mappings, and the outcome for each, are summarized in (7.18).

(7.18) Inconsistency detected for contrast pair r1s1 and r1s3, using s1 with underlying value +stress

<table>
<thead>
<tr>
<th>r1</th>
<th>Mappings</th>
<th>Consistent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>~stress r1s1: /pa+ká/ → páká</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>r1s3: /pa+ká/ → paká</td>
<td></td>
<td></td>
</tr>
<tr>
<td>+stress r1s1: /pá+ká/ → páká</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>r1s3: /pá+ká/ → paká</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Both pairs of mappings are inconsistent. This means that an underlying value of +stress for s1 leads to inconsistency. The learner still doesn’t know which underlying stress value for r1 is correct, but it doesn’t matter for now; either way leads to inconsistency with +stress for s1. Thus, the learner is able to
7.6 Contrast pairs

set s1 to –stress via inconsistency detection, by processing the words r1s1 and r1s3 simultaneously.

The tableau in (7.19) shows ERCs for r1s3 and r1s1, both with and without disparities, for the case in which r1 is assigned +stress. The “fully faithful (r1s1)” and “r1 unfaithful (r1s3)” ERCs are as in (7.14). The “s1 unfaithful (r1s1)” ERC is as in (7.16). The introduction of a stress disparity for r1 in r1s3 changes IDENT[stress] from W to e in r1s3, and the introduction of a stress disparity for s1 in r1s1 changes IDENT[stress] from W to e in r1s1. Either disparity on its own is tolerable, but the combination is inconsistent. Given an r1 stress disparity, the addition of the s3 stress disparity is a step too far. The “r1 unfaithful (r1s3)” and “s1 unfaithful (r1s1)” ERCs make opposing requirements of the ranking relation between MAINLEFT and MAINRIGHT. Either of these disparity-caused ERCs can be tolerated alone, but not both at once. The combination of the disparity in the alternating feature (stress for r1) and the disparity in the feature being evaluated (stress for s1) results in inconsistency between the two disparity-containing ERCs.

(7.19) Disparities for r1 in r1s3 and s1 in r1s1 are inconsistent

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>fully faithful (r1s3)</td>
<td>paká/ paká ~ páka</td>
<td>L</td>
<td>W</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1 unfaithful (r1s3)</td>
<td>paká/ paká ~ páka</td>
<td>L</td>
<td>W</td>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fully faithful (r1s1)</td>
<td>paká/ paka ~ paka</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>s1 unfaithful (r1s1)</td>
<td>paká/ paka ~ paká</td>
<td>W</td>
<td>L</td>
<td>E</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The tableau in (7.20) shows the Phonotactic2 ERC, along with ERCs for r1s1 both with and without disparities, for the case in which r1 is assigned –stress. The “r1 unfaithful (r1s1)” ERC is as in (7.15). The “r1, s1 unfaithful (r1s1)” ERC is as in (7.17). The “s1 unfaithful (r1s1)” ERC is the consequence of imposing a stress disparity on s1 but not r1, within word r1s1. The ranking restriction imposed by “s1 unfaithful (r1s1)” is the same as that imposed by “r1 unfaithful (r1s1)”: the introduction of a stress disparity for the winner, and the co-occurrence removal of one from the loser, causes the violations of IDENT[stress] to balance out, so that IDENT[stress] has no preference between the candidates (an evaluation of e, rather than W). The “r1, s1 unfaithful (r1s1)” ERC shows what happens with word r1s1 when r1 is assigned –stress and s1 is assigned +stress (both unfaithful to the surface). Both stress features are unfaithful in the winner, and faithful in the loser, so IDENT[stress] now prefers the loser. Given the r1 stress disparity, the addition of the s1 stress disparity has changed IDENT[stress] from e to L. That is a step too far, where
consistency with the phonotactics is concerned: the “r1, s1 unfaithful (r1s1)” ERC is inconsistent with the Phonotactic2 ERC (the fusion of the two has three Ls and no Ws). The combination of the disparity in the alternating feature (stress for r1) and the disparity in the feature being evaluated (stress for s1) results in inconsistency between r1s1 and the phonotactic ranking information.

(7.20) Disparities for both r1 and s1 in r1s1 are inconsistent with the phonotactics

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
<th>Id[stress]</th>
<th>Id[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonotactic2</td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td>W</td>
</tr>
<tr>
<td>fully faithful (r1s1)</td>
<td>/páka/</td>
<td>páka ~ paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1 unfaithful (r1s1)</td>
<td>/paka/</td>
<td>páká ~ paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td>e</td>
</tr>
<tr>
<td>s1 unfaithful (r1s1)</td>
<td>/páká/</td>
<td>páká ~ paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td>e</td>
</tr>
<tr>
<td>r1, s1 unfaithful (r1s1)</td>
<td>/paká/</td>
<td>páká ~ paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td>L</td>
</tr>
</tbody>
</table>

The morphemic alternation of stress in r1 between r1s1 and r1s3 forces the learner to accept that there must be a stress disparity in one or the other. Processing both words of the alternation at once forces the learner to consider the additional ranking restrictions imposed by either underlying value for stress in r1. The learner is then able to use inconsistency detection to set the stress feature for s1 based upon its behavior in r1s1, because the unfaithful underlying value for s1 in r1s1 is inconsistent with the ranking restrictions imposed by either underlying stress value of r1 (via whichever of r1s1 and r1s3 is unfaithful to r1 in stress). The learner still doesn’t know the correct underlying value for the stress feature of r1, but is able to finesse this by determining the non-phonotactic ranking consequences of each value and testing both against the unfaithful underlying value of the stress feature of s1. The learner can use morphemic alternation to assemble a disjunction of possible non-phonotactic ranking restrictions, which collectively can be used with inconsistency detection to set features in the contrasting morphemes.

7.6.4 The roles of alternation and contrast in contrast pairs

It isn’t hard to see why both pairs of mappings in (7.18) are inconsistent; just look at the inputs for the two words in each pair. In both cases, the two input forms are the same, yet the mappings attempt to map them to different outputs. The differing outputs make clear that the words are phonologically contrastive, yet the inputs fail to register any phonological contrast. More specifically, the morphological difference between the two words lies in the suffixes, s1 and s3. Since the words differ in output, the difference must be due to a contrast between s1 and s3. But giving +stress to s1 eliminates the relevant underlying
contrast between s1 and s3, making inconsistency inevitable. Because the two words are processed together, root r1 is required to have the same underlying form for both words, denying any possibility of having the contrast come from the root. This is the key difference from processing each of the words on its own. When processing each word separately, the underlying features of the root were matched to their surface realizations separately for each word, and because the root alternates on the surface between the two words, this effectively created an (hypothetical) underlying contrast in the roots of the mappings, which was then capable of accounting for the contrast on the surface. Processing the two words together eliminates that possibility: the root must have the same underlying form in both words, forcing the surface contrast to be accounted for by an underlying contrast in the suffixes.

To set an underlying feature value via inconsistency, the learner has to be processing a word in which the underlying value of the feature matters; the underlying feature needs to be contrastive for that word in the target language. If a feature is not contrastive for a word, then any value of the feature is consistent with the word, and inconsistency will not occur as a consequence of one value as opposed to the other.

Morphemic contrast narrows the possible sources of surface contrast to a single pair of morphemes: the responsibility for differences in the surface forms of the words must lie with differences in the underlying forms of the contrasting morphemes. In this way, morphemic contrast (a form of paradigmatic information) goes beyond phonotactic word contrast. A surface contrast between two words ensures that there is a meaningful difference between the inputs of the two words. Morphemic contrast ensures that there is a meaningful difference between the underlying forms of the two contrasting morphemes. You are guaranteed that some feature is contrastive for a morpheme in a word where that morpheme stands in morphemic contrast. Setting an underlying feature for a morpheme requires processing a word in which the feature is contrastive, meaning that the morpheme stands in morphemic contrast in that word with another (either actual or possible) word.

A contrast pair can enable a feature to be set that cannot be set by individual words when the contrast pair properly marshals the contributions of alternation and contrast. An alternating feature (morphemic alternation) within the pair forces a disjunction of possible disparities, each possibly requiring additional ranking commitments. The alternating feature(s), in order to alternate, must belong to an environment morpheme, appearing in both words of the pair. The contrast between the surface values of the two words, in particular the alternation in the alternating feature(s), must be caused by differences in the
underlying forms of the contrasting morphemes (morphemic contrast). At least one feature in one of the contrasting morphemes must be (part of) the cause for the contrast inherent in the alternation, and that feature will be settable if changing its underlying value to mismatch its surface realization removes the grammar’s ability to correctly realize both words.

7.6.5 Multiple words and relative similarity

The contrast pair consisting of the words r1s1 and r1s3 is shown in (7.21). The lexicon prior to the processing of the contrast pair is shown in (7.22): two of the underlying features for the relevant morphemes have already been set (the length features for r1 and s3), and four of the six underlying features are still unset (denoted with a question mark, “?”).

(7.21) Contrast Pair: r1s1 [pāka] r1s3 [paká]

(7.22) Current Lexicon: r1 /?,–/ s1 /?,?/ s3 /?,–/

Relative similarity must now be treated with more sophistication, because we are dealing with more than one output simultaneously. The relative similarity lattice for r1s1 by itself is shown in Figure 7.5. Because one feature has been set underlyingly (the length feature of r1), there are eight candidates that are still viable; candidates assigning +long to r1 are non-viable, indicated with shaded diamonds. The relative similarity lattice for r1s3 by itself is shown in Figure 7.6, again with shaded diamonds indicating non-viability. Because two features of r1s3 have been set underlyingly (the length features of both r1 and s3), there are only four candidates that are still viable.

To simplify the visual presentation, the non-viable candidates can be dropped, leaving two smaller similarity lattices consisting only of viable candidates (corresponding to the two viable sublattices). The lattice for r1s1 is shown in Figure 7.7, and the one for r1s3 is shown in Figure 7.8. In the rest of this section, references to relative similarity lattices/orders refer to relations containing only viable candidates.

To take advantage of output drivenness while processing this contrast pair, it is desirable to construct a combined relative similarity order for the contrast pair. The composite order, formed from the two individual relative similarity lattices for r1s1 and r1s3, is called the joint relative similarity order for r1s1 and r1s3. Because of the alternating feature, the stress feature of r1, the two lattices cannot be combined into a single lattice: they conflict on what the surface value is for r1. Because there is not a single surface value for stress of r1 that works for both words, the learner needs to actively consider all
7.6 Contrast pairs

Figure 7.5. The relative similarity lattice for r1s1. The shaded diamond nodes have r1 underlyingly +long, counter to what has already been set in the lexicon, and so are not viable candidates.

Figure 7.6. The relative similarity lattice for r1s3. The shaded diamond nodes have r1 underlyingly +long and/or s3 underlyingly +long, counter to what has already been set in the lexicon, and so are not viable candidates.
possible values of the feature. This can be done by factoring the joint order into two suborders, one in which r1 is set to –stress, and one in which r1 is set to +stress. The two suborders together constitute the joint relative similarity order for the contrast pair. Each suborder is itself a lattice; the combination is not.

The two suborders are necessarily disconnected from each other; no member of one suborder is ordered (above or below) with respect to a member of the other suborder. Every element of the suborder with r1 underlyingly –stress has a disparity in stress for r1 in r1s1, but no disparity in stress for r1 in r1s3. Every element of the suborder with r1 underlyingly +stress has a disparity in stress for r1 in r1s3, but no disparity for r1 in r1s1. Each element in the r1 –stress suborder has a disparity that every element in the r1 +stress suborder lacks, and
7.6 Contrast pairs

Figure 7.9. The joint relative similarity order for the contrast pair r1s1 and r1s3. Each node is labeled with underlying forms for r1s1 and r1s3, in that order. The left suborder contains all local lexica in which r1 is –stress underlyingly, while the right suborder contains all local lexica in which r1 is +stress underlyingly.

vice-versa. In general, this “splitting” of the joint relative similarity order will occur for every feature that alternates between the two words of the contrast pair.

The joint relative similarity order for the contrast pair r1s1 and r1s3 is shown in Figure 7.9. Each node is labeled with the underlying forms for the two words of the contrast pair, with r1s1 above r1s3. As before, each node is a local lexicon. Here, each local lexicon contains underlying forms for three morphemes: r1, s1, and s3. Each local lexicon determines a pair of candidates, one each for r1s1 and r1s3, using the underlying forms of the local lexicon and the already-determined outputs. Because morphemes s1 and s3 do not alternate on the surface within this contrast pair, their unset features have their (single) surface values in the top nodes of both suborders, and both suborders treat changes in the values of any of those features (stress and length for s1, stress for s3) as disparities.

Because this order is not a lattice, there is no single set of underlying forms (a node) that is guaranteed to map to the given outputs (there is no single top node
ordered above all nodes). However, the learner is guaranteed that the top node of one of the connected suborders must contain a set of underlying forms that maps to the given outputs. The top nodes for the two suborders, (/paka/, /paká/) and (/pákai/, /pákái/), are the two candidates in which the non-alternating (in this contrast pair) features all match their single surface values. They differ in the underlying value of the alternating feature, stress of r1.

Attempting to set the value of an underlying feature now involves testing with respect to all values of the alternating feature, not just one. Testing a feature, like the stress feature for s1, follows a logic similar to the testing of unset features with single words. The learner already knows that the value matching s1’s surface realization in r1s1, –stress, will be consistent; it is not an alternating feature in this contrast pair. Setting the stress feature for s1 here requires showing that the other value, +stress, is inconsistent. The learner needs to test the +stress value for s1 with all values for the alternating stress feature of r1. This means testing two nodes, one for each suborder: the node with inputs (/paká/, /paká/) for r1 with –stress, and the node with inputs (/pákái/, /pákái/) for r1 with +stress (note that both nodes have s1 set to +stress, in the first word).

The joint relative similarity order for the contrast pair is shown again in Figure 7.10, with the local lexica having s1 set underlyingly to +stress shaded. The two local lexica to be tested are the tops of the two shaded suborders. The implicational logic of output-driven maps applies here as before: if any local lexicon with s1 assigned +stress underlyingly is grammatical, then the local lexicon at the top of its shaded suborder must also be grammatical. If any local lexicon with s1 assigned +stress is consistent, then one of the two top shaded local lexica must be consistent. Contrapositively, if both of the two top shaded local lexica are inconsistent, then all local lexica with s1 assigned +stress are inconsistent. In this case, both of the two top shaded local lexica prove to be inconsistent with the available ranking information; neither value of the stress feature for r1 can bail out assigning +stress to s1. Thus, the learner can set s1 to –stress. The case in the left suborder, with r1 assigned –stress and s1 assigned +stress, was examined in (7.17). The case in the right suborder, with r1 assigned +stress and s1 assigned +stress, was examined in (7.16).

Within each suborder, a feature can be tested by evaluating only one node, the one with a disparity for the tested feature, and as few other disparities as possible across the words of the node. Each suborder has a disparity in the alternating feature running throughout its nodes; the suborders differ on which word in the pair contains the disparity. However, the learner needs to conduct a
Figure 7.10. Testing the stress feature for $s1$. The shaded nodes are the local lexica with $+stress$ assigned to $s1$ underlyingly.

separate evaluation for each suborder, because it needs to separately test under each value of the alternating feature.

The computational benefit of output-driven maps in the learning of underlying feature values is realized immediately. For the contrast pair just described, r1s1 and r1s3, even with two features already set, the number of local lexica is $2^4 = 16$ (four unset features, each binary): they are the sixteen nodes in the joint similarity order of Figure 7.9. CPR would evaluate all sixteen. Using output drivenness, the number of local lexica that need to be evaluated to determine whether to set a single feature is two. In the contrasting morphemes ($s1$ and $s3$), there are three combined features which are unset: testing all of them would require a total of six local lexicon evaluations, less than half of the total number of local lexica.

Potential for exponential computational complexity lurks in the processing of contrast pairs, however, behind the possibility of multiple alternating unset features. All values of each alternating feature must be considered independently, and there will be exponential growth in the number of combinations of values of such features. Fortunately, the potential for such growth is limited: it is only exponential growth in the number of unset features that alternate
within the pair of words under simultaneous consideration. Each combination of values for alternating, unset features constitutes a separate case in an overall disjunction, in the terms used in Section 7.6.2. Each case would constitute a separate suborder, like the two in Figure 7.10. Testing a (non-alternating unset) feature requires evaluating one local lexicon in each suborder. In Figure 7.10, the stress feature for s1 is tested by evaluating one local lexicon in each suborder, the highest shaded one. In general, testing a feature will involve testing one local lexicon in the second level from the top of each suborder; testing all unset features in the contrasting morphemes involves, in total, evaluating all of the local lexica in the second level from the top of each suborder.

7.6.6 Another illustration: setting the stress feature for r1

It was shown in Section 7.6.1 that the stress feature for r1 could not be set on the basis of a single word, examining in particular the words r1s1 and r1s3. Section 7.6.4 further argued that a contrast pair consisting of r1s1 and r1s3 could be used to set a feature in one of the contrasting morphemes (like the stress feature of s3), but not an alternating feature in r1. It is not the case, however, that the stress feature of r1 is unusually difficult to set. A pair of words is required in which the stress feature of r1 surfaces with the correct underlying value in one of the words, and a different feature alternates between the two words in a way that provides crucial ranking restrictions. Where, in a contrast pair, a feature is positioned is crucial to its role in that pair.

Consider the pair of words r1s3, paká, and r3s3, paka. The surface contrast between the words must be a consequence of at least one difference between the underlying forms for roots r1 and r3. Note that the stress feature of r1 does not alternate in this pair of words; it can’t, because r1 only appears in one of the words. The stress feature of s3, however, does alternate between the pair of words. The stress feature of s3 is going to play the same role for this contrast pair as the stress feature of r1 played for the contrast pair (r1s1, r1s3) in Section 7.6.3: each underlying stress value of s3 will be unfaithful to one of the words, r1s3 or r3s3, and will force additional ranking restrictions, allowing inconsistency to be detected.

We already know that the value for r1’s stress feature that matches its surface realization in r1s3, –stress, will be consistent. The matter to be determined is whether assigning +stress to r1 will be consistent. Because the stress feature of s3 alternates, we need to test the underlying value of +stress for r1 twice, once with s3 underlyingly –stress and once with s3 underlyingly +stress. The two pairs of mappings to be evaluated are shown in (7.23).
7.6 Contrast pairs

(7.23) Inconsistency detected for contrast pair r1s3 and r3s3, using r1 with underlying value +stress

<table>
<thead>
<tr>
<th></th>
<th>Mappings</th>
<th>Consistent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>−stress</td>
<td>r1s3: /pá+ka/ → paká</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>r3s3: /pá+ka/ → paka</td>
<td></td>
</tr>
<tr>
<td>+stress</td>
<td>r1s3: /pá+ká/ → paká</td>
<td>No</td>
</tr>
<tr>
<td></td>
<td>r3s3: /pá+ká/ → paka</td>
<td></td>
</tr>
</tbody>
</table>

Why does this contrast pair succeed for setting the stress feature of r1, when the pair with alternating r1 stress failed? The contrast pair with alternating stress on r1, (r1s1, r1s3), has one disparity for the pair of words, whichever underlying value for r1 stress is used; the stress feature of r1 provides the sole disparity, and in only one of the two words. The contrast pair with contrasting stress on r1, (r1s3, r3s3), has a disparity for the alternating feature, the stress feature of s3, and adds a second disparity when testing the stress feature of r1 by assigning it the value opposite its (single) surface realization in the pair. No matter which underlying value is assigned to the stress feature of s3, there are two disparities at once. Any single disparity resides in only one word, and its possible inconsistency implications can be determined by processing that word in isolation. The contrast pair with r1 as one of the contrasting morphemes provides, via alternation, an additional disparity, on top of the one introduced by testing the stress feature of r1. It is the forcing of the disjunctive possibilities, each with two disparities, that gives the contrast pair (r1s3, r3s3) the power to set the stress feature of r1, power that the contrast pair (r1s1, r1s3) lacks. An alternating feature only provides half the picture; the ranking restrictions imposed by the alternating feature must be utilized to set some contrasting feature.

7.6.7 Setting environment morpheme features

It is possible for a contrast pair to set the underlying value of a feature even when none of the features in the environment morphemes alternate in the contrast pair. The focus here is on features that cannot be set on the basis of either word alone; if a feature can be set on the basis of a single word, adding a second word to be considered will not block the setting. What is interesting about the case where none of the features in the environment morphemes alternate is that the feature being set must itself be a feature of a (non-alternating) environment morpheme.

Consider the following (somewhat contrived) situation. The language to be learned has no long vowels: \texttt{NoLong} \gg \texttt{ID[length]}. Stress can vary
lexically, and stress is by default initial (the example is not dependent on the value of the default): ID[stress] \(\gg\) MainLeft \(\gg\) MainRight. Words consist of two morphemes, either a root with a prefix or a root with a suffix. A full set of mappings for two-syllable words with one-syllable morphemes, ignoring vowel length, would be as in (7.24).

(7.24) Full stress contrast in one-syllable morphemes (assuming rich base)

<table>
<thead>
<tr>
<th>morpheme</th>
<th>stress</th>
<th>underlying form</th>
</tr>
</thead>
<tbody>
<tr>
<td>tāpa</td>
<td>stress</td>
<td>(\text{p1} = /\text{ta-}/)</td>
</tr>
<tr>
<td>tápa</td>
<td>stress</td>
<td>(\text{p2} = /\text{tá-}/)</td>
</tr>
<tr>
<td>pāka</td>
<td>stress</td>
<td>(\text{s1} = /-\text{ka}/)</td>
</tr>
<tr>
<td>pakā</td>
<td>stress</td>
<td>(\text{s2} = /-\text{ká}/)</td>
</tr>
</tbody>
</table>

If a learner has access to all of the indicated multimorphemic words, then regular contrast pairs as previously described will allow the learner to determine all of the underlying forms: the underlying forms for \(\text{r1}\) and \(\text{r2}\) could be set by a contrast pair like \(\text{p1r1} \text{ and } \text{p1r2}\), the underlying forms for \(\text{s1}\) and \(\text{s2}\) could be set by a contrast pair like \(\text{r1s1} \text{ and } \text{r1s2}\), and so forth. But suppose that the learner only has some of the words available. Suppose that the learner only has access to two of the eight words: \(\text{p2r1} \text{ and } \text{r1s2}\). This leaves the learner with only one prefix, one root, and one suffix represented in the available data. The two words are shown in (7.25).

(7.25) Two words, with no directly contrasting morphemes

<table>
<thead>
<tr>
<th>morpheme</th>
<th>stress</th>
<th>underlying form</th>
</tr>
</thead>
<tbody>
<tr>
<td>tāpa</td>
<td>stress</td>
<td>(\text{p2} = /\text{tá-}/)</td>
</tr>
<tr>
<td>pakā</td>
<td>stress</td>
<td>(\text{s2} = /-\text{ká}/)</td>
</tr>
</tbody>
</table>

Phonotactics alone indicates that lexical stress must play a role (stress is initial in one word, and final in the other). But what are the underlying forms for the morphemes? Feature setting won’t get anywhere on either individual word: the stress pattern could either be due to preserving an underlying lexical stress, or could be due to default stress resolving a clash of identical underlying stress specifications. The two words do constitute a contrast pair: the two words differ in only one morpheme (\(\text{p2} \text{ vs. } \text{s2}\)), and the outputs of the two words are not identical. However, the environment morpheme, \(\text{r1}\), does not alternate between the two words: it surfaces as unstressed in both.
In this instance, the contrast pair can still be the basis for setting an underlying feature: the stress feature of the environment morpheme r1. Clearly, consistency will result if r1 is tested with an underlying value of –stress, as it surfaces with that value in both words. Testing r1 with an underlying value of +stress is the interesting part. The stress feature of r1 cannot be set on the basis of either word on its own. This is shown for p2r1 in (7.26), which is satisfied by ID[stress] ≫ ML ≫ MR, and for r1s2 in (7.27), which is satisfied by ID[stress] ≫ MR ≫ ML.

(7.26) p2r1 by itself with r1 assigned +stress is consistent

<table>
<thead>
<tr>
<th></th>
<th>ID[stress]</th>
<th>ML</th>
<th>MR</th>
</tr>
</thead>
<tbody>
<tr>
<td>/tápa/</td>
<td>tápa ~ tapá</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>/paká/</td>
<td>paká ~ páka</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>/tápá/</td>
<td>tápa ~ tapá</td>
<td>W</td>
<td>L</td>
</tr>
</tbody>
</table>

(7.27) r1s2 by itself with r1 assigned +stress is consistent

<table>
<thead>
<tr>
<th></th>
<th>ID[stress]</th>
<th>ML</th>
<th>MR</th>
</tr>
</thead>
<tbody>
<tr>
<td>/tápa/</td>
<td>tápa ~ tapá</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>/paká/</td>
<td>paká ~ páka</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>/páká/</td>
<td>paká ~ páka</td>
<td></td>
<td>L</td>
</tr>
</tbody>
</table>

As can be anticipated by comparing the final rows of the two tables, inconsistency results when both words are evaluated simultaneously with r1 assigned +stress, as shown in (7.28): the last two rows make contradictory requirements of the ranking. Assigning +stress to r1 in the pair results in inconsistency, so the learner is justified in setting r1 to be underlyingly –stress.

(7.28) p2r1 and r1s2 together with r1 assigned +stress are inconsistent

<table>
<thead>
<tr>
<th></th>
<th>ID[stress]</th>
<th>ML</th>
<th>MR</th>
</tr>
</thead>
<tbody>
<tr>
<td>/tápa/</td>
<td>tápa ~ tapá</td>
<td>W</td>
<td>W</td>
</tr>
<tr>
<td>/paká/</td>
<td>paká ~ páka</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>/tápá/</td>
<td>tápa ~ tapá</td>
<td>W</td>
<td>L</td>
</tr>
<tr>
<td>/páká/</td>
<td>paká ~ páka</td>
<td></td>
<td>L</td>
</tr>
</tbody>
</table>

How does this relate to the previous discussion of contrast pairs? When r1 is assigned an underlying value opposite its surface value, it introduces two
disparities simultaneously, one in each word, a consequence of the fact that
the feature does not alternate between the words. When either of the words
is examined on its own, there is only one disparity present, and in each case
additional ranking commitments can account for the disparity. But when both
words are examined together, so are the disparities, and the additional rank-
ing commitments necessary for each disparity conflict with each other. In the
contrast pairs discussed previously, an alternating feature in an environment
morpheme forced one disparity, and the feature being tested in one of the con-
trasting morphemes provided the other disparity, yielding two disparities for
the pair where only one appeared in any single word. In the present example,
the feature being tested is a non-alternating feature of an environment mor-
pheme, and it is simultaneously the source of both disparities. In both cases,
the inconsistency is the result of two disparities across the contrast pair, where
each word tested on its own only has one disparity.

As alluded to above, this example is a bit contrived: the provided data
deny the learner any opportunity to observe two morphemes of the same type
contrasting with each other, even though the target language predicts such con-
trasts for every morpheme type. In fact, the provided data are insufficient to
fully determine the language. The learner will not be able to set the under-
lying feature value for either $p_2$ or $s_2$ via inconsistency detection, because
there are two different grammars consistent with the data, each allowing a
different feature to remain unset. In a grammar with $ML \gg MR$, $p_2r_1$ will
surface with stress on the prefix regardless of the underlying form of $p_2$, while
$s_2$ must be set to $+\text{stress}$. In a grammar with $MR \gg ML$, $r_1s_2$ will sur-
face with stress on the suffix regardless of the underlying form of $s_2$, while
$p_2$ must be set to $+\text{stress}$. The only real observable contrast in the two-word
dataset is the contrast between a prefix and a suffix, which by assumption is
directly given to the learner. None of the features alternate in the two-word
dataset; there is no direct motivation to posit any underlying-surface disparities
at all.

Nevertheless, the example shows that it is possible for a contrast pair without
an alternating feature to be the basis for setting a feature, and in particular a
feature of an environment morpheme. The two go together: it is precisely being
a non-alternating feature in both words of the pair that allows the assignment
of an opposing underlying value to the feature to introduce two disparities
simultaneously. What allows it to work here is that the two disparities, while
both involving the same underlying feature, require very different things of the
ranking, because of the differences between the outputs of the two words. In
$p_2r_1$, $r_1$ is final, so the disparity forces stress to be pulled away from the final
sylable. In r1s2, r1 is initial, so the disparity forces stress to be pulled away from the initial syllable. If both affixes had been suffixes, so that r1s1 and r1s2 both surfaced with stress on the suffix, the stress feature for r1 could not be set by the pair; assigning +stress to r1 could consistently be accommodated in both words of the pair with the ranking \textit{MainRight} \gg \textit{MainLeft}. Not just any pair of disparities will do.

7.7 Beyond error-driven learning

The traditional conception of error-driven learning assumes that, at any given time, the learner possesses a hypothesized grammar. The learner evaluates an observed word by determining if their hypothesized grammar generates that word. If the word is generated by the learner’s grammar, it is assumed that there is nothing to be learned from the word at this time, and the learner takes no further action on the word. If the word is not generated by the learner’s grammar, then the learner attempts to modify their grammar so that it does generate the word.

This traditional conception assumes that the learner relates an observed word to a particular grammar hypothesis in isolation, independently of any other grammar hypotheses that might be available. Further, the relationship between a word and a grammar has an “all or nothing” quality. The grammar either generates the word or it doesn’t. When attempting to modify its grammar to accommodate a word, the only signal the learner gets is if an alternative grammar generates the word or not. There is no sense of comparison in which one grammar is “closer” to generating the word than another. Similarly, there is no sense of comparison in which two grammars generate the word, but one grammar does a “better” job than the other. Finally, it assumes that a grammar hypothesis is fully determined. Since the grammar is all that the learner retains in memory in traditional error-driven learning, there is no way for the learner to tell which parts of the grammar were useful in generating previously observed words, and which parts are there simply out of the need to have a fully determined grammar.

In OT, a fully determined grammatical hypothesis would be a constraint hierarchy (with all conflicts resolved), and a lexicon of underlying forms with all features set. Given such a hypothesis grammar, detecting a learning error consists of determining if the constraint ranking of the hypothesized grammar maps the input (determined by the underlying forms of the lexicon) to the observed output of the word, which can be done using production-directed
On this view, if the hypothesized ranking does map the input to the correct output, then there is nothing for the learner to learn from the word at the present time, because learning only takes place in response to an error. If the hypothesized ranking does not map the input to the correct output, then the learner attempts to fix the grammar so that it does give the right output for the input. Among the choices the learner faces in this scenario is how to fix the grammar. Should the ranking be modified? Should the lexicon be modified? When there are multiple things about the grammar that might be altered to generate the observed word, and no way to tell which aspects of the learner’s grammar deserve to be preserved, it is far from obvious how to proceed.

There are at least two issues bundled up in error-driven learning that can in principle be addressed distinctly. One is the nature of the information that a learner actually stores and retains over time. The other is the way in which a learner processes a particular observed form. The answers that error-driven learning provides for these issues fit together in a natural way, and can apply quite generally. But that very generality is limiting, including in the ways just described.

The view of learning developed in this book departs significantly from traditional error-driven learning. The learner stores partial information about the correct grammar, rather than storing only a single complete grammar representing the learner’s current “best guess.” For ranking information, the learner maintains a support of winner–loser pairs instead of just a single hypothesized ranking. For lexical information, the learner sets features in the lexicon independently rather than all at once, so that the stored underlying form for a morpheme may have some features set and others not set, instead of just a single, fully set hypothesized underlying form. The learner also stores the surface forms it has actually seen. While it is commonplace to cite the lack of any data storage at all as a virtue of error-driven learning, the discussion of contrast pairs in Section 7.6 provides a strong counter-argument: the use of contrast pairs appears to be quite important, and it requires simultaneous access to more than one surface form.

This shift toward the storage of partial information about the target grammar leads to a shift in how a learner processes learning data. The shift is away from the question of whether a given word is generated by the learner’s hypothesized grammar, and towards the question of whether a given word can provide any

---

8 Or at least determining if the ranking maps the input to an output with an overt portion that matches the observed overt form, in cases where what is observed is ambiguous with respect to the full output structure.
new information with respect to the space of grammars the learner has under consideration. Can a word provide any new ranking information? Can it provide any new lexical information? Because of the strong interaction between ranking and lexicon, any attempt to obtain new information must reckon with the learner’s uncertainty about both the ranking and the lexicon.

The work in this book is not the first to propose an alternative to error-driven learning, and alternatives are not limited to work within Optimality Theory. One example is cue learning (Dresher 1999, Dresher and Kaye 1990), which is cast within the principles and parameters framework. Cue learning processes surface forms one at a time, but processes a form by matching against a list of cues, with no regard for the learner’s ability to generate the surface form. See Tesar 2004 for further discussion of cue learning and its relation to other learning approaches.

### 7.7.1 Uncertainty in the lexicon

MRCD can be used for learning ranking information when the learner is confident about the veracity of a mapping, embodied in a winner. In phonotactic learning, the input of a winner is identical to the output, and the learner knows that the input must map to the output, because the identity input is the top element of the word’s relative similarity lattice. The learner does not know if the input it has constructed accurately reflects the underlying forms of the morphemes in the target language, but the learner does know that the mapping is correct: the input of that winner must map to the output of that winner in the target grammar. In the learning of non-phonotactic ranking information, the learner uses, as an input, the top of the viable sublattice for the word. In both situations, output drivenness ensures that the chosen input must map to the output. The learner can evaluate a winner for potential ranking information using production-directed parsing, as described in Section 5.4 and Section 7.5.

By selecting the input at the top of the viable sublattice for a word, the learner is able to efficiently contend with uncertainty in the lexicon when pursuing ranking information. When it comes to learning lexical information, i.e., setting underlying features, the situation is different. Setting features involves testing inputs other than the top of the viable sublattice for inconsistency. Testing the input at the top of the sublattice works for obtaining ranking information precisely because it avoids the issue of which features need to

---

9 More generally, when different IO correspondence relations are possible between the same input and output, the learner also needs to be confident in the particular IO correspondence adopted in their winner candidate.
underlyingly match their surface realizations, precisely the matter that feature setting addresses. A simple approach to evaluating a word’s potential for lexical information would be for the learner to test each unset feature of each word every time it is considered. That means one round of inconsistency detection for each unset feature of the word. The ability to efficiently check each unset feature separately makes such an approach possible, but it would be helpful to be able to determine even more quickly if there was a good chance that some feature in the word could be set. This would tell the learner whether it is worth the computational effort to test each unset feature.

The payoff for such a technique would perhaps be particularly noticeable for words which had a number of feature instances that were not contrastive (in any environment), and so would never be set. If the learner determined that none of the unset features needed setting, it could move on without bothering to perform inconsistency detection on each of the unset features. If, on the other hand, the learner detected the need to set at least one of the features, then the learner could decide to attempt feature setting for that word.

The key to such a technique lies, once again, with the relative similarity relation. In the top node of the viable sublattice for a word output, all unset features match their realizations in the output form. Now consider the bottom node of the lattice. In the bottom node, all unset features have the opposite value from their realizations in the output form; the input is as dissimilar from the output as the learner’s lexicon will allow. Because it is the bottom node, success for the bottom node entails success for every node in the lattice, which is to say that if the input for the bottom local lexicon maps to the output, then the input for every (viable) local lexicon maps to the output. If a viable ranking maps the bottom node input to the observed output, then no unset feature can be set on the basis of inconsistency with respect to this word, because every local lexicon is consistent with at least that viable ranking.

Recall the relative similarity lattice for r1s1 given in Figure 7.5 and repeated below. The relative similarity lattice was constructed in a context in which the learner had set only one of the underlying features for the word: r1 is set to –long. The learner’s support at that point was given in (7.7) and is also repeated below. The bottom node of the viable sublattice for r1s1 is /pakáː/. It differs from the output form of r1s1, [páka], in three of the four features (all but the length feature for r1, which is already set). To evaluate, at this point in learning, the potential of r1s1 for providing lexical information, the learner applies inconsistency detection to the candidate with input /pakáː/ (the bottom of the viable sublattice) and output [páka] (the observed surface form). This is done using MRCD, as described in Section 6.5.
7.7 Beyond error-driven learning


L20 ranking information, based on phonotactics and phonotactically contrastive features

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
<th>Ident[stress]</th>
<th>Ident[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonotactic1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phonotactic2</td>
<td></td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td>W</td>
</tr>
<tr>
<td>NonPhon1</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td>W</td>
<td>L</td>
</tr>
</tbody>
</table>

The hierarchy generated by RCD for the support in (7.7) is given in (7.29).

(7.29) \{WSP, Ident[stress]\} \supseteq \{MainLeft, MainRight, Ident[length]\} \supseteq \{NoLong\}

Given the ranking in (7.29), the optimal output for /paká:/ is [paká:]. In particular, [paká:] is more harmonic than the correct output [páka], as shown in (7.30). The winner, with input /paká:/ and output [páka], is not optimal.

10 Because the learner is performing inconsistency detection, it can use any constraint hierarchy consistent with its support for this purpose. If a candidate is optimal for even one ranking consistent with the support, then that candidate is consistent with the support.
Exploiting output drivenness in learning

(7.30) For input /paká:/ and the ranking in (373), output [paká:] is more harmonic than output [páka]

<table>
<thead>
<tr>
<th>/paká:/</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>[páka]</td>
<td></td>
<td>* ! *</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>[paká:]</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
</tbody>
</table>

At this point, MRCD constructs a new winner–loser pair, with /paká:/[páka] as the winner and /paká:/[paká:] as the loser. It does not permanently add it to the support; it keeps it temporarily for purposes of inconsistency detection. The ERCs currently in use for inconsistency detection are shown in (7.31), with the temporary inconsistency detection ERC labeled InDet1.

(7.31) ERCs after first round in inconsistency detection; the ERC temporarily created for inconsistency detection is labeled InDet1

<table>
<thead>
<tr>
<th></th>
<th>WSP</th>
<th>ML</th>
<th>MR</th>
<th>NoLong</th>
<th>IDENT[stress]</th>
<th>IDENT[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phonotactic1</td>
<td></td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phonotactic2</td>
<td></td>
<td>L</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NonPhon1</td>
<td>W</td>
<td></td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>InDet1</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
</tr>
</tbody>
</table>

When RCD is applied to the list in (7.31), inconsistency is detected. Although WSP can be placed in the top stratum, once the WSP constraint and the NonPhon1 ERC are removed, every remaining constraint still prefers a loser in one of the remaining rows. This tells the learner that the supposed winner it was pursuing, /paká:/[páka], is not optimal under any viable ranking; it is not a possible optimum, given the support. Obtaining further ranking information would only further restrict the space of viable rankings. Thus, the learner knows that at least one of the unset features for r1s1 needs to be set. The learner does not at this point know which of the features need to be set, or even how many. The learner is not guaranteed that it will be able to set any of the features based on r1s1 alone at this point. But the knowledge that at least one of the unset features needs to be set may be justification enough for the learner to attempt feature setting for each unset feature of r1s1.

To summarize, the learner can test a word for new ranking information by applying MRCD to the input at the top of the word’s viable sublattice. The
7.7 Beyond error-driven learning

The learner can determine if a word might provide new lexical information by applying inconsistency detection to the input at the bottom of the word’s viable sublattice.

The differences between evaluation for ranking information and evaluation for lexical information follow from the structure of the relative similarity lattice. Recall that the optimality of any given node entails the optimality of any node above it in a relative similarity lattice, but doesn’t entail anything about the nodes below it. The non-optimality of a node doesn’t entail anything about the nodes above it, but it does entail the non-optimality of the nodes below it. This directly reflects the inherent asymmetry of logical entailment: if we are given that p entails q, then modus ponens allows us to conclude q based upon p, and modus tollens allows us to conclude not-p based upon not-q.

If the learner evaluates the top node of the viable sublattice for a word, and finds that it is not optimal with respect to the (viable) evaluation ranking, then the learner knows it can obtain ranking information to rule out that ranking (and possibly others). It knows this because if the correct output is not optimal for the top node of the viable sublattice, then it isn’t optimal for any node of the viable sublattice. The evaluation ranking must be to blame, because no viable input can produce the output. When the top node isn’t optimal for a ranking, it can act as a proxy for the entire viable sublattice with respect to that ranking, allowing the learner to manage the lexical uncertainty.

When learning lexical information, the learner is attempting to demonstrate the inconsistency of an input. To demonstrate inconsistency, it is not sufficient to show that there is a viable ranking for which the candidate with that input is not optimal. Inconsistency detection implicitly tests against all viable rankings: inconsistency means that the candidate is not optimal for any viable ranking. If a node succeeds on some viable ranking, then every node above it succeeds on that same ranking. Thus, if a node is consistent with the learner’s current knowledge, then every node above it is consistent with the learner’s current knowledge. Consistency is upward-entailing, because optimality is upward-entailing. Inconsistency/suboptimality is downward-entailing.

If the learner evaluates the bottom node of the viable sublattice for a word, and finds that it is consistent with respect to some viable ranking, then the learner knows that it cannot set any features based on the word (given the current state of the learner’s knowledge). It knows this because if the correct output is optimal for the bottom node for some viable ranking, then it is optimal for every node of the viable sublattice for that same viable ranking. When the
bottom node is consistent with the learner’s knowledge, it can act as a proxy for the entire viable sublattice with respect to the learner’s knowledge. Again, the learner can manage the lexical uncertainty, in this case via the bottom node.

7.7.2 Uncertainty about the ranking

In storing a support, the learner implicitly maintains a representation of a space of rankings, those that are viable (consistent with the support). Instead of having a single hypothesized constraint hierarchy, the learner has a collection of them. Constructing a constraint hierarchy from a support permits the learner to treat the constructed hierarchy as if it were “the” hypothesized ranking for purposes of evaluating words. Thus, traditional error-driven learning can be imitated in conjunction with a retained support, and this is exactly what MRCD does.

One could imagine a learner that evaluated a word with respect to all of the viable rankings. In a sense, this is what inconsistency detection does when used to evaluate a word with the bottom node of the viable sublattice for that word. It can be done efficiently because the condition being evaluated is inconsistency: failure to be optimal for all viable rankings. If the relevant candidate is not optimal for one viable ranking, then an informative loser is produced, pointing to a different viable ranking worth trying (the one constructed when the new winner–loser pair is combined with the learner’s support). The learner can efficiently work through a few viable rankings (out of a possibly vastly larger set) until either inconsistency is reached or a viable ranking making the candidate optimal is found. When evaluating a word for the learning of lexical information, uncertainty about the ranking can be efficiently dealt with.

The situation is different when the learner is evaluating a word for the learning of ranking information. There the condition being evaluated is the existence of at least one ranking for which the relevant candidate is not optimal. If the learner tries one viable ranking, and the candidate is optimal, there might still be a different viable ranking for which the candidate is not optimal. But no “informative loser” is produced when the candidate is found to be optimal. No clear indication is given of whether another viable ranking might be worth trying, let alone which one it would be. As discussed in Section 5.4.5, rigorously determining that a candidate is optimal with respect to all viable rankings appears to be quite computationally intensive. In response to these practical concerns, the learner constructs a single hierarchy when evaluating a word for ranking information.
7.7 Beyond error-driven learning

Even if one accepts, for the time being, the intractability of fully evaluating the space of viable rankings, there remains the matter of which hierarchy to construct from the support for use in evaluation. To obtain an informative loser, the winner must fail to be the sole optimum for the evaluation hierarchy. While it might seem counterintuitive, it is to the advantage of the learner to construct an evaluation hierarchy that is least likely to make the winner optimal. The learner is much better off if it accumulates winner–loser pairs that commit the learner to necessary ranking relations, rather than rely on ranking biases to repeatedly “guess right” across changing circumstances. This is especially true for a learner making use of inconsistency detection: inconsistency is checked relative to committed ranking information, not default biases. The more ranking information accumulated, the more potential for detecting inconsistency (the more there is to be inconsistent with).

It was shown in Chapter 5 that the “all constraints as high as possible” bias for original RCD was poor at enforcing restrictiveness. Because of restrictiveness concerns, the “best-guess” constraint hierarchy for the learner at any given time is the one constructed using the “faithfulness low” bias, the one constructed by BCD. The “faithfulness low” constraint hierarchy was used for error detection during the learning of phonotactic ranking information. This choice was validated by the analysis of phonotactic learning given in Section 5.7. Much of what phonotactic ranking information does is indicate where faithfulness must dominate markedness in order to preserve observed phonotactic contrasts. Obtaining such information via MRCD requires that the learner construct a hierarchy in which the relevant faithfulness constraints do not already dominate the relevant markedness constraints. Ranking faithfulness constraints as low as possible is a good general-purpose technique for avoiding faithfulness-over-markedness relations that the support has not already committed to. Apart from the learner’s estimate of the target ranking, the “faithfulness low” bias produces a hierarchy expected to be more likely to yield an error, and thus advance phonotactic learning.

The story is different for the learning of non-phonotactic ranking information. The pursuit of non-phonotactic ranking information described in Section 7.5 occurs when the learner sets a feature that alternates. The learner then evaluates words in which the newly set feature surfaces unfaithfully; it is the disparity involving that newly set feature that creates the potential for non-phonotactic ranking information. Accounting for the disparity will likely require some markedness-over-faithfulness ranking relations, with the disparity being forced in order to satisfy markedness at the expense of faithfulness.
Obtaining such information via MRCD requires that the learner construct a hierarchy in which the relevant markedness constraints do not already dominate the relevant faithfulness constraints. The likelihood of this can be increased by using a “markedness low” ranking bias: rank every markedness constraint as low as possible. This is like BCD, but with the roles of markedness constraints and faithfulness constraints reversed.

The following example illustrates the advantage of “markedness low” over “faithfulness low” for the learning of non-phonotactic ranking information. The learner’s support is shown in (7.32). The hierarchy that results from using the “faithfulness low” bias is shown in (7.33), while the hierarchy that results from using the “markedness low” bias is shown in (7.34).

(7.32) The support before obtaining non-phonotactic ranking information

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>páka</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td>L</td>
<td></td>
</tr>
</tbody>
</table>

(7.33) \{WSP\} \gg \{IDENT[stress]\} \gg \{MAINLEFT, MAINRIGHT\} \gg \{IDENT[length]\} \gg \{NoLong\}

(7.34) \{IDENT[stress], IDENT[length]\} \gg \{WSP, MAINLEFT, MAINRIGHT, NoLong\}

The learner has just set suffix s4 to +long (on the basis of r1s4 [paká:]). The learner then checks to see if it has any stored words in which s4 surfaces as –long, and comes up with r3s4 [páka]. As described in Section 7.5, the learner assigns all unset feature values to match their surface values, resulting in the input /páka:/ for r3s4, with only a single disparity, the suffix length. Production-directed parsing is performed on this input to see if an informative loser can be produced.

If production-directed parsing uses the “faithfulness low” hierarchy in (7.33), the optimal candidate for /páka:/ will have output [páka], as shown in (7.35). This matches the winner, so no informative loser is produced, and no non-phonotactic ranking information is obtained.

11 This example comes from the extended illustration of the learning of language L20, specifically in Section 7.9.2.2.
(7.35) Candidate /páka:/ is optimal with the “faithfulness low” hierarchy

<table>
<thead>
<tr>
<th>/páka:/</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>[páka]</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[paká]</td>
<td></td>
<td>*!</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[páka:]</td>
<td>*!</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[paká:]</td>
<td>*!</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If production-directed parsing uses the “markedness low” hierarchy in (7.34), the optimal candidate for /páka:/ will have output [páka:], as shown in (7.36). This does not match the winner, and is therefore an informative loser, providing new non-phonotactic ranking information.

(7.36) Candidate /páka:/ is optimal with the “markedness low” hierarchy

<table>
<thead>
<tr>
<th>/páka:/</th>
<th>IDENT[length]</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>WSP</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>[páka]</td>
<td>*!</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[paká]</td>
<td>*!</td>
<td>**</td>
<td>*</td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>[páka:]</td>
<td>*!</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[paká:]</td>
<td>*!</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
</tbody>
</table>

This informative loser is then used to form a new winner–loser pair to be added to the learner’s support. The new winner–loser pair is shown in (7.37).

(7.37) New winner–loser pair containing non-phonotactic ranking information

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>/páka:/</td>
<td>páka</td>
<td>páka:</td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

This winner–loser pair contains information about the necessary domination of a faithfulness constraint, IDENT[length]. That is precisely the faithfulness constraint that is violated by the disparity resulting from the newly set feature (the length feature of the suffix s4). The “faithfulness low” hierarchy in (7.33) already had WSP dominating IDENT[length], not due to explicit commitment but due to ranking bias. The “markedness low” hierarchy in (7.34) has IDENT[length] dominating WSP, due to the ranking bias favoring faithfulness over markedness, and as a result produces the informative loser, resulting in new non-phonotactic ranking information.
Because the learner maintains a support, it can construct different constraint hierarchies from the same support for different purposes. For finding informative losers, the learner is best served if it constructs a hierarchy that is more likely to result in a learning error. When learning phonotactic ranking information, the “faithfulness low” bias is the most useful. When learning non-phonotactic ranking information, the “markedness low” bias is the most useful. In pursuing ranking information, the learner should not necessarily use the constraint hierarchy that is most likely to reflect the target constraint ranking; it should use the constraint hierarchy most likely to produce an informative loser.

7.7.3 Single form learning
If the pursuit of informative losers for a winner were exhaustive, then the learner could only obtain further ranking information when an alternating feature had been set, or when a new word was encountered. However, as just discussed in Section 7.7.2, this learner does not exhaustively pursue informative losers, so when a previously processed word is processed again, it is possible that more ranking information is now accessible. This is because the learner’s support might have changed, and a different stratified hierarchy might be generated, one that does not make the observed word optimal. Therefore, when processing a single word (whether new or repeated), the learner wants to evaluate that word’s potential for providing both new lexical information and new ranking information.

The learner can search for new ranking information by constructing a candidate with the top of the viable sublattice as input, and applying MRCD. The learner uses the constraint hierarchy constructed with the faithfulness low bias for evaluation. The faithfulness low bias is intended to reduce the likelihood that the candidate will be optimal as a result of faithfulness preserving the values assigned to unset features, because those values match the surface realizations. If the constructed candidate for the observed word is not the sole optimum, then the learner constructs a new winner–loser pair and adds it to the permanent support.

To pursue lexical information, the learner constructs a candidate with the bottom of the viable sublattice as input. The learner then pursues inconsistency detection on that candidate. If inconsistency is detected, then the learner proceeds to attempt to set each unset feature for the word. If inconsistency is not detected, then the learner cannot set any features based on the word at that time.

There is a single initial test that the learner can perform on a word to determine if it might be able to learn something from that word alone at that time. The evaluation ranking used is the one produced with the faithfulness low
bias, as used when testing for new ranking information. The evaluation input used is the bottom of the viable sublattice, the same input used when testing for new lexical information. If the resulting candidate is the sole optimum for that evaluation input with respect to the evaluation ranking, then the learner cannot learn anything from that word alone at that time. Since the ranking maps the input at the bottom of the viable sublattice to the correct output, by output drivenness it will map the input at the top of the viable sublattice to the same (correct) output, so there is no new ranking information to be gained. Since there exists a viable ranking (namely, the evaluation ranking) that maps the bottom of the viable sublattice to the correct output, by output drivenness every input in the sublattice maps to the correct output for that same viable ranking, so inconsistency detection will not set any underlying features based on that word alone.

This initial test, which we will label initial word evaluation, looks a lot like error detection, and it functions somewhat similarly. The learner is using a ranking hypothesis that is its “best guess,” but using a lexical hypothesis that is in some sense its “worst guess.” The goal of initial word evaluation is not to determine if the observed word is generated by the learner’s best-guess grammar, but to filter out, with modest computational effort, words that have no chance of providing new information, either ranking or lexical. If the evaluation candidate is the sole optimum for the evaluation ranking, then the learner does not further process the word at that time; the word can be said to pass initial word evaluation. If the evaluation candidate is not the sole optimum, then the word can be said to fail initial word evaluation, and the learner fully processes the word for possible ranking information and possible lexical information.

If, when pursuing lexical information, the learner succeeds in setting at least one feature, then the learner goes on to pursue non-phonotactic ranking information based on the newly set features. For a newly set feature, the learner checks its stored words for words in which that same feature (i.e., with the same morpheme) surfaces unfaithfully. If such a word is found, call it the alternating word (because it shows that the feature alternates), then the learner constructs, for the alternating word, an input in which all (still) unset features are assigned values matching their surface realization in the alternating word (the top of the viable sublattice for the alternating word). The learner then applies MRCD to this candidate, using the markedness low ranking bias, as discussed in Section 7.7.2.

The procedures just described can be grouped together into a component of the learner called single form learning. When single form learning is performed on a word, the learner first performs initial word evaluation. If the
word fails initial word evaluation, indicating that the word might provide new information, then the learner processes the word for new ranking information and then processes it for new lexical information.

### 7.7.4 Contrast pair learning

The use of contrast pairs is another significant departure from traditional error-driven learning. What is particularly interesting is that processing two morphologically related words at once often involves only a small amount of additional processing effort relative to processing a single word, but the impact on learning is quite large. The potential for significantly greater processing effort for a contrast pair lies in the possibility of a pair with a large number of alternating unset features in the environment morphemes, because the number of cases to be tried grows exponentially in the number of alternating features within the pair.

More significant than the computational effort required to process a contrast pair are the changes to the architecture of learning required to make use of contrast pairs. Most significantly, the learner must have some memory of surface forms. At most one of the words forming a contrast pair can be the word that the learner most recently heard; the other word (if not both) must be selected by the learner from memory. Once we consider a learner capable of storing some surface forms, the door is open to a learner that can reexamine a surface form even when the form hasn’t just been observed. Such a learner can intelligently select individual stored words for further processing, as well as contrast pairs of stored words.

The learner can use initial word evaluation as a filter when selecting the first word for a contrast pair. Two words that pass initial word evaluation will not yield any information when processed as a contrast pair. That is because they are both consistent with the same constraint hierarchy, namely the one generated by BCD and used for evaluation. Neither word will restrict the space of viable rankings in a way that will produce any inconsistency for the other. It is not necessary that both words in a contrast pair fail initial word evaluation, but at least one of them must, if the pair is to be informative. This fact helps the learner to focus the search for contrast pairs: start with a word that fails initial word evaluation, and then look for stored words that will form a proper contrast pair with it.

In the Output-Driven Learner, described below in Section 7.8, the learner searches for a contrast pair when it has at least one word that fails initial word evaluation, and single form learning on all stored words fails to produce any further new information. Those criteria prioritize single form learning ahead
of the use of contrast pairs. That prioritization is computationally motivated (it takes less computational effort to process a single word than a contrast pair), but is not essential to the success of learning: any feature that can be set via inconsistency detection on the basis of a single word can also be set on the basis of a contrast pair containing that word.

7.8 The Output-Driven Learner (preliminary)

The Output-Driven Learner (ODL) is the term for the algorithmic synthesis of the learning proposals in this book. A preliminary outline of the ODL is given in (7.38); it will be augmented with an additional component to address further restrictiveness issues in Chapter 8.

(7.38) Preliminary Outline of The Output-Driven Learner

1. Phonotactic learning (prior to morphological awareness).
2. Single form learning on all stored words.
   a. If new grammar information was obtained for a word, repeat single form learning on all stored words.
3. If no stored words fail initial word evaluation:
   a. Wait until a new word is observed.
   b. Perform single form learning on that word.
   c. If new grammar information was obtained, return to step 2.
      Otherwise, repeat step 3.
4. If a stored word fails initial word evaluation:
   a. Look for a word sharing all but one morpheme with the failed word, such that an unset feature in a shared morpheme alternates between the two words. Form a contrast pair with the two words.
   b. Attempt to set each unset, non-alternating feature of the contrast pair.
   c. If a feature was newly set, look for instances in other stored words where the newly set feature is unfaithfully realized, and test for new (non-phonotactic) ranking information. Then return to single form learning (step 2).
   d. If no feature was newly set, continue searching for an informative contrast pair for the word (step 4a).
   e. If no informative contrast pair is found for that word, repeat step 4 for any other stored words that fail initial word evaluation.
5. If no informative contrast pair could be found in step 4:
   a. Wait until a new word is observed.
   b. Perform single form learning on that word.
   c. If new grammar information was obtained, then return to step 2.
      Otherwise, repeat step 5.

This learner makes use of Biased Constraint Demotion, Multi-Recursive Constraint Demotion, the inconsistency-based setting of underlying features from
Exploiting output drivenness in learning

CPR, contrast pairs, and output drivenness. It stores the surface forms for words it observes and can recall stored words during later processing.

### 7.9 Learning language L20

Language L20 was described in Section 5.10. The details are repeated here for convenience.

<table>
<thead>
<tr>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>pá:ka</td>
<td>s1 = /-ka/</td>
</tr>
<tr>
<td>páka</td>
<td>pá:ka</td>
<td>pá:ka</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>pá:ka</td>
<td>s3 = /-ká/</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>pá:ka</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>

r1 /−,−/ r2 /−,+/ r3 /+−/ r4 /+,+/

s1,s2 /−,?/ s3 /+−/ s4 /+,+/

WSP ⇒ Ident[stress] ⇒ MainLeft ⇒ MainRight ⇒ Ident[length] ⇒ NoLong

#### 7.9.1 Phonotactic learning

For phonotactic learning, the learner attends only to word forms, without use of any morpheme identity information. The phonotactic word inventory for L20 is given in (7.39).

(7.39) páka paká pá:ka paká:

The learner initially has an empty support, and BCD assigns the initial hierarchy in (7.40).

(7.40) {WSP, MainLeft, MainRight, NoLong} ⇒ {Ident[stress], Ident[length]}

Suppose the first word processed by the learner is [páka]. The learner constructs the zero-disparity candidate for this output, /páka/[páka], and adopts this as a winner. Given the hierarchy in (7.40), there are two candidates which tie for optimality (using the CTie criterion) for the input /páka/: [páka] and [paká]. The winner is not the sole optimum. As shown in (7.41), the two candidates conflict on the top stratum, with MainLeft preferring [páka] (the winner), and MainRight preferring [paká], which the learner then adopts as the loser.
Applying BCD to this single winner–loser pair results in the hierarchy in (7.42).

(7.42) \( \{\text{WSP, MainLeft, NoLong}\} \gg \{\text{MainRight}\} \gg \{\text{Ident[stress]}, \text{Ident[length]}\} \)

This hierarchy makes the winner, [páka], the sole optimum. The winner then moves on to another word.

Suppose the next word is [paká]. The currently optimal output for input /paká/ is [páka], due to the domination of MainLeft. The learner adopts this currently optimal candidate as the loser, paired with the winner. The learner’s support, after the addition of this second pair, is shown in (7.43).

(7.43) The learner’s support, after the second phonotactic word

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>NoLong</th>
<th>Ident[stress]</th>
<th>ML</th>
<th>MR</th>
<th>Ident[length]</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>páka</td>
<td>W</td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Applying BCD to this list yields the hierarchy in (7.44).

(7.44) \( \{\text{WSP, NoLong}\} \gg \{\text{Ident[stress]}\} \gg \{\text{MainLeft, MainRight}\} \gg \{\text{Ident[length]}\} \)

This hierarchy makes the winner, [paká], the sole optimum, so the learner proceeds to the next word.

Suppose the next word is [paká:]. The currently optimal output for input /paká:/ is [paká]. This is distinct from the winner, so the learner constructs a third winner–loser pair and adds it to the support. The learner’s support is shown in (7.45), and the corresponding BCD-derived hierarchy is shown in (7.46).
Exploiting output drivenness in learning

(7.45) The phonotactic support

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>\text{IDENT}[stress]</th>
<th>ML</th>
<th>MR</th>
<th>\text{IDENT}[length]</th>
<th>\text{NO}_{\text{LONG}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>páka</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This support is sufficient to make all four phonotactic words sole optima. The support in (7.45) is the product of phonotactic learning.

Phonotactic learning has made progress: some of the twenty-four languages of the typology have been ruled out. But multiple languages have not. The ranking information in (7.45) is consistent with L4, L6, L7, L9, L10, L12, L13, L14, L15, L16, L20, and L23, that is, twelve of the twenty-four possible languages.

7.9.2 Initial single form learning

When the learner begins to make use of paradigmatic information, it initially processes words using single form learning.

7.9.2.1 r1s1

Suppose the next word processed by the learner is r1s1, with output [páka]. The word fails initial word evaluation. Because no underlying features have yet been set, evaluating for ranking information simply duplicates phonotactic learning for output [páka], and no new ranking information is obtained.

Evaluating r1s1 for lexical information uses the input at the bottom of the word’s viable sublattice. Since no features have been set, this is /paːká:/, the input in which every feature is assigned the value opposite its surface realization. Inconsistency detection detects inconsistency: input /paːká:/ cannot map to output [páka] consistent with the learner’s support. This is shown in (7.47), with the additional winner–loser pair generated by inconsistency detection shown below the second horizontal double-line (this pair is not added to the support). Because inconsistency was detected, the learner attempts to set the underlying features of r1s1.
(7.47) Inconsistency detected for candidate /paːká:/[páka]

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>paka</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká</td>
<td>W</td>
<td></td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>págá:</td>
<td>págá:</td>
<td>págá</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

The stress feature of r1 surfaces +stress, so the learner tests it by constructing the minimal disparity input with r1 assigned –stress underlyingly: /paka/. The candidate /paka/[páka] is consistent with the learner’s support, so the feature cannot be set at this time. The consistency is demonstrated in (7.48): when the constraints are ranking in the order displayed in the tableau, all the winner–loser pairs are satisfied, and the new candidate is optimal.

(7.48) /paka/[páka] is consistent, so r1’s stress feature cannot be set

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NoLonG</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>paka</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká</td>
<td>W</td>
<td></td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>páká</td>
<td>páká</td>
<td>páká</td>
<td>L</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

The length feature of r1 surfaces –long, so the learner tests it by constructing the minimal disparity input with r1 assigned +long underlyingly: /pá:ka/. The candidate /pá:ka/[páka] proves to be inconsistent, as shown in (7.49). The new winner–loser pair, shown below the second horizontal double-line, is fully inconsistent with the third phonotactic winner–loser pair, as the two make opposing requirements of the relative ranking of IDENT[length] and NoLong.

(7.49) /pá:ka/[páka] is inconsistent, so r1 can be set to –long

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>paka</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká</td>
<td>W</td>
<td></td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>páká</td>
<td>páká</td>
<td>páká</td>
<td>L</td>
<td></td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
</tr>
</tbody>
</table>
Exploiting output drivenness in learning

The learner then sets r1 to be underlyingly –long in the lexicon, yielding the lexicon shown in (7.50).

(7.50) Learner’s lexicon after setting r1 to –long

<table>
<thead>
<tr>
<th>r1</th>
<th>r2</th>
<th>r3</th>
<th>r4</th>
</tr>
</thead>
<tbody>
<tr>
<td>?,?</td>
<td>?,?!</td>
<td>?,?!</td>
<td>?,?!</td>
</tr>
<tr>
<td>s1</td>
<td>s2</td>
<td>s3</td>
<td>s4</td>
</tr>
<tr>
<td>?,?!</td>
<td>?,?!</td>
<td>?,?!</td>
<td>?,?!</td>
</tr>
</tbody>
</table>

Having set r1 to –long, the learner can then check to see if any non-phonotactic ranking information can be obtained, by looking for words in which the length feature of r1 surfaces unfaithfully (that is, words in which r1 surfaces as +long). However, the length feature of r1 does not alternate in L20; it is always short. So no non-phonotactic ranking information can be obtained at this point.

The learner will then test the stress and length features for s1 in the word r1s1. Neither feature can be set at this point, however: /páka:/ and /pák¡/ are both consistent with the learner’s current grammatical information. The learner has finished processing r1s1.

Note that initial word evaluation, if reapplied to r1s1, would now use the input /paká:/, with the length feature for r1 fixed at its set value, and the other three features assigned values opposite their surface realizations in r1s1. Under the ranking in (7.46), the candidate /paká:/ is not optimal, losing to /paká:/.

7.9.2.2 r1s4

Suppose the next word processed by the learner is r1s4, with output [paká:]. The learner’s lexicon has one relevant underlying feature set in the lexicon (r1 set –long), and its value matches the output realization in this word. The candidate /paká:/ provides no new ranking information.

Evaluation for lexical information uses the input at the bottom of the viable sublattice for r1s4. The constructed candidate, /páka/[paká:], proves to be inconsistent with the learner’s support. The learner then attempts to set the underlying features of r1s4.

The stress feature of r1 surfaces –stress in r1s4, so the learner tests it by constructing the minimal disparity input with r1 assigned +stress underlyingly: /pák!:/. The candidate /pák!: is consistent with the learner’s current grammatical information, so the feature cannot be set at this time. The same proves to be true of the stress feature for s4: the candidate /paka:/ is also consistent.
The length feature of s4 surfaces as +long, so the learner tests it by constructing the minimal disparity input with s4 assigned –long underlyingly: /paká/. The candidate /paká/ proves to be inconsistent, as shown in (7.51). The new winner–loser pair, shown below the second horizontal double-line, is fully inconsistent all by itself: an underlyingly short vowel cannot surface as long in this system.

\[(7.51) \quad /paká/ [paká:] \text{ is inconsistent, so } s4 \text{ can be set to } +\text{long}\]

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>pákà</td>
<td>pákà</td>
<td>pákà</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>pákà</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká</td>
<td>W</td>
<td></td>
<td></td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká:</td>
<td>paká</td>
<td></td>
<td>L</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The learner then sets s4 to be underlyingly +long in the lexicon, yielding the lexicon shown in (7.52).

\[(7.52) \quad \text{Learner’s lexicon after setting } s4 \text{ to } +\text{long}\]

\[
\begin{array}{ccccccc}
r1 & /?,?/ & r2 & /?,?/ & r3 & /?,?/ & r4 & /?,?/\\
s1 & /?,?/ & s2 & /?,?/ & s3 & /?,?/ & s4 & /?,,+/
\end{array}
\]

Having set s4 to +long, the learner can then check to see if any non-phonotactic ranking information can be obtained, by looking for words in which the length feature of s4 surfaces unfaithfully (that is, words in which s4 surfaces as –long). The word r3s4 is such a word: it surfaces as [pákà]. The learner constructs a candidate by assigning to all unset underlying features for r3s4 the values that match their surface realization in r3s4. The result is the candidate /pákà/.[pákà]. The disparity in this candidate is forced by the fact that s4 has now been set to +long (due to word r1s4). Assigning all of the other features their surface realizations in r3s4 gives the input at the top of the viable sublattice for r3s4, thus it must be grammatical. In fact, this is exactly the case discussed in Section 7.5.

The learner evaluates the candidate /pákà/.[pákà] with the constraint hierarchy constructed using the markedness low bias, as discussed in Section 7.7.2. When RCD with a markedness low bias is applied to the support in (7.45), the result is the hierarchy in (7.53).

\[(7.53) \quad \{\text{IDENT[stress], IDENT[length]}\} \gg \{\text{WSP, MAINLEFT, MAINRIGHT, NoLong}\}\]
Under this ranking, the optimal output for /páka:/ is [páka:]. The learner can now form a winner–loser pair, with /páka://[páka] as the winner and /páka://[páka:] as the loser. This winner–loser pair can be added to the learner’s support, joining the phonotactic winner–loser pairs. The new support is shown in (7.54).

(7.54) The support after obtaining non-phonotactic ranking information from r3s4

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NO_LONG</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páká</td>
<td>paká</td>
<td>W</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>páká</td>
<td>páká</td>
<td></td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
<td></td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If we apply BCD, with its faithfulness low bias, to the support in (7.54), we get the same hierarchy as obtained prior to this latest winner–loser pair, the one in (7.46). But new ranking information has nevertheless been obtained. In particular, the position of WSP now has more direct support. With just the phonotactic winner–loser pairs, WSP could appear anywhere in the hierarchy and be consistent with the pairs; its position at the top was purely a matter of ranking bias. With the addition of the fourth (non-phonotactic) winner–loser pair, WSP is constrained to dominate both IDENT[length] and NO_LONG.

This additional ranking information eliminates more languages of the typology as possibilities: L6, L10, L12, L15, L16, and L23. The remaining viable members of the typology are L4, L7, L9, L13, L14, and L20. The learner has implicitly narrowed the range of possibilities to six. The possibilities that have been eliminated are those that permit long vowels in unstressed syllables. Phonotactically, no long vowels appear in unstressed syllables in L20. That was realized only implicitly in phonotactic learning. The faithfulness-low bias of BCD allowed WSP to be undominated, as part of finding the most restrictive ranking consistent with the support, but the support itself contained no commitment to the domination of IDENT[length] by WSP; languages allowing unstressed long vowels were not explicitly ruled out. The observation of an alternation in vowel length, in which a vowel which must be long underlyingly surfaces as short in unstressed position, provides the basis for the construction of an explicit winner–loser pair requiring such domination, in order to enforce the neutralization vowel length in unstressed syllables.
7.9 Learning language L20

7.9.2.3 r2s1, r4s1

When the learner processes r2s1, with output [pá:ka], none of the features for the word have yet been set underlyingly. Evaluation for ranking information simply produces an identity candidate, /pá:ka/[pá:ka], which is optimal with respect to the hierarchy constructed by BCD (recall that, although the support has grown, BCD still produces the hierarchy in (7.46)).

Evaluation for lexical information will use the input at the bottom of the viable sublattice, yielding the candidate /paká/[pá:ka], which proves to be inconsistent. The learner then attempts to set the underlying features of r2s1.

The learner is unable to set the stress features for either r2 or s1 at this point. The length feature of r2 surfaces +long in r2s1, so the learner tests it by constructing the minimal disparity input with r2 assigned –long underlyingly: /pákα/. The candidate /pákα/[pá:ka] is inconsistent in the same way that /paká/[paká:] was inconsistent for r1s4: there is no way for an underlyingly short vowel to surface as long. This inconsistency allows the learner to set r2 to +long underlyingly.

Having set r2 to +long, the learner can then check to see if any non-phonotactic ranking information can be obtained, by looking for words in which the length feature of r2 surfaces unfaithfully (that is, words in which r2 surfaces as –long). The word r2s3 is such a word: it surfaces as [paká]. The learner constructs a candidate by assigning to all unset underlying features for r2s3 the values that match their surface realization in r2s3. The result is the candidate /pa:ká/[paká].

The learner tests /pa:ká/[paká] to see if it is currently optimal, using the hierarchy constructed with a markedness-low bias for the support in (7.54): That hierarchy is shown in (7.55).

\[
\{\text{Ident}[\text{stress}]\} \gg \{\text{WSP}\} \gg \{\text{Ident}[\text{length}]\} \gg \{\text{MainLeft, MainRight, NoLong}\}
\]

Under this ranking, the optimal output for /pa:ká/ is [paká]. Because the winner /pa:ká/[paká] is already optimal, no new winner–loser pair is formed. Not surprisingly, the length disparity for r2 in r2s3 is accounted for by the winner–loser pair added to the support earlier to account for the length disparity for s4 in r1s4.

The learner finally turns its attention to the length feature of s1 in r2s1. The relevant minimal disparity candidate, /pá:ka:[pá:ka], is consistent, so the length feature of s1 will not be set here.

An analogous sequence of events will take place when the learner processes r4s1, which surfaces as [pá:ka], the same output as for r2s1. Because r4 surfaces
as long, its underlying length feature will be set to +long. The length feature of s1, and the stress features of both morphemes, cannot be set at this point on the basis of r4s1.

The learner’s lexicon at this point is shown in (7.56): all of the morphemes with vowels that surface long in some environment have been set to +long underlyingly.

(7.56)  The learner’s lexicon after setting r2 and r4 to +long

<table>
<thead>
<tr>
<th></th>
<th>r1</th>
<th>r2</th>
<th>r3</th>
<th>r4</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>r?,-l</td>
<td>r?,-l</td>
<td>r?,-l</td>
<td>r?,-l</td>
</tr>
<tr>
<td>s2</td>
<td>r?,-l</td>
<td>r?,-l</td>
<td>r?,-l</td>
<td>r?,-l</td>
</tr>
</tbody>
</table>

7.9.2.4  r3s1, r1s3
When the learner processes r3s1 (or any word with r3), which surfaces as [páka], it will be able to set r3 to –long underlyingly. This is directly analogous to the way that r1 was set to –long above; the inconsistency will be the same as shown in (7.49). The relevant vowel surfaces as –long in a stressed environment, and the support mandates that a vowel’s length feature surfaces faithfully in a stressed environment. The same process will allow the learner to set s3 to underlyingly –long when it processes a word in which s3 surfaces as stressed, such as r1s3 (or r2s3).

The same will not extend to the length features of s1 and s2. The reason is rather simple: s1 and s2 are never stressed on the surface. If you look back over the extent of this illustration, you can observe that every set length feature was set on the basis of a word in which the relevant vowel was stressed. In this language, length is contrastive in stressed position, a property established for the learner during phonotactic learning. Length is neutralized in unstressed position, a property the learner accepted implicitly during phonotactic learning and committed to explicitly after determining that an underlyingly long vowel shortened in unstressed position. The vowels of morphemes s1 and s2 never appear in stressed position, so the learner is unable to set their length feature underlyingly.

The learner’s lexicon at this point is shown in (7.57). No further underlying features can be set on the basis of single forms, given the learner’s current support. None of the stress features can be set, for the reasons discussed in Section 7.6.1. The length features for s1 and s2 cannot be set, as just discussed.
The learner’s lexicon after the initial round of single form learning

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>?,-/</td>
<td>r2</td>
<td>?,+/</td>
<td>r3</td>
<td>?,-/</td>
<td>r4</td>
</tr>
<tr>
<td>s1</td>
<td>?,-/</td>
<td>s2</td>
<td>?,-/</td>
<td>s3</td>
<td>?,-/</td>
<td>s4</td>
</tr>
</tbody>
</table>

The learner has reason to pursue further learning, however. Some words are failing initial word evaluation. Specifically, stress is still indeterminate: the learner’s support is consistent with grammars that assign default initial stress and grammars that assign default final stress. The word r1s3, which surfaces as [paká], has both length features set underlyingly, so the input used for initial word evaluation has both stress features assigned values opposite their surface values, /páka/. The evaluation hierarchy for initial word evaluation, generated with the faithfulness-low bias, chooses /páka/[páka] as the sole optimum, mismatching the attested surface form of r1s3. This indicates that there is more for the learner to learn. But the learner is unable to learn anything further on the basis of single forms alone. This motivates the learner to search for a contrast pair.

7.9.3 Contrast pair learning

To summarize, the learner’s grammatical information at this point is as shown in (7.58) and (7.59) below.

(7.58) The support

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT [stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT [length]</th>
<th>NoLONG</th>
</tr>
</thead>
<tbody>
<tr>
<td>páką</td>
<td>páką</td>
<td>paká</td>
<td></td>
<td></td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>páką</td>
<td></td>
<td></td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>paká:</td>
<td>páką:</td>
<td>páką:</td>
<td></td>
<td></td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
</tr>
<tr>
<td>páką:</td>
<td>páką:</td>
<td>páką:</td>
<td></td>
<td></td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
</tr>
</tbody>
</table>

(7.59) The lexicon

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>r1</td>
<td>?,-/</td>
<td>r2</td>
<td>?,+/</td>
<td>r3</td>
<td>?,-/</td>
<td>r4</td>
</tr>
<tr>
<td>s1</td>
<td>?,-/</td>
<td>s2</td>
<td>?,-/</td>
<td>s3</td>
<td>?,-/</td>
<td>s4</td>
</tr>
</tbody>
</table>

Recall that a successful contrast pair must have a couple of properties. First, at least one of the words must have at least one unset underlying feature that needs to be set. Second, the environment morpheme(s) must alternate in the value of an unset feature.
The word r1s1 fails initial word evaluation. The bottom of the viable sub-
lattice is the input /paká:/, and candidate /paká:/[páka] is not only not optimal
for the evaluation hierarchy, but in fact inconsistent with the support in (7.58).
Clearly, one of the unset features for r1s1, that is, one of the stress features,
needs to be set. The word r1s1 is a good candidate for the first member of a
contrast pair. To form a proper contrast pair, the learner needs to find a stored
word that shares a morpheme with r1s1 such that the shared (environment)
morpheme has an unset feature which alternates in surface value between the
two words.

If the learner chooses to focus first on setting the stress feature of s1, then
it will adopt s1 as the contrast morpheme of r1s1, with r1 as the environment
morpheme of r1s1. It will then look for a stored word with a different suffix
combined with r1, where the different suffix surfaces differently than s1 does in
r1s1, and where r1 surfaces with a different value for stress (the unset feature)
than it does in r1s1. One such word is r1s3, which surfaces [paká]. s1 and
s3 surface non-identically in r1s1 and r1s3, respectively: s1 is unstressed, while
s3 is stressed. The environment morpheme, r1, alternates in stress between the
two words: r1 is stressed in r1s1, and unstressed in r1s3. Thus, r1s1 with r1s3 is
a reasonable contrast pair for the learner to construct and evaluate, given what
it currently knows.

The contrast pair r1s1 with r1s3 is precisely the pair that was presented above
in Section 7.6, and the processing of the contrast pair was illustrated in detail
there. The result shown there of processing the contrast pair is the setting of
the stress feature of s1 to –stress. The contrasting morphemes are s1 and s3,
and the unset features of those morphemes are the targets for possible setting.
If the learner takes the opportunity to also test the stress feature of s3, it will be
able to set it to +stress. The updated lexicon after processing this contrast pair
is shown in (7.60).

(7.60) The lexicon after processing the contrast pair

<table>
<thead>
<tr>
<th></th>
<th>r1</th>
<th>r2</th>
<th>r3</th>
<th>r4</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>/?,−/</td>
<td>/?,+/</td>
<td>/?,−/</td>
<td>/?,+/</td>
</tr>
<tr>
<td>s2</td>
<td>/?,−/</td>
<td>/?,+/</td>
<td>/+,−/</td>
<td>/?,+/</td>
</tr>
</tbody>
</table>

Having set more underlying features, the learner immediately searches for
possible additional non-phonotactic ranking information, by looking for words
in which the newly set features are realized unfaithfully. The stress feature
of s1 does not alternate in the language: s1 never surfaces as stressed. The
stress feature of s3 does surface unfaithfully, however, in r3s3 and r4s3. The
learner can evaluate r3s3 in an effort to find further non-phonotactic ranking information.

Because the learner is searching for ranking information, it selects the input at the top of the viable sublattice for the word r3s3. This will be the input in which every unset feature is assigned the value it has on the surface in r3s3. The surface form for r3s3 is [páka], and the only unset feature for this word is the stress feature of r3, so the constructed input will be /páká/. The learner now sets out to see if any additional ranking information is required to make the candidate /páká/[páka] optimal.

The learner constructs the constraint hierarchy using the markedness low bias, shown in (7.61).

(7.61) \{\text{Ident}\[\text{stress}]\} \gg \{\text{WSP}\} \gg \{\text{Ident}\[\text{length}]\} \gg \{\text{MainLeft, MainRight, NoLong}\}

Under that ranking, /páká/ has a tie for optimality between [páka] and [paká]. Because the correct output, [páka], is not the sole optimum, the learner constructs a new winner–loser pair, with /páká/[páka] as the winner, and /páká/[paká] as the loser, and adds it to the learner’s support, resulting in the support in (7.62).

(7.62) The support after processing r3s3

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páká</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>páká</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>páká</td>
<td>páká</td>
<td>páká</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>páka</td>
<td>páká</td>
<td>páká</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The learner’s new markedness-low ranking, given in (7.63), makes /páká/[páka] optimal.

(7.63) \text{Ident}[\text{stress}] \gg \text{WSP} \gg \text{Ident}[\text{length}] \gg \text{MainLeft} \gg \{\text{MainRight, NoLong}\}

To appreciate the nature of the learner’s ranking information at this point, observe that the faithfulness-low ranking for the learner’s support, given in (7.64), is not all that much different.

(7.64) \text{WSP} \gg \text{Ident}[\text{stress}] \gg \text{MainLeft} \gg \text{MainRight} \gg \text{Ident}[\text{length}] \gg \text{NoLong}
The learner has now eliminated more grammars as non-viable. L4, L7, and L9 are no longer viable given the learner’s support: all three prefer candidate /páká/[paká] to the above winner /páká/[páka] (and all three require Main-Right ≫ MainLeft). The remaining viable members of the typology are L13, L14, and L20.

Having successfully obtained both lexical information and ranking information by processing a contrast pair, the learner now returns to single form processing, to see if any more can be learned on the basis of single forms, given what it has learned from the contrast pair.

7.9.4 Second single form learning

The current state of the learner (after processing the contrast pair) is shown in (7.65) and (7.66), with (7.67) showing the hierarchy resulting from applying BCD.

(7.65) The support

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páká</td>
<td>páká</td>
<td>paká</td>
<td>W</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>páká</td>
<td></td>
<td></td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pák:</td>
<td>pak:</td>
<td>pák:</td>
<td>W</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pák:</td>
<td>pák:</td>
<td>pák:</td>
<td>W</td>
<td></td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pák:</td>
<td>pák:</td>
<td>pák:</td>
<td>W</td>
<td></td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(7.66) The lexicon

\[
\begin{array}{cccccccc}
\text{r1} & /?,-/ & \text{r2} & /?,+/ & \text{r3} & /?,-/ & \text{r4} & /?,+/ \\
\text{s1} & /-,?/ & \text{s2} & /?,?/ & \text{s3} & /+,,-/ & \text{s4} & /?,+/ \\
\end{array}
\]

(7.67) WSP ≫ IDENT[stress] ≫ MainLeft ≫ MainRight ≫ IDENT[length] ≫ NoLong

Some words, like r1s1, pass initial word evaluation, despite having some unset features. For r1s1, the bottom of the viable sublattice is /paka:/, but the candidate /paka:/ is already optimal under (7.67). But other words fail initial word evaluation, indicating potential for providing further information.

7.9.4.1 r3s3

Word r3s3 fails initial word evaluation. There is only one unset feature for the word, the stress feature for r3. To test the stress feature for r3, the learner
tests candidate /paká/[páka], which proves to be inconsistent with the learner’s support. This is shown in (7.68), where the bottom winner–loser pair is for r3s3 with r3 assigned the value –stress. The new winner–loser pair is the opposite of the second winner–loser pair, ensuring inconsistency. Thus, the learner is able to set r3 to +stress in the lexicon.

(7.68) Assigning –stress to r3 is inconsistent

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT [stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT [length]</th>
<th>NoLong</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>páka</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká</td>
<td></td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>páka:</td>
<td>páka:</td>
<td>páka:</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>paká</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Having set r3 to +stress, the learner goes searching for possible further non-phonotactic ranking information. However, the feature just set, stress for r3, doesn’t alternate in the language, so there are no available forms to test. The learner is now done with word r3s3.

7.9.4.2 r4s3, r1s3, r2s3
Processing word r4s3 will set r4 to +stress, in a way quite analogous to the setting of r3 to +stress by r3s3 above.

Processing (individually) r1s3 and r2s3 will set the stress features of r1 and r2, respectively. Both are set to –stress. At this point, all features, both stress and length, have been set for all four root morphemes. The learner’s lexicon at this point is given in (7.69).

(7.69) The lexicon after all of the roots have been set

<table>
<thead>
<tr>
<th>r1</th>
<th>/−,−/</th>
<th>r2</th>
<th>/−,+/</th>
<th>r3</th>
<th>/+,−/</th>
<th>r4</th>
<th>/+,+/</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>/−,?/</td>
<td>s2</td>
<td>/?,?/</td>
<td>s3</td>
<td>/+,−/</td>
<td>s4</td>
<td>/?,+/</td>
</tr>
</tbody>
</table>

When each of those features is set, the learner will check for non-phonotactic ranking information, but will not obtain any.
Exploiting output drivenness in learning

7.9.4.3 r1s2, r2s4

The learner will be able to set the stress feature for s2 when processing a word like r1s2, which surfaces as [páka]. Because r1 has been set underlyingly to –stress, if s2 is assigned +stress underlyingly, the resulting candidate /paká/[páka] will be inconsistent, due to the unavoidable dominance of IDENT[stress].

The learner will be able to set the stress feature for s4 when processing a word like r2s4, which surfaces as [pakáː]. If s4 is assigned –stress, the resulting candidate /paːkaː/[pakáː] will be inconsistent, due to the effect of MAINLEFT dominating MAINRIGHT. The two syllables of the input are identical (long and unstressed), so neither faithfulness constraint has a preference between /paːkaː/[pakáː] and /paːkaː/[páːka]. The inconsistency is shown in (7.70).

(7.70) Assigning –stress to s4 is inconsistent

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[stress]</th>
<th>ML</th>
<th>MR</th>
<th>IDENT[length]</th>
<th>NO LONG</th>
</tr>
</thead>
<tbody>
<tr>
<td>pákα</td>
<td>pákα</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>pákα</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pakáː</td>
<td>pakáː</td>
<td>paká</td>
<td></td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>pákαː</td>
<td>pákαː</td>
<td>pákα</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paːkaː</td>
<td>paːkaː</td>
<td>pákα</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The inconsistency allows the learner to set s4 to +stress. The resulting lexicon, after setting the stress features for s2 and s4, is shown in (7.71).

(7.71) The lexicon after setting the stress features of s2 and s4

<table>
<thead>
<tr>
<th>r1</th>
<th>/−,−/+</th>
<th>r2</th>
<th>/−,+/</th>
<th>r3</th>
<th>/+,−/+</th>
<th>r4</th>
<th>/+,+/</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>/−,−/+</td>
<td>s2</td>
<td>/−,−/</td>
<td>s3</td>
<td>/+,−/+</td>
<td>s4</td>
<td>/+,+/</td>
</tr>
</tbody>
</table>

Having set the stress feature for s4 to +stress, the learner checks for possible further non-phonotactic ranking information. Suffix s4 surfaces unstressed in r3s4 (as well as r4s4). The word r3s4 surfaces as [páka]. The features of both r3 and s4 are fully set, so the input for r3s4 is fully determined, /pákáː/. The learner knows that candidate /pákáː/[páka] is grammatical. To test this candidate, the learner constructs the markedness low hierarchy for its support. That hierarchy was previously stated in (7.63), and is repeated below.
With this ranking, the optimal candidate for input /páka:/ is [paká:], due to the dominance of \texttt{Ident[length]} over \texttt{MainLeft}. That candidate is then adopted as an informative loser, and another winner–loser pair is added to the learner’s support; the updated support is shown in (7.72). The new winner–loser pair requires that either \texttt{MainLeft} or \texttt{NoLong} dominate both \texttt{MainRight} and \texttt{Ident[length]}. Earlier phonotactic information requires that \texttt{NoLong} be dominated by \texttt{Ident[length]}, so the learner has determined that \texttt{MainLeft} must dominate \texttt{Ident[length]}. This turns out to be the final piece of the ranking puzzle; the learner has obtained all of the ranking information available from the data of the language.

7.9.4.4 The final learned grammar for L20

The only features still unset are the length features for s1 and s2. However, these will never be set; they are truly non-contrastive, and either value of the feature results in identical phonological behavior for each morpheme. The words containing s1 and the words containing s2 will pass initial word evaluation, as will all of the other words of the language. The learner now has a correct grammar for the entire language. The final support is given in (7.72), the final lexicon in (7.73), and the learner’s final constraint hierarchy, generated by BCD, is given in (7.74).

(7.72) The support (final)

<table>
<thead>
<tr>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>\texttt{Ident[stress]}</th>
<th>ML</th>
<th>MR</th>
<th>\texttt{Ident[length]}</th>
<th>\texttt{NoLong}</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>paká</td>
<td>W</td>
<td>W</td>
<td></td>
<td>L</td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká</td>
<td>W</td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>páká:</td>
<td>páká:</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>páká</td>
<td>páká</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>páká:</td>
<td>páká:</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td></td>
<td>W</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(7.73) The lexicon (final)

<table>
<thead>
<tr>
<th>r1</th>
<th>/−.−/</th>
<th>r2</th>
<th>/−.+/</th>
<th>r3</th>
<th>/+.−/</th>
<th>r4</th>
<th>/+.+/</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>/−.?./</td>
<td>s2</td>
<td>/−.?./</td>
<td>s3</td>
<td>/+.−/</td>
<td>s4</td>
<td>/+.+/</td>
</tr>
</tbody>
</table>
7.10 The map

The extended illustration just completed showed how the Output-Driven Learner efficiently succeeds in the case of language L20. Every underlying feature that needs to be set is set. The final support contains six winner–loser pairs, and they determine sufficient ranking conditions for the language. Three of the six winner–loser pairs in the support came from phonotactic learning. When learning with paradigmatic information, one contrast pair was necessary; the rest was based on single form learning.

In computer simulations, the above version of the ODL was tested on all twenty-four languages of the Stress/Length typology. For twenty-two of them, the language was completely learned. Two of the languages, L8 and L17, were problematic. The two languages are symmetric versions of the same pattern: the only difference between the two is the default location of stress (initial vs. final). The phenomenon causing the problem is a combination of restrictiveness and paradigmatic relations. This phenomenon, here labeled paradigmatic subsets, is discussed in detail in Chapter 8, along with the extended version of ODL that succeeds in learning all twenty-four languages of the Stress/Length typology, including L8 and L17.
8 Paradigmatic subsets

The two languages mentioned as problematic at the end of the previous chapter each have a particular kind of relation to a different language in the Stress/Length typology. One of the languages, L8, is a paradigmatic subset of language L7 (the other problematic language, L17, is a paradigmatic subset of L13 in just the same way). The nature of paradigmatic subsets is discussed in Section 8.1. As the term “subset” suggests, the relationship concerns restrictiveness, but of a more subtle and sinister sort than the restrictiveness addressed by phonotactic learning and BCD.

As is demonstrated in Section 8.2, neither the faithfulness-low bias of BCD nor the inconsistency detection of feature setting is adequate on its own to fully deal with the paradigmatic subset relations between L8 and L7. There is, however, a way to extend the Output-Driven Learner to deal with paradigmatic subsets, a way that further capitalizes on output drivenness. Section 8.3 discusses the key observation behind the solution: mapping greater numbers of inputs to the same outputs correlates with greater restrictiveness. Translating this observation into algorithmic terms compatible with the ODL is the subject of Section 8.4. The key is to use the structure of output-driven maps to gauge the size of the subspace of inputs mapping to a given output without having to actually evaluate all of the inputs in that subspace.

Section 8.5 gives an evaluation of the full Output-Driven Learner, including the addition described in Section 8.4. Simulations of the ODL are summarized, showing that it successfully learns all languages in the Stress/Length typology. A discussion of issues regarding the ODL is then presented: strengths and weaknesses of the proposal as given here, and questions that are potential topics for future research.

8.1 The phenomenon: paradigmatic subsets

8.1.1 Language L8, the subset language

Language L8 has stress attracted to length, with default final stress, and shortens long vowels in unstressed position. A ranking generating L8 is shown in (8.1). The language itself is given in (8.2).
Paradigmatic subsets

(8.1) WSP $\gg \text{Ident}[\text{length}] \gg \text{NoLong} \gg \text{MainRight} \gg \text{MainLeft} \gg \text{Ident}[\text{stress}]$

(8.2) The language L8

<table>
<thead>
<tr>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá:/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>pá:ka</td>
<td>paká</td>
<td>pá:ka</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
</tr>
<tr>
<td>paká</td>
<td>pá:ka</td>
<td>paká</td>
<td>pá:ka</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
</tr>
</tbody>
</table>

The rich input base contains four roots and four suffixes, but there is significant total neutralization in the language. The root pairs (r1,r3) and (r2,r4) and the suffix pairs (s1,s3) and (s2,s4) each neutralize in all environments. This is a consequence of the domination of Ident[stress]: the underlying specification of stress never has an effect on the output in this language, so morphemes differing only in underlying stress behave identically. In terms of distinguishable morpheme behavior, the language effectively has two roots and two suffixes, illustrated in (8.3).

(8.3) Language L8 compressed (all morphemes underlyingly unstressed)

<table>
<thead>
<tr>
<th>/pa/</th>
<th>/pa:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>pá:ka</td>
</tr>
<tr>
<td>paká:</td>
<td>-ka/</td>
</tr>
</tbody>
</table>

It is important to note that while the notation used for the underlying forms in (8.3) suggests that all morphemes are underlyingly unstressed, this is not in any way important; it would be equally correct to depict all of the morphemes, or any portion of them, as underlyingly stressed. An example is shown in (8.4). The underlying forms of the morphemes are not identical, but the surface alternation behaviors are.

(8.4) Language L8 compressed (all morphemes underlyingly stressed)

<table>
<thead>
<tr>
<th>/pá/</th>
<th>/pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>pá:ka</td>
</tr>
<tr>
<td>paká:</td>
<td>-ká:/</td>
</tr>
</tbody>
</table>
8.1 The phenomenon: paradigmatic subsets

The phonotactic inventory of L8 has three words.

(8.5) L8 Phonotactic Inventory: paká paká: pákə

8.1.2 Language L7, the superset language

Language L7 has lexically specified stress, with default final stress, and shortens long vowels in unstressed position. A ranking generating L7 is shown in (8.6). The language itself is given in (8.7).

(8.6) WSP \gg \text{Ident}\{\text{stress}\} \gg \text{Ident}\{\text{length}\} \gg \text{NoLong} \gg \text{MainRight} \gg \text{MainLeft}

(8.7) The language L7

\begin{align*}
\text{r1 = /pa/} & \quad \text{r2 = /pa:/} & \quad \text{r3 = /pá/} & \quad \text{r4 = /pá:/} \\
paká & \quad \text{pákə} & \quad \text{pákə} & \quad \text{pákə} & \quad s1 = /-ka/ \\
paká: & \quad \text{pákə:} & \quad \text{pákə} & \quad \text{pákə} & \quad s2 = /-ka:/ \\
paká & \quad \text{pákə} & \quad \text{pákə} & \quad \text{pákə} & \quad s3 = /-ká/ \\
paká: & \quad \text{pákə:} & \quad \text{pákə:} & \quad \text{pákə:} & \quad s4 = /-ká:/
\end{align*}

The rich input base contains four roots and four suffixes, and L7 has no total neutralization: every pair of roots behaves differently in at least one environment, as does every pair of suffixes. This is in part a consequence of the relatively high ranking of the faithfulness constraints \text{Ident}\{\text{stress}\} and \text{Ident}\{\text{length}\}.

The phonotactic inventory of L7 has four words.

(8.8) L7 Phonotactic Inventory: paká paká: pákə pákə

8.1.3 L8 is a paradigmatic subset of L7

A comparison of the phonotactic inventories of L8 and L7 quickly reveals that, phonotactically, L8 is a subset of L7. L7 contains all three surface forms of L8, plus an additional one, [pákə]. This form appears as the output of two words of L7: r3s1 and r3s2. For the three phonotactic words of L8, the ranking of L8 maps each of the sixteen possible inputs to one of those words, while the ranking of L7 maps only fourteen of the sixteen to one of those words. In L8, stress appears initially only when the initial vowel is long. In L7, stress can appear initially with both long and short initial vowels.
Paradigmatic subsets

A comparison of rankings generating the two languages accords with this. The rankings generating L8 and L7 are shown (again) in (8.9) and (8.10). The two rankings only differ in the location of one constraint, Ident[stress]. This faithfulness constraint is ranked significantly higher in the ranking for L7, yielding an expectation that L7 would be less restrictive. The r-measure of (8.9) is 5, while the r-measure of (8.10) is only 2.

(8.9)  \[ \text{L8: WSP} \gg \text{Ident[length]} \gg \text{NoLong} \gg \text{MainRight} \gg \text{Ident[stress]} \]

(8.10)  \[ \text{L7: WSP} \gg \text{Ident[stress]} \gg \text{Ident[length]} \gg \text{NoLong} \gg \text{MainRight} \gg \text{MainLeft} \]

The restrictiveness relation between the languages goes deeper, however. If we restrict our attention to only certain of the possible morphemes in L7, we can find a subparadigm that is surface-identical to L8, alternations and all. L8 is a paradigmatic subset of L7. L7 is shown again in (8.11), with a subsystem of forms shaded: the shading corresponds to combining only roots r3 and r4 with only suffixes s3 and s4. Compare the shaded region of L7 with the compressed version of L8 in which all morphemes are underlyingly stressed, given in (8.4) and repeated below.

(8.11)  \[
\begin{array}{cccc}
\text{r1} = /pa/ & \text{r2} = /pa:/ & \text{r3} = /pá/ & \text{r4} = /pá:/ \\
\text{paká} & \text{pá:ka} & \text{páka} & \text{pá:ka} \\
paká: & \text{paká:} & \text{páka} & \text{pá:ka} \\
paká & \text{paká} & \text{paká} & \text{pá:ka} \\
paká: & \text{paká:} & \text{paká:} & \text{pá:ka} \\
\end{array}
\]

(8.4)  \[
\begin{array}{cc}
\text{/pá/} & \text{/pá:/} \\
paká & \text{pá:ka} \\
paká: & \text{/-ká:/} \\
\end{array}
\]

The surface alternations of the morphemes in L8 are fully consistent with the surface alternations of a subset of the morphemes of L7. Any dataset consistent with L8 will also be consistent with L7. If all four underlying forms used have the same value for the stress feature, then stress contrast is lexically hidden.
An analogous projection of L8 into L7 is shown in (8.12) and (8.13), this time with all morphemes underlyingly set to \(-\text{stress}\).

(8.12) Language L7, with a different L8-equivalent subset shaded

<table>
<thead>
<tr>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>páká</td>
<td>páká</td>
<td>s1 = /-ká/</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>páká</td>
<td>s2 = /-ká:/</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
<td>s3 = /-ká:/</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>

When comparing (8.11) and (8.4), or (8.12) and (8.13), this does not seem particularly remarkable: a language in which only length is contrastive might be expected to exhibit such a relationship to a language in which both length and stress are contrastive. But this situation is more complex than that: there is another way to “project” L8 into L7. Compare the shaded region of L7 shown below in (8.14) with the compressed version of L8.

(8.13) Language L8 compressed (all morphemes underlyingly unstressed)

<table>
<thead>
<tr>
<th>/pa/</th>
<th>/pa:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>páká:</td>
</tr>
<tr>
<td>paká:</td>
<td>-ká:/</td>
</tr>
</tbody>
</table>

Choosing, from L7, underlying roots /pa:/ and /pá:/ and underlying suffixes /-ká/ and /-ká:/ produces the same system of morpheme behaviors. What’s different is that the underlying contrast between the roots is one of stress, rather than length: the roots are both long, but differ in underlying stress. The underlying long vowel for /pa:/ never surfaces in the observed environments (with suffixes /-ká/ and /-ká:/) because it is unstressed in those environments. The suffixes contrast only in length (as before).
The underlying forms shown for the morphemes in the shaded words of (8.11) will yield the same observed surface behaviors with the rankings for both L8 and L7; the same is true of the underlying forms for the shaded words in (8.12). Language L7 has contrast in stress while L8 does not, but the indicated underlying forms don’t differ in stress, so the difference between the two rankings isn’t apparent. The same is not true for the underlying forms for the shaded words in (8.14). The paradigm of words resulting from those underlying forms with the ranking for L8 is shown in (8.15). This set of morpheme behaviors is distinct from the compressed version of L8 overall. This is easily explained: the root underlying forms don’t contrast in length, so their behaviors are identical in L8.

(8.15) Words using the ranking of L8: the roots don’t contrast in length, so they behave identically

<table>
<thead>
<tr>
<th>/pa:/</th>
<th>/pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>pá:ka</td>
<td>pá:ka</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
</tr>
</tbody>
</table>

The subsystem shaded in (8.14) is not consistent with the ranking of L8. It is, however, consistent with the surface forms of L8. The existence of this subsystem has significance for learning. In a nutshell: when attempting to set underlying feature values, the learner is unable to set the values of some key features because different values of those key features work with different solutions (L8 vs. L7). Although the learner would prefer the more restrictive L8 ranking as a ranking, it is unaware of those distinctions in the restrictiveness of the rankings while attempting to set underlying feature values.

8.2 The problem: attempting to learn L8

If a learner is provided with a fully representative dataset for L7, it will have no difficulty distinguishing L7 from L8: as always, the superset language is distinguishable given data that aren’t consistent with the subset language’s grammar. The challenge arises when given data fully consistent with L8 (the subset language). A bias towards low-ranked faithfulness constraints (BCD) is designed to bias the learner towards the most restrictive ranking given the data. BCD alone as a method for enforcing restrictiveness proves to be inadequate here, not because of any inability to discern rankings, but because the learner, as characterized thus far, cannot learn enough about the underlying feature values.
8.2 The problem: attempting to learn L8

If the ODL, as described in Section 7.8, is applied to the data of L8, the learner will construct the following ranking information and lexicon and be unable to proceed further. When BCD is applied to the ranking information shown in (8.16), the result is a hierarchy that generates L8. The lexical information in (8.17), however, is incomplete: the length feature for root r1 has not been set. Further, the ranking information in (8.16) is also consistent with the hierarchy in (8.10), which generates L7.

(8.16) Learned ranking information for L8 (incomplete)

<table>
<thead>
<tr>
<th>Word</th>
<th>Input</th>
<th>Winner</th>
<th>Loser</th>
<th>WSP</th>
<th>IDENT[Length]</th>
<th>NoLong</th>
<th>MR</th>
<th>ML</th>
<th>IDENT[Stress]</th>
</tr>
</thead>
<tbody>
<tr>
<td>r2s2</td>
<td>paká: paká:</td>
<td>paká:</td>
<td>paká:</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r1s2</td>
<td>paká:</td>
<td>paká:</td>
<td>paká:</td>
<td>W</td>
<td>L</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>r2s1</td>
<td>p´a:ka</td>
<td>p´a:ka</td>
<td>paká</td>
<td>W</td>
<td>L</td>
<td>L</td>
<td>W</td>
<td>W</td>
<td></td>
</tr>
<tr>
<td>r1s1</td>
<td>paká</td>
<td>paká</td>
<td>p´a:ka</td>
<td>W</td>
<td>L</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(8.17) Learned lexicon for L8 (incomplete)

| r1 | /?,?/ | r2 | /?,+/ | s1 | /?,−/ | s2 | /?,+/ |

The heart of the problem can be illustrated with respect to the root morpheme that is consistently short and unstressed on the surface. Call this morpheme rx. In L8, rx must be short underlyingly, and stands in contrast to the other root, which is underlyingly long. With respect to the data of L8, assigning rx an underlying length value of short will clearly be consistent: it matches the surface value in every environment. In order to set the length feature, the learner must find evidence that an underlying length value of long for rx is inconsistent.

The subsystem shaded in (8.14) stands in the way. Root rx (r2 of L7 in the table) has an underlying form that is +long in that analysis, and yet is consistent, given the ranking for L7. The length feature of rx thus cannot be set via inconsistency detection: a value of −long is consistent with the ranking for L8, and a value of +long is consistent with the ranking for L7.

The learner will not fare any better when attempting to set the value of the stress feature for rx. In L8, only length is contrastive; underlying stress is irrelevant. Thus, rx in L8 is short underlyingly, but could be either unstressed or stressed underlyingly with no change in surface behavior. The consistency of these two analyses ensures that the stress feature will not be set via inconsistency
Paradigmatic subsets

detection: there is a solution (using the ranking for L8) that is consistent with
rx being assigned –stress underlyingly, and a solution (again using the ranking
for L8) that is consistent with rx being assigned +stress underlyingly.

The learner cannot get away with setting none of the features for rx, because
it must contrast with the other root. The ranking for L8 allows rx to be unset for
stress provided it is set to –long underlyingly, while the ranking for L7 allows
rx to be unset for length provided it is set to –stress underlyingly. One way or
the other, a feature of rx must be set, but every possible value of every feature
of rx appears in at least one consistent solution.

Restrictiveness considerations clearly favor a solution using the ranking of
L8, with the contrast between the roots being realized as a length contrast,
just as the suffix contrast is realized as a length contrast. The L8 solution
relies on restrictiveness enforced by the ranking, rather than ad hoc gaps in
the observed lexical forms. But getting the learner to conclude the L8 solution
requires that the process for setting underlying forms involve more than just
inconsistency detection; it must involve something further that is sensitive to
restrictiveness.

One point worth addressing is the behavior of CPR (see Section 6.7). CPR,
when run on the very same Stress/Length system, successfully learns all of
the languages, including L8, without any explicit further mechanism sensitive
to restrictiveness (Merchant 2008). How? The answer lies in the part of CPR
labeled initial lexicon construction. CPR assumes that it has, at the outset of
paradigmatic learning, access to data sufficient to determine which features of
morphemes alternate and which do not. The features that do not alternate are
set at this stage, to the value identical to their (sole) surface realization. For L8,
the two problematic features, length for r1 and r3, do not alternate; they always
surface –long. Initial lexicon selection sets both features to –long, the value that
those features must have in the target grammar for L8. Whether or not features
implicated in paradigmatic subset relations will always be non-alternating has
yet to be investigated. In any event, doing without initial lexicon construction
is a significant benefit to the ODL (see Section 6.8), a benefit that Justifies the
alternative pursued here.

8.3 Paradigmatic restrictiveness and the lexicon

Richness of the Base requires that a grammar be accountable for every input
allowed by the linguistic theory. A grammar is more restrictive not in virtue of
accepting fewer inputs, but by collectively mapping the same set of inputs onto
a smaller set of outputs. A more restrictive language, with fewer phonotactic
outputs, can be expected to map more of the inputs to those outputs. Restrictiveness is reflected in the lexicon in terms of the sets of inputs that are mapped to the same output by a grammar. For languages L8 and L7, this was shown in Section 8.1.2: L8 maps all sixteen inputs to the observed outputs (the outputs of L8), while L7 maps only fourteen of the sixteen inputs to the observed outputs. This can be separated by output: looking at each of the three distinct output forms from L8, the number of inputs mapped to that output by the ranking of each language is shown in (8.18). The difference in phonotactic restrictiveness between the two languages lies in the number of inputs mapped to the output [paká:).

(8.18) The number of inputs mapped to an output (column) by a language (row)

<table>
<thead>
<tr>
<th></th>
<th>paká</th>
<th>paká:</th>
<th>pák:ka</th>
</tr>
</thead>
<tbody>
<tr>
<td>L8</td>
<td>4</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>L7</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

The difference in restrictiveness between the languages becomes even more apparent when one considers not only phonotactic restrictiveness but also paradigmatic restrictiveness. It isn’t empirically sufficient for inputs to each map to some observed output: the underlying forms for morphemes, when combined to form inputs, have to map to the paradigmatically appropriate outputs. The learner must ultimately construct a lexicon of underlying forms for morphemes, rather than isolated inputs for words.

As illustrated in Section 8.1.3, there is more than one lexicon that correctly “projects” the paradigmatic relations of L8 into L7. In fact, there are six. Three of them were shown in Section 8.1.3; all six are shown below in (8.19). Note that there is not free combination of possible inputs for each morpheme: although more than one underlying form is possible for the first root of L8, and more than one underlying form is possible for the first suffix of L8, not all combinations of the two will work.

(8.19) The six possible paradigmatic projections of L8 into L7

<table>
<thead>
<tr>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
<th>s1 = /-ka/</th>
</tr>
</thead>
<tbody>
<tr>
<td>paká</td>
<td>pák:ka</td>
<td>pák:</td>
<td>pák:ka</td>
<td>s2 = /-ka:/</td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>pák:</td>
<td>pák:</td>
<td>s3 = /-ká/</td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>pák:ka</td>
<td>pák:</td>
<td>s4 = /-ká:/</td>
</tr>
</tbody>
</table>
With the ranking of L8, on the other hand, there are two possible underlying forms for each of the four morphemes, and any combination of them will work equally well. Thus, there are \(2^4 = 16\) lexica that yield the correct paradigmatic results with the ranking of L8, a stark contrast with the six lexica that yield correct results with the ranking of L7.
For a given set of morphemes, the number of possible lexica for those morphemes is the product of the numbers of possible underlying forms for each morpheme. If we are provided with data containing two roots and two suffixes, assuming four possible underlying forms for a morpheme, the total number of lexica is $4^4 = 256$. The number of successful lexica, relative to a given constraint ranking, will typically be much lower than the number of possible lexica and can reach the number of possible lexica only for a ranking in which each word is correctly generated by every combination of possible underlying forms for the morphemes of the word. This means that, for each word, all possible inputs are mapped to the correct output, entailing that every word has the same output. This should make intuitive sense: a most restrictive ranking is one generating a language with only one output, meaning that every possible input maps to that same output, and every word has that same output.

The presence of contrasts in a language necessarily reduces the number of successful lexica. If a contrast exists between two morphemes, then any lexicon in which those two morphemes have the same underlying form cannot be successful. Getting the two morphemes to exhibit not just distinct behaviors but the correct distinct behaviors will further reduce the number of successful lexica. In general, the number of successful lexica with respect to a given ranking can be expected to correlate with the degree of paradigmatic restrictiveness imposed by that ranking. More successful lexica means more neutralization (fewer contrasts) and thus greater restrictiveness.

8.4 The solution: Fewest Set Features

8.4.1 The Fewest Set Features procedure

In phonotactic learning, the r-measure was proposed as a property of grammars that correlated reasonably with restrictiveness and was much easier to work with computationally than extensional expressions of languages (lists of all possible forms of each language). Here I propose something analogous for lexicon learning: the number of set features in a hypothesized lexicon correlates negatively with the amount of neutralization, and thus the degree of restrictiveness of the overall hypothesized grammar. The number of set features is also rather easy to compute.

The notion of (un)set feature isn’t part of the linguistic theory itself, any more than the r-measure is. The relevant notion of (un)set feature is one that arises in the context of a learner that sets underlying features only when necessary. Restrictiveness itself is a property of the constraint ranking: a ranking imposes
restrictiveness via neutralization, mapping multiple inputs to the same output. The linguistic theory only requires that the grammar’s underlying form for each morpheme be one of the viable possibilities (given the ranking). The linguistic theory does not require that the lexical underlying forms explicitly represent the range of viable underlying forms for each morpheme; imposing neutralization is the job of the ranking, not the lexicon, as far as the linguistic theory is concerned.

A learner, however, can pursue the relationship between neutralization and restrictiveness in reverse, actively searching for a ranking that achieves greater neutralization as a means of identifying a more restrictive ranking. Further, it can use the learning construct of unset features to estimate the amount of neutralization taking place in a hypothesis (and thus the amount of restrictiveness). The learner’s goal is the most restrictive ranking; unset features are a means that the learner uses towards that end.

Recall from Section 7.7 that when the learner is evaluating a word for which some underlying features had not yet been set, the learner can evaluate all of the viable inputs at once by using the input at the bottom of the relative similarity order’s viable sublattice for that word. If there is a viable ranking mapping that input to the correct output, then by output drivenness the ranking will map every viable input to the correct output: the ranking neutralizes all of the viable inputs to the correct output. By declining to set any further features for the word at that point, the learner is hanging on to the possibility of retaining all of that neutralization.

If the candidate with the input at the bottom of the viable sublattice is inconsistent with the learner’s ranking information, then the space of possible inputs for the word needs to be reduced, by setting at least one feature. But the learner can still seek to retain as much restrictiveness as possible. The size of the space of viable inputs is determined by the number of unset features in the morphemes of the word. The more unset features there are, the more neutralization is implicitly being pursued. Setting a feature with respect to a word cuts the space of viable inputs for that word in half. Minimizing the number of set features is equivalent to maximizing the number of unset features, which in turn means greater neutralization.¹

¹ Looking beyond the scope of this book, when deletion of input segments is permitted, the space of possible inputs for an output can be infinite, as can be the corresponding space of feature instances. Maximizing the number of unset features in such a situation becomes meaningless: the number of unset features will remain infinite, so long as only a finite number of features are set. Minimizing the number of set features, however, remains perfectly meaningful. For that reason, the defining characterization is here given as minimizing the number of set features.
Recall the learned incomplete lexicon for L8 given in (8.17) and repeated here. The learner, given data from L8, had gotten stuck at this point, and in particular was unable to set the length feature of r1, because of the paradigmatic subset relation between L8 and L7.

(8.17) Learned lexicon for L8 (incomplete)

<table>
<thead>
<tr>
<th>r1</th>
<th>/?,?/</th>
<th>r2</th>
<th>/?,+l</th>
</tr>
</thead>
<tbody>
<tr>
<td>s1</td>
<td>/?,−l</td>
<td>s2</td>
<td>/?,+l</td>
</tr>
</tbody>
</table>

The learner can tell at this point that learning is incomplete via initial word evaluation. Of the four words in the dataset, three pass initial word evaluation. For instance, for the word r1s2, the bottom (of the viable sublattice) input, /pá:ka:/, is still mapped to the correct output, [paká:], by the learner’s evaluation ranking, as shown in (8.20), despite having three disparities (out of a possible four), one for each feature unset in the learner’s current lexicon.² For the words r1s2, r2s1, and r2s2, there is no reason to attempt to set any further features at this point.

(8.20) r1s2: input /pá:ka:/ has optimal output [paká:]

<table>
<thead>
<tr>
<th>/pá:ka:/</th>
<th>Output</th>
<th>WSP</th>
<th>IDENT[length]</th>
<th>NoLong</th>
<th>MR</th>
<th>ML</th>
<th>IDENT[stress]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observed</td>
<td>paká:</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td>Competitor</td>
<td>pá:ka</td>
<td>*</td>
<td>*</td>
<td>*!</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The situation is different for word r1s1. The length feature for s1 is already set in the lexicon, but the other three features are not, and so the candidate with the bottom input has three disparities. That candidate is not optimal under the learner’s evaluation ranking, as shown in (8.21). The tested candidate (with three disparities) loses to the competitor with an output identical to the input (with zero disparities). In fact, the winner–loser pair in (8.21) is inconsistent with the learner’s support.

(8.21) Input /pá:ka/ currently maps to output [paká], instead of the observed [paká] for r1s1

<table>
<thead>
<tr>
<th>/pá:ka/</th>
<th>Output</th>
<th>WSP</th>
<th>IDENT[length]</th>
<th>NoLong</th>
<th>MR</th>
<th>ML</th>
<th>IDENT[stress]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observed</td>
<td>paká</td>
<td>*!</td>
<td></td>
<td></td>
<td>*</td>
<td>**</td>
<td></td>
</tr>
<tr>
<td>Competitor</td>
<td>pá:ka</td>
<td></td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

² Regarding the candidates not shown: any candidate allowing a long vowel in an unstressed syllable will violate WSP (and therefore lose), while any candidate not preserving length in the stressed syllable will have two violations of IDENT[length] to the observed candidate’s one (and therefore lose).
This tells the learner that either some of the unset features for r1s1 need to be set, or more ranking information needs to be obtained (or both). The existing mechanisms for obtaining ranking information require that another underlying feature be set before any more ranking information can be obtained, so the best course of action is for the learner to try to set some underlying features. Specifically, the unset features of r1s1, the word that failed initial word evaluation, should be the targets.

The relative similarity lattice for r1s1 is shown in Figure 8.1. Because suffix s1 has already been set underlyingly to –long, inputs with s1 assigned +long are not viable, and are marked with shaded diamonds. The viable sublattice for r1s1, with the nonviable inputs removed, is shown in Figure 8.2.

The bottom node of the viable sublattice is the one that was used in initial word evaluation, with three disparities corresponding to the three unset features. At least one of those features must be set to the same value as its output correspondent (eliminating the corresponding disparity). Restrictiveness considerations favor setting the fewest such features.

Restrictiveness also warrants consideration of the other words, the ones that passed initial word evaluation: r1s2, r2s1, and r2s2. At present, all viable inputs for each of those words are mapped to their correct outputs, all by
the same viable ranking, the evaluation ranking (the one generated using the faithfulness-low bias). Restrictiveness is favored if no additional features for the inputs of those words need to be set (beyond any sharing of the feature being set in the failed word). If no further features are set for those words, then their sets of viable inputs will not shrink, maintaining full neutralization of each set of inputs. Words r2s1 and r2s2 each have two unset features, the stress features for both root and suffix. Word r1s2 has three unset features, the stress feature for s2 and both features for r1. So long as the ranking information does not change, these words will not warrant having any further features set.

The concern, with respect to the passing words, is if setting a feature in the failed word requires additional ranking information to succeed, and that additional ranking information causes some of the previously passing words to fail. If the additional ranking information requires further features to be set for the previously passing words, then restrictiveness as realized in the lexicon is receding further: more than one feature is being set.

The learner deals with this concern in the way that it tests a feature that it is considering setting. In the present example, the word being processed is r1s1, which has three unset features. The learner tests each feature separately. It tests the stress feature of r1 by constructing an input in which the stress feature of r1 matches its surface realization in the output for r1s1, but the length feature of r1 and the stress feature of r2 both mismatch their surface realizations in the output for r1s1. This is the input /pa:ka/, relative to r1s1’s correct output [paká]. The candidate /pa:ka/[paká] for r1s1 is evaluated for consistency with the learner’s support not in isolation, but in combination with the evaluation...
candidates for all of the words that pass initial word evaluation (the evaluation candidate for a word is the one formed with the input at the bottom of the viable sublattice). The evaluation candidates, /pa:ka:/ for r1s2, /pa:ká/[pá:ka] for r2s1, and /pá:ka:/ for r2s2, are combined with the candidate for r1s1, and all four are evaluated for consistency together: is there a viable ranking that makes all four candidates optimal? The option of setting the stress feature of r1 to –stress will be rejected if there is no such ranking.

In Figure 8.3, the inputs that, when combined with the three evaluation candidates for the passing words, are inconsistent with the learner’s existing ranking information are shaded. The bottom node is inconsistent on its own. The other two shaded nodes correspond to setting the stress feature of s1 to +stress, and setting the stress feature of r1 to –stress, respectively. In each of those two cases, setting the relevant feature causes inconsistency with the evaluation candidates of the words {r1s2, r2s1, r2s2}; any resolution that retained setting s1 to +stress or r1 to –stress would require setting at least one other feature as well.

The second row from the bottom in Figure 8.3 contains nodes for the inputs in which two of the unset features mismatch their surface correspondents, while one matches (the candidates with two disparities each). Two of the three are inconsistent, as just described, but one is consistent, the input /pá:ka/, which matches the surface in having r1 assigned the value –long, but mismatches the output in having r1 assigned +stress and s1 assigned –stress. The combination of this input with the three evaluation candidates for the other words is consistent with the learner’s support.
It follows from output drivenness that the three inputs above /páka/ in the relative similarity order are also consistent. But there is one consistent input, /pa:ká/, that is not ordered with respect to /páka/. It is in the next row up in the diagram, and matches the output in two features (the stress features), but mismatches the output in having r1 assigned the value +long. Adopting that solution instead would mean setting two features in the lexicon, the stress feature of r1 to –stress and the stress feature of s1 to +stress, while leaving the length feature of r1 unset.

If the length feature of r1 is set to –long, then the other two unset features (the stress features) need not be set to maintain consistency. This means that the learner is attempting to map four inputs, including /páka/, to the output [paká]. If the length feature of r1 is left unset (allowing it to possibly be +long), then both of the stress features must be set in order to maintain consistency, so only one underlying feature would remain unset. That would mean that the learner is only attempting to map two inputs to the output: /pa:ká/ and /paká/. Restrictiveness considerations weigh in favor of the option mapping the most inputs to the output, that is, the one with the fewest set features, which is the input /páka/, achieved by setting r1 to be –long.

A simple implementation of this preference within the ODL is as follows. The learner learns as much as possible using single form learning and contrast pairs. If no more underlying features can be set in that fashion, but one or more words still fail initial word evaluation, then the learner selects one of those words for further processing. The learner evaluates the inputs for the selected word with only one unset feature matching its output correspondent, by testing it, in combination with the evaluation candidates for the passed words, for consistency with respect to the learner’s support. In other words, form a candidate for each input on the second row from the bottom of the viable sublattice and test each of them. If one of the candidates passes the test, set the matching feature for that candidate’s input. If none of the candidates in that row passes the test, continue on the next row up in the viable sublattice, until an input is found that passes the test and set the corresponding features for that candidate. Once a feature has been set, return to learning as before: look for instances in which the newly set feature is unfaithfully realized to obtain new ranking information, pursue single form learning, process contrast pairs, etc.

The procedure just described is here labeled the **Fewest Set Features** procedure. The description just given leaves several computational issues unaddressed; see Section 8.4.2 for further discussion of the computational issues. It is, however, enough to yield successful learning of the final two languages of the
Stress/Length typology, L8 and L17. With the addition of Fewest Set Features, the ODL successfully learns all twenty-four languages of the typology.

In the case of L8, the learner applies Fewest Set Features by selecting r1s1, the single word failing initial word evaluation, for further processing. The learner tests each of the three viable inputs for r1s1 that have two unset features mismatching the surface: /pá:ká/, /páka/, and /pa:ka/ (the three inputs in the second row from the bottom of the relative similarity lattice for r1s1). Of those three, only one passes the test of Fewest Set Features. Because any other passing inputs will be further up the lattice, and thus will be predicted to be less restrictive, the learner need not bother with them. The input /páka/ matches the output of r1s1, [paká], in the value of one unset feature, the length feature of r1. The learner thus sets r1 to –long underlyingly. The result of this is the lexicon shown in (8.22).

(8.22) Learned lexicon for L8 (complete)

| r1 | ?,–l | r2 | ?,+l |
| s1 | ?,–l | s2 | ?,+l |

Once the learner sets r1 to –long, it has successfully learned L8. The lexicon is complete: the length feature of every morpheme has been set. None of the stress features need to be set. The learner’s current ranking information is sufficient, and all words pass initial word evaluation.

8.4.2 Algorithmic details of Fewest Set Features

The Fewest Set Features procedure is the least developed and least understood of the learning proposals in this book. The implementation described here worked for languages L8 and L17, and will work for similarly simple cases, but it is unclear how generally successful it will be as currently stated. Future research will need to determine the limits of this implementation, and what more sophisticated implementations of the Fewest Set Features idea could be justified.

It is entirely possible that feature values can be set improperly if the Fewest Set Features procedure is invoked prematurely. In the simulations for the languages of the Stress/Length typology, the Fewest Set Features procedure was invoked only during learning with morphological awareness, and only after all the specified inconsistency detection techniques (single forms, contrast pairs) had been invoked without further progress. That simple approach might not be feasible for more complex systems; a more sophisticated criterion for the
8.4 The solution: Fewest Set Features

invocation of Fewest Set Features may be needed. Even in cases where the learner is able to wait until inconsistency detection has been exhausted, it is not known if Fewest Set Features will ever incorrectly set a feature.

As described, Fewest Set Features focuses on a single word that has failed initial word evaluation. It finds the feature (or smallest set of features) that needs to be set for that word, while ignoring any other words that previously failed initial word evaluation. The exclusion of other words previously failing initial word evaluation is based on the expectation that those words will involve other morphemes and may need other features to be set. If the previously failing words were included, then inconsistency could ensue not as a result of the feature being tested but because of separate unresolved issues with the failing words. Focusing on one failed word at a time keeps the combinatorics of possible underlying forms under control.

The inclusion of the words previously passing initial word evaluation is motivated by concerns about consistency and restrictiveness. In the L8 illustration, when the key input for r1s1 was tested (the one with r1 assigned the value -long), it was not only consistent, but also mapped to the correct output with the learner’s evaluation ranking. No new ERCs were created, and there was no need to alter the learner’s support to accommodate the newly set feature. In such an instance, the evaluation ranking will not change, and words not containing the newly set feature should pass/fail initial word evaluation exactly as they did before the feature was set. The need to explicitly consult the words that previously passed initial word evaluation arises when the input proposed by Fewest Set Features is consistent, but is not mapped to the correct output by the evaluation ranking. This requires that additional ranking information be added to the learner’s support, and there is no guarantee that the words that previously passed initial word evaluation would pass with respect to the new evaluation ranking. Testing the new candidate for the previously failed word in tandem with the evaluation candidates for the words previously passing initial word evaluation determines if there is a viable constraint hierarchy making all of those candidates optimal. If no such hierarchy exists, then the set of candidates will be collectively inconsistent, and the learner will decline to set the proposed feature at that point. In deference to restrictiveness, the learner will first test the possibilities of setting other single features, before considering options requiring the setting of more than one feature.

Fewest Set Features does not specify any basis for choosing, from among the words that fail initial word evaluation, which one to focus on first. In the illustration using L8, it didn’t matter. But it is not obvious that that will always be the case. Perhaps, in some more complex linguistic systems, applying Fewest
Set Features to one word first will affect how subsequent words are ultimately processed by learning. If choosing different words for Fewest Set Features ultimately resulted in different grammars (different underlying forms and/or different rankings), then the issue could be a significant one.

It may be possible that more than one feature (or, more generally, more than one same size set of features) could separately pass the test imposed by Fewest Set Features. Both possibilities would allow the same number of unset features, predicting equivalent restrictiveness. It remains to be determined under what conditions this could happen, and how a learner should best resolve it.

8.4.3 The relation to maximum likelihood

Enforcing restrictiveness by maximizing the number of inputs that map to each observed output has a conceptual similarity to work in statistical learning based on the method of maximum likelihood (Fisher 1922). The method of maximum likelihood chooses from among a class of models on the basis of the joint probability of occurrence assigned by each model to the observed dataset. The joint probability of a dataset (for a given model) is sometimes referred to as the likelihood of the dataset. The method of maximum likelihood prefers the model that maximizes the likelihood of the dataset. The preferred model is the one that holds the observed dataset as most expected. Put another way, the preferred model is the one under which the observed dataset is the least surprising or unusual. In essence, it takes the likelihood of the dataset given a model, and treats the likelihood like a property of that model, preferring the model with the highest value of the property.

The method of maximum likelihood is automatically biased toward more restrictive models, and in an attractive way. The key is that a model defines a probability distribution over possible forms, and the total amount of probability is fixed: the sum of the probabilities of all possible forms must be exactly 1. Thus one model, relative to another, cannot assign higher probabilities to some forms without compensatorily assigning lower probabilities to other forms; it always has to sum to 1. The method of maximum likelihood will prefer the model that assigns the highest probabilities to the forms that are observed (included in the dataset), and as a side effect assigns the lowest probabilities to the forms that are not observed. Models will be preferred that assign very little probability to forms that are systematically missing from the observed dataset.

To see the relationship between maximum likelihood and the subset problem in language learning, think of models as probabilistic grammars: a grammar assigns a probability of occurrence to each possible output. An output that might be said not to occur in the language defined by a particular grammar
will be assigned a very small probability of occurrence, in the extreme case a probability of zero.\footnote{In practice, allowing models to assign zero probability to outcomes frequently causes problems and is generally avoided, but for reasons that need not concern us here.} Recall the phonotactic output inventories of L8 and L7, repeated below.

\begin{center}
(8.5) \hspace{1cm} \text{L8 Phonotactic Inventory: paká paká: pá:ka}

(8.8) \hspace{1cm} \text{L7 Phonotactic Inventory: paká paká: pá:ka páka}
\end{center}

A probabilistic grammar for L8 would be expected to assign meaningfully large probabilities to the three outputs of L8, paká, paká:, and pá:ka, while assigning a very small probability to output páka. A probabilistic grammar for L7 would be expected instead to assign meaningfully large probabilities to all four outputs. The L8 grammar could assign a probability of almost $1/3$ to each of the three outputs in the inventory. The L7 grammar cannot assign more than a probability of $1/4$ to each of the forms in its inventory (if they are to be equally probable). Because L8 assigns very little probability to output páka, it has more left over to assign to each of the other three outputs. Thus, given a dataset which contains only the three outputs of the L8 inventory, the L8 grammar will assign a higher likelihood to the dataset than the L7 grammar. The method of maximum likelihood will prefer the more restrictive L8 grammar.

The maximum likelihood approach to enforcing restrictiveness is attractive because it works while focusing solely on the actually observed data. While the difference between L8 and L7 is most easily described in terms of the output that isn’t in L8, páka, the method of maximum likelihood expends no effort speculating on what outputs are systematically missing from the dataset and instead focuses on maximizing the probability of occurrence of the forms that are found in the dataset. This is particularly useful in plausible language situations, where the actual languages are infinite, and no dataset will actually contain all of the possible outputs for any language. The key to this is that probability is a fixed quantity, distributed over the possible outputs, so that assigning more to outputs that are observed necessarily means assigning less to outputs that are not observed.

Jarosz (2006, 2009) uses the method of maximum likelihood to enforce restrictiveness in phonotactic learning. Instead of each model being associated with a single Optimality Theoretic constraint ranking, her approach to phonotactic learning involves models that are associated with probability distributions over all possible rankings of the constraints. A model in this sense will have
higher likelihood to the extent that it assigns greater probability to the most successful rankings.

In Jarosz’s work, each model presumes a fixed uniform distribution over the space of possible inputs, meaning that each input is equally likely to occur. The probability of an output for a given ranking is the sum of the probabilities of occurrence of the inputs that map to that output. The overall probability of an output for a given model is the sum, over each possible ranking, of the probability of the output for that ranking weighted by the probability of the ranking itself. Put slightly differently, the joint probability of a ranking and an input is the product of the probability of the ranking and the probability of the input, and the probability of an output is determined by adding together the joint probabilities of each ranking/input combination that generates that output.

Because the probabilities of the inputs are all the same, the probability assigned to a given output by a ranking is directly proportional to the number of inputs that are mapped to that output by that ranking. A ranking which maps more inputs to an output will assign a greater probability to that output than a ranking which maps fewer inputs to that output. Thus, selecting the model with maximum likelihood is equivalent to selecting the model that assigns the greatest probability to the rankings that map the most inputs onto the observed outputs. Because of the principle of Richness of the Base, the space of possible inputs is fixed across all models and serves as a fixed quantity that must be distributed over the possible outputs (analogous to probability in a probabilistic model). This allows the assignment of inputs to serve in a restrictiveness-enforcing capacity just as probability is: mapping more inputs to the observed outputs necessarily means mapping fewer inputs to the unobserved outputs.

Jarosz only actively enforces restrictiveness in this way during phonotactic learning; her proposal for post-phonotactic learning (with morphological awareness) abandons the distribution over inputs for words and instead uses a (non-fixed) probability distribution over possible lexica of underlying forms for morphemes. A distribution over lexica consists of a set of independent distributions, one for each morpheme, each defining a distribution over the possible underlying forms for that morpheme. The learner alters these underlying form distributions during the course of learning, along with the ranking distribution. This allows maximum likelihood to be used to learn underlying forms, by preferring lexical distributions that assign greater probability to underlying forms that are successful.

The distribution over inputs for a given output is now a function of the distributions over the underlying forms for the morphemes of the word. Because
the learner can alter the underlying form distributions, the distribution over the possible inputs is no longer fixed and in general will not be uniform. This breaks the link that existed in phonotactic learning between the quantity of possible inputs and the quantity of probability to be allocated. To increase the probability of an observed output, the learner is no longer forced to prefer rankings that map more inputs to that output; it now has the option of shifting more lexical probability towards the inputs that do map to the output, thus increasing the probability of the output without increasing the number of inputs mapping to the output.

The ODL, as described in Chapter 7, is non-probabilistic: features either have a set value in the lexicon, or they do not, and ERCs either do or do not reside in the learner’s support. In effect, the learner has a subspace of lexica under consideration, and a subspace of rankings under consideration; the only “probabilistic” distinction is the distinction between possibilities that are still under consideration and possibilities that are not. However, the Fewest Set Features procedure does associate the mapping of more inputs to observed outputs with greater restrictiveness. It isn’t using maximum likelihood, but it is using maximum input quantity (in the form of minimum number of set features). Richness of the Base is fully retained, so the base of possible inputs (and possible underlying forms for morphemes) remains a fixed quantity, and assigning more inputs to the observed outputs means assigning fewer to unobserved outputs.

Computationally, Jarosz’s use of maximum likelihood to enforce restrictiveness is significantly different from the ODL’s use of Fewest Set Features. Jarosz’s proposal uses maximum likelihood to enforce restrictiveness during phonotactic learning and then releases the uniform distribution over possible inputs when learning underlying forms, relying on the bias introduced by the prior phonotactic learning to retain restrictiveness. The ODL does not use Fewest Set Features during phonotactic learning, relying instead on the faithfulness-low bias of BCD to enforce restrictiveness. During the learning of underlying forms, the ODL selectively invokes Fewest Set Features in circumstances where the restrictiveness complications of paradigmatic subsets can prevent straight inconsistency detection from learning the complete grammar.

Fewest Set Features shares with maximum likelihood the property of enforcing restrictiveness while focusing only on the data that are observed, rather than attempting to explicitly generate and evaluate unobserved forms. The example in Section 8.4.1 is based on a single morphologically analyzed word and works within the space of possible inputs for that morphologically identified word,
restricted by the set features in the underlying forms for each of the morphemes in the word. But even within those restrictions, inputs that are mapped to the output of the word are inputs not mapped to other outputs; the “zero sum” logic still follows from the Richness of the Base. Maximum likelihood attempts to maximize the quantity of probability assigned to the observed words, while Fewest Set Features attempts to maximize the quantity of inputs assigned to the observed words. In both cases, assigning more to the words that are observed implicitly assigns less to words that are not observed.

8.4.4 Summary: restrictiveness in the lexicon
Not only can paradigmatic subset relations be addressed, but they can be addressed by exploiting the same property of output drivenness used elsewhere in learning. Because one input can be a proxy for a whole space of inputs with respect to optimality and inconsistency, the learner can gain a sense of how many viable input forms can map to a given output without having to generate and test every single one of them. This is a great computational advantage.

This capacity for evaluating restrictiveness in the lexicon provides a second motivation for having a learner set underlying features only when necessary, in addition to the original motivation of avoiding premature commitment. Unset features can implicitly turn an input into a set of inputs, one for each combination of values for the unset features. Having more unset features pushes the learner to try to map larger numbers of inputs to observed outputs, increasing restrictiveness even during the learning of underlying forms.

8.5 Evaluating the Output-Driven Learner

8.5.1 The Output-Driven Learner (revised)
The revised outline of the ODL, shown in (8.23), adds Fewest Set Features to the outline given in Section 7.8.

(8.23) Revised Outline of the Output-Driven Learner
1. Phonotactic Learning (prior to morphological awareness).
2. Single Form Learning on all stored words:
   a. If new grammar information was obtained for a word, repeat single form learning on all stored words.
3. If no stored words fail initial word evaluation:
   a. Wait until a new word is observed.
   b. Perform single form learning on that word.
   c. If new grammar information was obtained, return to step 2. Otherwise, repeat step 3.
4. If a stored word fails initial word evaluation:
8.5 Evaluating the Output-Driven Learner

a. Look for a word sharing all but one morpheme with the failed word, such that an unset feature in a shared morpheme alternates between the two words. Form a contrast pair with the two words.
b. Attempt to set each unset, non-alternating feature of the contrast pair.
c. If a feature was newly set, look for instances in other stored words where the newly set feature is unfaithfully realized and test for new (non-phonotactic) ranking information. Then return to single form learning (step 2).
d. If no feature was newly set, continue searching for an informative contrast pair for the word (step 4a).
e. If no informative contrast pair is found for that word, repeat step 4 for any other stored words that fail initial word evaluation.
5. If no informative contrast pair was found:
a. Select a word that is failing initial word evaluation.
b. Perform Fewest Set Features with the selected word.
c. Return to single form learning (step 2).

In the simulations run with the ODL, all of the available words were provided to the learner during phonotactic learning, and again during initial Single Form Learning. This simplified the determination of when to search for a contrast pair: a contrast pair was constructed when the learner could not make further progress on the given words via single form learning, and at least one word was still failing initial word evaluation. In the larger learning picture, more will likely need to be said about when a learner searches for contrast pairs. Some criteria could be specified, perhaps the observation of a certain concentration of paradigmatically related words.

The only real concern addressed by limiting the frequency of contrast pair processing is computational effort. The criteria for invoking Fewest Set Features is more significant, because premature invocation could actually cause learning to fail. The learner should use some criteria making it likely that it has seen all the relevant information it is going to see about a particular word or set of words before invoking Fewest Set Features.

8.5.2 Simulation results
A computer simulation of the ODL was run on all twenty-four languages of the Stress/Length system. All twenty-four languages were learned successfully.

The largest support constructed in the process of learning any of the languages was ten winner–loser pairs; twenty-three of the twenty-four languages had fewer than ten pairs in the resulting support. The number of phonotactic winner–loser pairs ranged, across all of the languages, between one and six.
The number of non-phonotactic winner–loser pairs ranged between zero and four.

Of the twenty-four languages in the Stress/Length typology, two were learned from phonotactic learning alone. Those were the two languages that each contained only one word, so that (literally) everything in the language was predictable. Six languages were learned only on the basis of phonotactic learning and single form learning. Fourteen languages required the construction of a contrast pair during learning. Two languages required the invocation of the Fewest Set Features procedure. Neither language invoking Fewest Set Features benefitted from an informative contrast pair. The simulation details reported below are partitioned into four sets along these lines.

There is no single natural metric for measuring the computational effort of algorithms like these. The simulation details below give information at more than one level of analysis of the algorithm. In the column headings for the summary tables, “SF” stands for single form processing, and “CP” stands for contrast pair processing. “SF Passes” is the number of times that single form learning made a pass through the observed words. “SF IWE Fails” is the number of times, during single form processing, that some word failed initial word evaluation. “SF Feature Evaluations” is the number of times some feature was evaluated for feature setting during single form processing. “CPs Generated” is the total number of contrast pairs generated during the course of learning a language. “CP Feature Evaluations” is the total number of times some feature was evaluated for feature setting during contrast pair processing.

8.5.2.1 No contrast languages
Languages: L1, L21

These two languages have global neutralization to a single output. Phonotactic learning constructs one winner–loser pair, and the hierarchy generated by BCD at that point is sufficient to map every input to the single attested output. No underlying features are set.

8.5.2.2 Single form learning, no contrast pairs
Languages: L3, L5, L11, L18, L22, L24

These six languages are successfully learned with a combination of phonotactic learning and single form learning. No contrast pair construction is attempted by the learner, because the initial run of single form learning is sufficient. A performance summary of the learning of these languages is given in (8.24).

Two of the languages detected a total of four failures with initial word evaluation, and made a total of sixteen feature setting evaluations. The other
four languages detected a total of seven failures with initial word evaluation, and
made a total of twenty-two feature setting evaluations. None of the languages
obtained any non-phonotactic ranking information; in all six cases, phonotactic
ranking information was sufficient.

To put the number of feature evaluations in perspective, the entire lexicon
has eight morphemes, with two features each, for a total of sixteen features.
For these languages, the number of feature setting attempts is at or only a little
above the number of features.

### Simulation performance for languages requiring only single form learning

<table>
<thead>
<tr>
<th>SF Passes</th>
<th>SF IWE Fails</th>
<th>SF Feature Evaluations</th>
<th>CPs Generated</th>
<th>CP Feature Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>Range</td>
<td>Mean</td>
<td>Range</td>
<td>Mean</td>
</tr>
<tr>
<td>2</td>
<td>2.2</td>
<td>6</td>
<td>4.7</td>
<td>20</td>
</tr>
</tbody>
</table>

#### Contrast pairs


Fourteen of the twenty-four languages are learned with a combination of
phonotactic learning, single form learning, and contrast pair processing. None
of the languages needed more than one informative contrast pair, a fact quite
likely attributable to the smallness of the linguistic system. Contrast pair selec-
tion was quite rapid: for six of the fourteen languages, the first contrast pair
constructed by the learner was informative and sufficient, so no other contrast
pairs were constructed or processed. For the other eight languages, the first
contrast pair constructed was uninformative, but the second contrast pair was
informative and sufficient. The learner did not construct more than two contrast
pairs for any of these fourteen languages.

For each of the fourteen languages, the single informative contrast pair
resulted in the construction of one additional winner–loser pair (added to the
support). The total number of non-phonotactic winner–loser pairs added to
the support, including the one directly resulting from a contrast pair, ranged
between one and four. The language with the largest support had a total of ten
winner–loser pairs, six phonotactic pairs, and four non-phonotactic pairs.

The total number of feature evaluations for contrast pair processing is the
number of times some feature was evaluated with respect to some local lexicon,
summed across all processed contrast pairs. When a feature is tested with
respect to multiple local lexica in a contrast pair due to the presence of one
or more alternating features in the contrast pair, each local lexicon is counted separately and added to the total count of CP Feature Evaluations.

The table in (8.25) contains an additional main column, “Total Feature Evaluations,” which is based on the sum, by language, of the feature evaluations in feature setting for single form processing and contrast pair processing. The range endpoints for the total feature evaluations does not match the sums of the range endpoints for SF Feature Evaluations and CP Feature Evaluations, because the minimum for SF did not occur in the same language as the minimum for CP (and similarly for the maximum values).

(8.25) Simulation performance for languages requiring contrast pairs

<table>
<thead>
<tr>
<th>SF Passes</th>
<th>SF IWE Fails</th>
<th>SF Feature Evaluations</th>
<th>CPs Generated</th>
<th>CP Feature Evaluations</th>
<th>Total Feature Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>Range</td>
<td>Mean</td>
<td>Range</td>
<td>Mean</td>
<td>Range</td>
</tr>
<tr>
<td>5</td>
<td>4..7</td>
<td>43</td>
<td>26..55</td>
<td>82</td>
<td>66..99</td>
</tr>
</tbody>
</table>

8.5.2.4 Fewest Set Features
Languages: L8, L17

These two languages each required two rounds of the Fewest Set Features procedure to succeed. After phonotactic learning, the first pass through single form learning sets all but two of the length features, and adds the final winner–loser pair to the support. After that, single form learning is unable to set any further features. The learner then attempts to construct a contrast pair, but none of the plausible contrast pairs sets another feature. The learner then uses fewest set features, and sets one of the remaining length features.

Once the learner has set one length feature via Fewest Set Features, it makes another pass through single form learning and then works through all possible contrast pairs again, but none of those set another feature. At that point, the learner again turns to Fewest Set Features, which sets the final length feature. At that point, the learner is finished: all words pass initial form evaluation. A performance summary of the learning of these two languages is given in (8.26).

The total number of contrast pairs generated is the sum across both passes through contrast pair construction: eight contrast pairs on the first pass, and four contrast pairs on the second pass. In the case of these two languages, each contrast pair was responsible for seven feature evaluations.

For each of the two languages, the total number of feature setting attempts across both single form learning and contrast pair learning was 130.
8.5 Evaluating the Output-Driven Learner

(8.26) Simulation performance for languages requiring the Fewest Set Features procedure

<table>
<thead>
<tr>
<th>SF Passes</th>
<th>SF IWE Fails</th>
<th>SF Feature Evaluations</th>
<th>CPs Generated</th>
<th>CP Feature Evaluations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>Range</td>
<td>Mean</td>
<td>Range</td>
<td>Mean</td>
</tr>
<tr>
<td>3</td>
<td>3..3</td>
<td>14</td>
<td>14..14</td>
<td>46</td>
</tr>
<tr>
<td>12</td>
<td>12..12</td>
<td>84</td>
<td>84..84</td>
<td></td>
</tr>
</tbody>
</table>

8.5.2.5 A little perspective

To put the simulation performance results in perspective, recall the combinatorics of the Stress/Length system from Section 6.2.4. The system contains six constraints. Each lexicon contains eight morphemes, with two features each, for a total of sixteen features. There are 65,536 possible lexica, and 720 possible total rankings, for a total of 47,185,920 possible grammatical hypotheses. Across all twenty-four languages, the maximum support size was ten winner–loser pairs. The maximum number of feature evaluations was 130, which occurred for the two languages requiring the use of fewest set features. A single contrast pair, with a total of six features, has $2^6 = 64$ possible local lexica if none of the features have yet been set. Evaluating all of the local lexica for a single such contrast pair (applying inconsistency detection to each local lexicon), as CPR does, would be equivalent to nearly half of the maximum total number of feature setting evaluations (each applying inconsistency detection to a local lexicon) for an entire language for the ODL.

8.5.3 Issues

The ODL succeeded in overcoming the combinatorial explosiveness of lexical search and lexicon/ranking interaction. Simultaneously, it succeeded in properly enforcing restrictiveness in learning, even with respect to instances of paradigmatic subset languages. This success is crucially dependent upon the output drivenness of the languages of the linguistic system. The structure imposed on the lexicon by output drivenness is quite powerful in learning. This section quickly sketches some issues that arise with the ODL as currently described, representing possible topics of future investigation.

The use of production-directed parsing with particular constraint hierarchies for informative loser generation is reasonably effective, but can fall short, as described in Section 5.4.5. The ODL provides an opportunity to improve on this by generating constraint hierarchies via different biases in different circumstances, notably the generation of a hierarchy using the markedness low
bias during the pursuit of non-phonotactic ranking information, as described in Section 7.7.2. There is still no guarantee that all of the ranking information implicit in a winner will be obtained through the application of production-directed parsing and a single stratified hierarchy, whatever construction bias is used. The extent to which such failure of exhaustiveness could significantly impact the performance of the ODL, and under what circumstances, is not known.

Biased Constraint Demotion does a respectable job of enforcing restrictiveness with respect to a support, but it is not guaranteed to find the most restrictive ranking during phonotactic learning, as discussed in Section 5.6.4.3. Paradigmatic information (post-phonotactic learning) can give the learner significantly more leverage in enforcing restrictiveness, as alternations can positively attest to the imposition of disparities. The ODL adds an additional dimension to the enforcement of restrictiveness by enforcing it in the lexicon as well as with ranking bias. Setting features only when necessary, and testing words with respect to the bottom node of the viable sublattice, allow the learner to further enforce restrictiveness, and in a (relatively) computationally tractable fashion. It is not currently known what residual effects the limitations of BCD could possibly have on the final outcome of learning, given the additional force provided by the ODL.

With the ODL, much of the burden of setting features is borne by single form learning. Output drivenness makes single form learning very efficient, requiring only a single evaluation per unset feature, despite the exponential growth in the number of local lexica. Much less of the actual processing appears to be done by contrast pair processing, but it is an essential part of the ODL. An informative contrast pair will have at least one alternating unset feature, and the effort to set any (non-alternating) feature must evaluate over all of the possible combinations of values for the alternating unset features. The number of such combinations is necessarily smaller than the number of possible local lexica for even one of the words in the contrast pair, as the set of features shared between the words will be a subset of all of the features for either word. Nevertheless, there is potential for exponential growth in the amount of computational effort required to process a contrast pair. It is unknown to what extent the processing of contrast pairs in larger linguistic systems could become computationally expensive as a matter of practice, and if so, what further strategies could be taken by the learner to speed up the processing. As it is, the potential for such growth is limited to features in the shared morphemes of plausibly informative contrast pairs that alternate between the two words and cannot be set by single form learning.
The selection of plausible contrast pairs is another issue about which more could be learned. The number of plausible contrast pairs will generally be much smaller than the number of pairs of words on a given set of stored words. One of the words must fail initial word evaluation. To be worth considering, a pair of words must share at least one morpheme. Among such pairs, the plausible pairs are further restricted by the requirement that an unset feature in one of the shared morphemes alternate between the two words. The current implementation of the ODL requires that the words of a contrast pair share all but one morpheme of each word, but in a system where every word is bimorphemic, that requirement is redundant with the joint requirements that the words share a morpheme and be non-identical. It remains to be seen if there exist systems with words containing more than two morphemes such that contrast pairs between words differing in more than one morpheme are essential to success in learning.

The most fundamental issues for the ODL concern the overall adequacy of contrast pairs and Fewest Set Features for successful learning. In the small-scale linguistic systems that have been examined to date, contrast pairs have been adequate for all cases except for those involving paradigmatic subsets. Because paradigmatic subsets involve multiple grammars of differing restrictiveness that are equally consistent with the data, expanding contrast pairs to contrast triples, or any larger subset of data, will do no good; the feature that requires setting has both of its possible values consistent with all of the data. It is not currently known if there are any cases where contrast pairs fail to make successful learning possible for reasons other than paradigmatic subsets, and if so, if larger contrast sets would be of any benefit. It would be a striking result if it could be shown that any language successfully learnable through inconsistency detection of the sort embodied in the ODL (without recourse to Fewest Set Features) could be done so on the basis of processing at most two words simultaneously. The limitations on the present understanding of Fewest Set Features were presented in Section 8.4.2; most notable are the decision of when to invoke Fewest Set Features (when to conclude that a paradigmatic subset relation is involved), and which word (or words) to apply it to.

8.6 The map

The paradigmatic subsets in the Stress/Length system do not derive from any complex or esoteric constraints in the system. The two input-referring constraints are both standard IDENT[F] constraints. The other four constraints...
Paradigmatic subsets are quite familiar markedness constraints. This at least suggests that paradigmatic subsets are likely to be a general issue for the learning of non-trivial phonological systems.

The phenomenon of paradigmatic subsets is not something you can find evidence for in any particular language. It isn’t a property of a language; it is a relation between languages. Like phonotactic subsets, paradigmatic subsets involve matters of relative restrictiveness. With phonotactic subset relations, the observed output forms are consistent with more than one language, and absent access to paradigmatic information, the learner needs to be able to evaluate the relative restrictiveness of the hypotheses apart from their performance on the observed data. With paradigmatic subset relations, not only are the observed output forms consistent with more than one language, but the paradigmatic relationships (including phonological alternations) are consistent with more than one language. Thus, the learner needs to be able to evaluate the relative restrictiveness of the hypotheses apart from both the set of observed surface forms and the paradigmatic relations between them.

The learning solution proposed here, Fewest Set Features, uses the structure imposed on the space of inputs by output drivenness to estimate the relative restrictiveness of related sets of lexical feature commitments. This is the same structure that was used to accelerate the learning of underlying features via inconsistency detection in Chapter 7. The structure of output-driven maps has significant contributions to make to learning involving underlying forms. It plays an important role in both the use of inconsistency detection and in the enforcement of restrictiveness.

Paradigmatic subsets pose a challenge for learning because the grammars in question cannot be distinguished on the basis of inconsistency detection alone. This highlights an important point: learning is fundamentally about distinguishing possibilities. The ease or difficulty of a learning problem resides in the complexity of the relations between the possibilities, not the internal descriptive complexity of the individual possibilities. The ease or difficulty of learning in a particular phonological theory depends upon the structure (or lack thereof) of the space of possible grammars predicted by the theory.

In other words, linguistic theory and language learning are fundamentally interrelated. Work in each has the potential to significantly inform the other. Chapter 9 discusses several significant issues that arise in the present work on output-driven maps, issues in which the concerns of phonological theory and learning interact substantially.
9  Linguistic theory and language learnability

The concept of output drivenness does crucial work in the theory of phonological learning presented in this book. In fact, output drivenness originated in a search for structure in the input that would support efficient learning of underlying forms. Ideally, work in linguistic theory informs work on language learning, and work on language learning informs work in linguistic theory. This chapter provides further discussion of some key issues where linguistic theory (output-driven maps) and learnability (the Output-Driven Learner) fruitfully interact.

9.1  Contrast and the final lexicon

The ODL embodies a learning strategy in which underlying features are only set when some value for the feature is necessary, that is, when setting the feature to some other value would result in an incorrect output for some input containing the relevant morpheme. If the value of a particular feature for a particular morpheme never matters, then that feature will never be set underlyingly. In a sense, one could label a feature that is never set as non-contrastive: it doesn’t play a role in distinguishing the phonological identity of the morpheme from other possible morphemes. If no further mechanism is added to the learner, then the learner predicts in many cases an adult lexicon with some features left unset.

The learned grammar for L20, given in Section 7.9.4.4, is a useful example. The learned lexicon, (7.73), has two unset features: the length features for suffixes s1 and s2. These two features are non-contrastive, in the sense of “contrastive for a morpheme,” as defined in Section 7.1.2. Recall the full paradigm for L20, as listed in (5.2), repeated below.
Linguistic theory and language learnability

(5.2) Language L20

<table>
<thead>
<tr>
<th>r1 = /pa/</th>
<th>r2 = /pa:/</th>
<th>r3 = /pá/</th>
<th>r4 = /pá:/</th>
</tr>
</thead>
<tbody>
<tr>
<td>páka</td>
<td>páka</td>
<td>páka</td>
<td>pá:ka</td>
</tr>
<tr>
<td>s1 = /-ka/</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>páka</td>
<td>pá:ka</td>
<td>páka</td>
<td>pá:ka</td>
</tr>
<tr>
<td>s2 = /-ka:/</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká</td>
<td>paká</td>
<td>páka</td>
<td>pá:ka</td>
</tr>
<tr>
<td>s3 = /-ká/</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>paká:</td>
<td>paká:</td>
<td>páka</td>
<td>pá:ka</td>
</tr>
<tr>
<td>s4 = /-ká:/</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Observe that suffixes s1 and s2 are phonologically indistinguishable on the surface: they surface identically in every morphological context, and the full words in which they appear surface identically for each context. The suffixes differ underlyingly only in the value of the length feature. The length feature is not contrastive for suffixes s1 and s2. Length is contrastive for s3 and s4, as evidenced by the context of root r1: r1s3 surfaces as [paká] (with a short final vowel), while r1s4 surfaces as [paká:] (with a long final vowel). The difference between the two sets of suffixes lies in the underlying value of the stress feature. Because the default stress position is initial, suffixes can only be stressed on the surface if they are +stress underlyingly. Suffixes s1 and s2 are –stress underlyingly, so they will never surface as stressed. Because long vowels are shortened in unstressed position, the surface length of an underlyingly –stress suffix is predictably short: if underlyingly long, it will be shortened, and if underlyingly short, it will remain short. In L20, there are only three phonologically distinguishable suffixes: –stress, +stress with –long, and +stress with +long.

The contingency of contrast in length in this example cannot be reduced to a matter of “phonemic inventory.” It is not the case that length is never contrastive for vowels that are underlyingly –stress. Roots r1 and r2 are both underlyingly –stress, yet they are distinguished by their underlying length specification: r1s1 has a different output from r2s1. Length is contrastive in roots that are specified –stress in the environment of a suffix that is underlyingly –stress. The underlying specifications [–stress, –long] and [–stress, +long] are contrastive in roots, but not in suffixes, meaning that they contrast in some environment in roots, but they do not contrast in any environment in suffixes. The distinction arises from the fact that suffixes never appear word-initially.

The tokenness of feature contrast cannot be reduced to a matter of reference to distinct morphological categories, or even to specific morphemes. Consider a language in which stress is predictably word-initial, vowels are shortened in unstressed position, and roots can be multisyllabic, so that a root could contain
multiple vowels always surfacing in distinct syllables. In a two-syllable root, the first vowel would be contrastive for length, but the second vowel wouldn’t be, because it would never be word-initial, and thus never surface as stressed.1

The idea of not listing the values of non-contrastive features in inputs is certainly not new. Pre-generative structuralist analyses often characterized phonemes in terms of “distinctive” features. In early generative phonology, Chomsky and Halle were explicit: “The lexical entry . . . must contain just enough information for the . . . phonology to determine its phonetic form in each context” (Chomsky and Halle 1968: 12). Subsequent theoretical developments reduced the importance of not explicitly listing predictable feature values. In Optimality Theory, the principle of Richness of the Base requires that the theoretical inventory of underlying representations be universal, obliging the phonological map to account for all possible fully determined input representations, further diminishing if not eliminating the theoretical significance of the listing of non-contrastive features.

The issue of not setting non-contrastive features raised here should not be confused with underspecification, in particular with temporary (input-only) underspecification (Archangeli 1984, 1988, Kiparsky 1982, Steriade 1987, as cited in Steriade 1995). An unset feature is a construct of the learner, not the linguistic theory itself. A non-contrastive feature could be set to any value without effect. If a learner feels compelled to assign a value to a non-contrastive unset feature, it doesn’t matter which value the learner selects. By contrast, an underspecified feature is by design such that assigning a value to the feature might well make a difference. Such features, when underspecified, are contrastive. The point of temporary underspecification in linguistic theory is precisely to have phonological effects.2 In such a theory, part of the learner’s job is to determine which feature instances must specifically be made underspecified.

Privativity, or “permanent” underspecification (Archangeli 1988, Steriade 1987, as cited in Steriade 1995), is also not the issue. What matters for present purposes is how faithfulness constraints assess features in candidates. Conceptualizing a feature as “present” in one instance and “absent” in the other makes no difference for the current discussion so long as relevant faithfulness constraints actively recognize the difference between presence and absence. The “absence” of a privative feature is effectively the second value of the feature. One would expect that a standard IDENT[F] constraint for a privative feature

---

1 Similar cases involve languages with syllable-final or word-final obstruent devoicing, and contrastive obstruent voicing in other obstruent-friendly environments.
2 It has been suggested that Optimality Theory reduces or eliminates the need for such underspecification (Itô et al. 1995, Smolensky 1993).
would incur a violation when the input correspondent lacks the privative feature while the output correspondent has it, as well as when the input correspondent has the feature and the output correspondent lacks it. A learner would then distinguish between an unset feature and a feature that has been set to the value “absent.”

Accepting that in Optimality Theory there is no necessity to leaving any underlying feature values unset, the question then arises as to whether it is necessary to set all features. This is a potentially interesting question. Because unset features are purely a construct of the learner, they have no explicit status in conventional Optimality Theory. But if, as a matter of psychological reality, native speakers (mature learners) can have underlying forms with unset features, then their language processing mechanisms must be able to deal with them properly. If we charge grammar with the responsibility for interpreting underlying forms with unset features, it isn’t immediately clear how Ident[F] constraints should evaluate IO correspondents where the input correspondent has the relevant feature unset.

For mature grammars, it seems intuitive to expect that an input with unset non-contrastive features should map to the same output as it would if those features were set. For basic, symmetric Ident constraints, given that any underlying value will yield the same result for a non-contrastive feature, and given that the (faithful) surface value of the feature is one of the possible values, it might be workable to grant unset underlying features as vacuously satisfying Ident. The situation will be more complicated for unset contrastive features. If a mature speaker has been exposed to data such that every morpheme stored in the lexicon has been robustly represented in every relevant phonological context, then we would expect that every contrastive feature would have been set. But such “closure” among observed morphemes is hardly guaranteed. If a morpheme is observed in a context where a feature is neutralized, but not in a possible context where the feature would be contrastive, and the speaker is

---

3 One could, of course, define a value-restricted Ident constraint that could only be violated if the input correspondent has a specific value, such as the constraints discussed in Section 3.4.4. But, as those examples clearly indicate, they are not in any way dependent on the excluded feature value being ‘absent’.

4 It may be worth emphasizing that, to be workable, a mature grammar should unambiguously determine what is optimal for an input. A strategy such as assigning, to unset features, the values of their surface realizations is a non-starter, because it presumes that the learner already knows what the output is. That strategy is OK when a learner is dealing with learning data, and does know the output, but is not workable for language use.
called upon to produce a word with the morpheme in such a contrastive context, the feature would be unset, and the speaker would need to handle the unset feature.

An example of this sort that has been studied is voicing alternation in Dutch (for a recent overview, see Kerkhoff 2007). The word [bet] “bed” is analyzed with the final consonant underlyingly voiced, /bɛd/, due to the surface form of related words like [bedən] “beds.” However, if a language user has only observed the monomorphemic [bet], with the final voicing neutralized, and is called upon to produce the plural, something further must be called upon to determine what to produce. A number of hypotheses can be imagined for this situation. The core theory itself could be extended to define outputs for inputs with unset features, perhaps by defining faithfulness constraints in the way suggested above. Taking inspiration from analogy-based theories, including the work of Bybee (1985, 1998, 2001), a speaker might look to similar words stored in the lexicon and construct correspondences between the inputs in order to infer values for the unset input features. Taking inspiration from ideas on paradigm uniformity (Benua 1997, Kenstowicz 1996), the learner might assign values to unset features based on the surface realizations of those features in morphologically related words, even though those surface realizations result from neutralization.

A number of psycholinguistic studies have been conducted that are relevant here, including studies on voicing in Dutch specifically (Ernestus and Baayen 2003). Also relevant is the voluminous literature on phonological acquisition, including similar Wug-style tests on Dutch children (Kerkhoff 2007). There are a number of complex issues involved in interpreting the results of such studies. For example, both the design of and the discussion of different studies can involve differing assumptions about which features are or are not expected to be unset/underspecified/not present in the inputs. I will not discuss this work in greater detail here.

The alternative, in which all underlying features are obligatorily set by the end of learning, has been proposed, most notably in the form of the principle of lexicon optimization (Prince and Smolensky 1993/2004: 225–231). Initially formulated in terms of choosing an input for a single form, lexicon optimization proposes choosing, from among the inputs that the grammar maps to the desired output, the one giving rise to the most harmonic candidate, comparing harmony across the (phonetically identical) optimal candidates for the relevant

5 Note that this suggestion is different from the standard analogy-based proposals, in which the output for a novel word is determined via analogy with the outputs of similar words.
inputs.\(^6\) Note that this presumes a choice from among theoretically proper (fully determined) inputs.

It is conceivable that lexicon optimization could be extended so that inputs containing unset features were included among the options. This would again require that the harmonic evaluation of candidates with unset underlying features be spelled out, so that one could determine which such inputs map to the desired output. It isn’t entirely clear what the point of such an extension would be, however. If a properly functioning harmonic evaluation of candidates with unset underlying features is given, then the basic concern raised above about underlying forms with unset underlying features has already been resolved; there is then no apparent remaining problem for lexicon optimization to solve.

### 9.2 Forms of restrictiveness enforcement

The ODL uses two different mechanisms for explicitly enforcing restrictiveness. One is the faithfulness-low ranking bias of BCD, a bias on rankings, and the other is the Fewest Set Features procedure, reflecting a bias on the lexicon of underlying forms. These two mechanisms function differently during learning, but toward the same goal, that of finding the most restrictive grammar consistent with the data.

The learner’s permanent accumulation of ranking information is the support, the list of winner–loser pairs from which rankings are generated. Inconsistency detection is based on the winner–loser pairs, not the particular hierarchy generated by BCD; inconsistency detection does not take into account the restrictiveness biases of BCD. The learner’s permanent accumulation of lexical information is the set features of the lexicon. Feature setting via inconsistency detection is not directly subject to restrictiveness biases. The distinction between set and unset features is useful in allowing the learner to keep track of the lexical values it has committed to, a function not immediately concerned with restrictiveness. The accumulation of winner–loser pairs and the setting of features based on inconsistency detection restrict the learner’s grammar hypotheses based on consistency with observed data, but do not directly serve the preference for the most restrictive hypothesis consistent with the data.

---

The bias towards setting features only when necessary implicitly serves to enforce restrictiveness when unset features are interpreted as features in which any of the possible values should work. The a priori “most restrictive” lexicon is one in which no features are set, implicitly expressing the hypothesis that all inputs map to the same output (setting aside any non-featural elements of contrast, like varying input length in a system with no insertion or deletion). The learner determines where restrictiveness must yield as it sets underlying features. While not an explicit mechanism for restrictiveness enforcement, the “set only when necessary” core of the learner does conveniently support the enforcement of restrictiveness via the lexicon.

The main effect of BCD’s explicit faithfulness low bias lies in the hierarchy that the learner might actually use at any given point, including the hierarchy that the learner ultimately ends up with (when the learner is no longer altering their grammar). Restrictive ranking relations that are not explicitly indicated by alternations (and the disparities they impose) are imposed on the hierarchy by BCD. In learning, the faithfulness-low bias impacts error detection (phonotactic learning) and initial word evaluation (paradigmatic learning), which is performed on the basis of the hierarchy generated by BCD. This has its greatest impact in phonotactic learning. Because phonotactic learning works exclusively from identity candidates for observed forms, a learner which only used a ranking with faithfulness constraints ranked as high as possible would not learn much if anything from phonotactics alone. By persistently using the faithfulness-low bias of BCD, the phonotactic learner starts out presuming the most restrictive ranking and then accumulates information (in the form of winner–loser pairs) about the circumstances in which restrictiveness must yield.

Restrictiveness concerns also are explicitly addressed when the learner must go beyond inconsistency detection in setting underlying features. Fewest Set Features sets as few features as possible in order to account for a word. Minimizing the number of set features means maximizing the number of underlying forms that should be equivalent for each morpheme. Fewest Set Features directly examines different inputs for a word to determine what sets of inputs can map to the correct output; the learner intends to arrive at a (minimal) collection of set features so that any input consistent with the lexical entries yields the correct output. Underlying feature setting based on inconsistency only sets a feature if it can determine that only one value for a given feature can allow the correct output to be generated. When the learner is unable to make further progress based on inconsistency detection alone, it concludes that the problem
must be the existence of paradigmatic subset relations, and at that point uses Fewest Set Features.

9.3 Evaluation metrics

Each restrictiveness bias (on the ranking and on the lexicon) is something like an evaluation metric on grammars (Chomsky 1965, Chomsky and Halle 1968). A grammar with a higher r-measure is assigned a higher, or preferred, value, because it is predicted to be more restrictive. Similarly, a grammar with fewer set features for the same morphemes is assigned a higher value, because it is predicted to be more restrictive. The purpose of an evaluation metric is to compare, with respect to a given linguistic theory, different grammars that are consistent with observed data, and measure the (relative) degree to which the different grammars capture linguistically significant generalizations. Linguistic generalizations typically result in neutralization, with distinct inputs mapping to the same output in order to satisfy the generalizations. More neutralization, resulting from a more restrictive grammar, is indicative of stronger linguistic generalization.

Capturing generalizations is frequently associated with conceptions of descriptive brevity. For example, Chomsky and Halle (1968: 334–335) proposed an evaluation metric that was the reciprocal of the number of symbols needed to state the grammar in a particular form: the fewer the required number of symbols, the higher the assigned value. They are quite explicit about the connection between capturing generalizations and their evaluation metric (Chomsky and Halle 1968: 335): “The only claim being made here is the purely empirical one that under certain well-defined notational transformations, the number of symbols in a rule is inversely related to the degree of linguistically significant generalization achieved in the rule.”

Evaluation with respect to description length has limited application to learning in Optimality Theory. The set of constraints is universal, and there is no

7 As emphasized by Chomsky (1965: 38–39), this is a very different matter from the comparison of different linguistic theories.

8 What Chomsky and Halle actually stated was an evaluation metric based on the length of the expression of the rules. As pointed out by Prince (2007: 37–38), limiting the evaluation metric to the rules alone is problematic: having no rules, and listing everything in the lexicon, gets the highest evaluation, because that evaluation metric does not take into account the size of the information stored in the lexicon; see also Ristad 1990. Prince suggests that an actual lexicon + rule metric is implicit, if unstated, in SPE. Given a lexicon + rule metric, a rule captures enough of a linguistic generalization to be worth considering if its inclusion in the grammar reduces the expression of the lexicon by more symbols than are required to express the rule itself.
obvious sense in which one ranking is shorter than another. The set of features is also taken to be universal, so grammars will not differ in the number of features. The idea of limiting the number of features that are actually set bears some resemblance to older notions of minimizing the number of distinctive features (see Section 9.1), but is really something else entirely: the decision of whether to set a feature is made separately for each individual instance of a feature (token), not for all occurrences of a feature at once (type). In the current proposal unset features aren’t really absent from representations, they are indicated as unset. Given the principle of Richness of the Base, generalizations result in neutralization: inputs are not prevented from being inputs, they are neutralized with other inputs in the outputs. Generalizations restrict the space of optimal outputs for a grammar, and with a fixed set of inputs, reducing the number of outputs means more inputs per output. There is no obvious sense in which an OT grammar with a greater degree of neutralization has a shorter description than an OT grammar with a lesser degree of neutralization.

It should be emphasized that what is being discussed above is the comparison of different analyses within a given Optimality Theoretic system, with fixed sets of constraints and candidates. That is very different from the comparison of different linguistic theories, including different Optimality Theoretic systems, which can differ in the nature of the candidates and in the content of constraints. All metrics of theory comparison are potentially relevant to those sorts of comparisons (simplicity, minimum description, etc.). The linguist is trying to figure out what the constraints and candidates are, as well as the correct rankings and structural analyses for different languages. The learner, by contrast, knows what the constraints are and what the possible candidates are and is more narrowly trying to determine the highest-valued ranking and lexicon, given a set of learning data.

Just as the faithfulness-low ranking bias elevates faithfulness constraints in the ranking only when needed to account for phenomena that are less than fully restrictive, the bias against set features only sets features to specific values when needed to account for phenomena that are less than fully restrictive. In order for a contrast between morphemes to surface in a language, there must be an explicit distinction between the underlying forms of the two morphemes, and there must be at least one constraint active in the ranking that is sensitive to that difference between the underlying forms (these conditions are necessary, not sufficient). The two things must work together. It is not surprising, then, that when viewed as evaluation metrics, maximizing the r-measure and minimizing the number of set features appear to be somewhat redundant. Setting underlying features only reduces restrictiveness if there are faithfulness constraints sensitive to the
Linguistic theory and language learnability

feature values ranked high enough to make a difference. The learner starts out at one extreme for each measure: all markedness constraints dominating all faithfulness constraints, and no features set. Learning generally moves each measure away from its extreme over time in response to data, elevating the ranking of some faithfulness constraints and setting some underlying features. The learner is generally predicted to progress from more restrictive to less restrictive hypotheses as data accumulate.

For further recent discussion of evaluation metrics in phonology, see Prince 2007.

9.4 What is stored by the learner (and why)

9.4.1 Ranking information

The ODL stores ranking information in the form of winner–loser pairs. Each pair is an inference about the ranking, in the form of an ERC. The learner’s support (their list of winner–loser pairs) implicitly represents a space of possible rankings, those rankings that are consistent with all of the winner–loser pairs in the support. A support is a very compact representation of information about a ranking, yet it can be used to reason with quite efficiently. Consistency with a list can be determined via MRCD. Different hypothesis constraint hierarchies consistent with a support can be generated efficiently, reflecting various biases: faithfulness low, markedness low, all constraints as high as possible. Information from different data forms are easily combined in this approach, as new winner–loser pairs are simply added to the support. The algorithms for working with a support treat a list of pairs from several different data forms in the same way as a list of pairs all from a single form.

Output drivenness greatly assists the learning of ranking information, because it allows the learner to make valid ranking inferences from data without needing to have fully determined the inputs for the data. By assigning values to all unset features that match their surface realizations within a winner, the learner can obtain information about the ranking from a datum involving unset features, despite the fact that winner–loser pairs require fully determined inputs. This is crucial to the ODL’s ability to contend with the mutual dependence of ranking information and lexical information. The ODL can obtain and store ranking information based on whatever lexical information it has (even no lexical information at all, in phonotactic learning), and it can use that stored information to reason about the values of unset features.

The ODL proceeds by accumulating ranking information over time. The learner responds to informative data not by directly altering an old constraint
hierarchy to get a new one, but by adding one or more winner–loser pairs to an existing support. By doing this, the learner is able to distinguish commitments made in response to data from bias-determined defaults, so that inconsistency detection is performed only with respect to the former, not the latter. The learner has accumulated sufficient ranking information when it is able to resolve all constraint conflicts that arise in the grammar. The learner does not necessarily pursue a total ranking: if there are constraints that do not conflict with each other, the learner will not necessarily commit to a ranking relation between them. The learner follows the lead of the linguistic theory, and its relation to the available data.

9.4.2 Lexical information

The ODL stores lexical underlying form information by representing an underlying form as a sequence of segments, each with a fixed inventory of features whose values are initially unset. The feature instances can be set independently by the learner. In general, a learner will possess a set of underlying forms in which some features have been set and others have not. Such a set implicitly represents a space of possible lexica, those that are consistent with the set features. Such a representation is very compact, yet it can be used to reason with efficiently. Individual features can be set via inconsistency detection with MRCD. Information from different forms are easily combined in this approach, as different features of an underlying form can be set on the basis of different words in which that morpheme appears.

Output drivenness greatly assists the learning of underlying form information, because it allows the learner to make valid inferences from data without needing to have fully determined the ranking or the underlying forms of the other morphemes. Output drivenness allows a kind of conditional independence between features: a feature can be tested and possibly set on its own, but not at the cost of prohibiting features from interacting with each other in the linguistic theory. The ability to independently set features in underlying forms is crucial to the ODL’s ability to contend with mutual dependence, not only between ranking information and lexical information, but between the underlying forms of different morphemes. The ODL can set a feature in an underlying form on the basis of one word (in combination with some other morphemes), and then consider a different word in which the newly set feature surfaces unfaithfully, both to set a different feature and to obtain non-phonotactic ranking information.

The ODL accumulates lexical information over time. The learner responds to informative data not by exchanging one fully determined hypothesis for another, but by setting additional features. This is true both at the level of the lexicon
and at the level of the individual underlying form. This allows the learner to distinguish lexical commitments made in response to data from default feature values, and the learner is able to selectively enforce consistency with only the set features when reasoning with respect to its lexical representations. The learner has accumulated sufficient lexical information when its underlying forms adequately account for the observed phonological behavior of all of the observed morphemes. The learner does not necessarily pursue a totally determined lexicon (with all features set): if there is a feature that is not contrastive in any observed environment, then the learner will not commit to a value for that feature. The learner follows the lead of the linguistic theory, and its relation to the available data.

9.4.3 Structural ambiguity and multiple hypotheses
One issue not addressed elsewhere in this book is structural ambiguity, which concerns the distinction between an output and an overt form, the portion of an output that is directly audible to the learner. An overt form exhibits structural ambiguity if it is consistent with more than one output. Each output consistent with an overt form is commonly called an interpretation of that overt form. Structural ambiguity in phonology commonly involves prosodic constituent structure. To use one classic example, consider the overt form for a three-syllable word with main stress on the middle syllable. Even if we assume that, for purposes of discussion, the analysis into syllables is unambiguous (only one syllabification of the overt form is viable), the overt form is still ambiguous between at least two metrical foot analyses (and possibly more, depending on one’s metrical theory): one which groups the first two syllables into an iambic foot, and one which groups the last two syllables into a trochaic foot. The foot boundaries themselves are not directly audible, and thus not part of the overt form itself. The foot structure is, however, part of an output, and the two foot structures correspond to two interpretations of that overt form: two different outputs with identical overt forms.

Structural ambiguity is a significant issue for a language learner, because the different outputs typically make different (and often conflicting) demands on the grammar. The information necessary to resolve structural ambiguity in an overt form (determining which of the interpretations is the correct one for an ambiguous overt form) lies in its relations to other data, that is, other overt forms, which may be structurally ambiguous as well. The learner’s task is to find a grammar which can simultaneously sustain, for each overt form, one of its interpretations.
Structural ambiguity in phonological learning has been the subject of several lines of research (Apoussidou and Boersma 2003, Dresher 1999, Dresher and Kaye 1990, Eisner 2000, Pulleyblank and Turkel 1998, Tesar 1998a, Tesar 2004, Tesar and Smolensky 2000). More recent work has begun to examine the simultaneous acquisition of ambiguousmetrical structure and underlying forms (Apoussidou 2007). Of particular interest here is work by Akers (Akers 2012), because it extends the ODL to simultaneously deal with ambiguousmetrical structure and underlying forms.

Akers’ approach to structural ambiguity builds on the Inconsistency Detection Learner, or IDL (Tesar 2004). The IDL contends with an ambiguous overt form by constructing, in parallel, all of the possible interpretations of that overt form and separately checking each one for consistency with any previously acquired knowledge of the grammar. If the learner has not yet accumulated sufficient information about the grammar to eliminate all but one interpretation, then the learner retains each consistent interpretation as a separate hypothesis, with the intent of eliminating the incorrect one(s) later, once the learner has obtained further grammatical information. When the learner possesses more than one hypothesis during learning, it separately confronts new overt forms with each hypothesis, both to test each hypothesis for consistency with the new overt form, and to gain further information for the hypothesis in the event of consistency. This is a different approach to “partial” knowledge from that employed by the ODL. The ODL breaks knowledge of the ranking into pieces (the winner–loser pairs), and breaks knowledge of the lexicon into pieces (values for individual feature instances), retaining at any given time only those pieces that it is certain of. The IDL instead works with entire interpretations and retains multiple competing hypotheses while awaiting further knowledge, rather than trying to decompose interpretations into pieces. In combining the two, Akers uses the “pieces” approach to storing ranking information and lexical information, and uses the “parallel hypotheses” approach to storing information about interpretations of overt forms.

It is instructive to consider what could motivate Akers’ approach. The hidden structure in the output, such as prosodic structure, is different from the hidden structure in underlying forms and rankings in several respects. It isn’t obvious, in current prosodic theory, what “pieces” would be appropriate so that a learner could commit to one piece prior to certainty about the others. Different interpretations of the same overt form typically put feet in different places, and feet typically cannot freely combine: where one foot goes affects where other feet could possibly go, as a matter of what GEN itself permits. Furthermore, if a learner were to store a piece of an interpretation in some respect, it isn’t
obvious how the learner would use it to, for example, obtain ranking information; that would require identifying some appropriate competitor interpretation piece such that one piece should be more harmonic than the other, independent of the other parts of the output not yet determined.

Ranking information lends itself well to the “pieces” approach, provided the “pieces” are understood to be winner–loser comparisons involving entire outputs. Winner–loser comparisons are inherent to the theory. A learner can make use of such partial ranking knowledge by generating a constraint hierarchy consistent with that knowledge, via RCD or a similar procedure. Underlying forms also lend themselves well to the “pieces” approach, because they are easily decomposed into feature instances, which can independently be set to values. A learner can make use of partial underlying form knowledge by generating fully determined underlying forms consistent with the entailment relations of output drivenness (assigning unset features values that match their surface realizations for a given word). For information about both the ranking and the underlying forms, linguistic theory makes it possible for a learner to store partial information (winner–loser pairs, underlying forms with only some features set), and to “project” partial information into hypothesized complete structures (constraint hierarchies, fully determined underlying forms) that can be profitably used to reason further about new data.

To say that “it isn’t obvious” how to take a pieces approach to hidden output structure is not to say that it is impossible. A clever scheme for doing so with current prosodic representations may be proposed, or a new prosodic theory may come along that better lends itself to such decomposition for the purposes of learning. However, the present lack of such a scheme or theory is suggestive, especially in light of the fact that prosodic structure is not commonly thought to be stored in an adult grammar, whereas ranking information and lexical information must be. Prosodic structure, and hidden structure in the output more generally, are usually taken to be transient, constructed on the fly for purposes of mediating between overt data and principles of grammars, but not permanently stored in the lexicon, preserved by faithfulness constraints or otherwise referred to in the input.

9.5 Beyond output drivenness in learning

Output drivenness appears to hold for much of basic phonology. But there are a number of phenomena for which the standard analyses result in non-output-driven maps, such as synchronic chain shifts (see Chapter 4). Short of reanalyzing all such cases in purely output-driven terms, how such phenomena
could be accommodated in learning will be highly dependent on how one chooses to account for them within core linguistic theory.

One possibility would be to develop a theory of phonological maps that is less restrictive than purely output-driven maps, but retains restrictive properties that could be exploited for learning along the same lines as is done with output-driven maps. This would be a plausible approach if non-output-driven maps are dealt with by using constraints that can introduce non-output-driven effects into the maps defined by Optimality Theoretic systems, such as the kinds of constraints discussed in Chapter 4. Accommodating this in learning might involve some more relaxed conditions on the input-referring constraints, such that they exhibit certain non-ODP behaviors but not others, so that some kind of exploitable structure on the space of inputs is still entailed. Future pursuit of such conditions would ideally be informed both by properties of learning and by linguistic typology: restrictions on constraints that simultaneously allowed efficient learning of underlying forms (despite permitting some non-output-driven patterns) and made empirically supported predictions about the types of non-output-driven patterns that are possible.

Another possibility would account for non-output-driven phenomena by analyzing languages as the composition of several output-driven but non-identical maps. As discussed in Section 4.12, the composition of two non-identical output-driven maps is possibly not output-driven. In Stratal Optimality Theory (Kiparsky 2003), the output is derived from the input by a sequence of strata, where each stratum is an OT grammar with a distinct ranking. Adapting a line of thinking suggested by Bermudez-Otero (2003), one could propose that each of the component OT grammars is output-driven, with non-output-driven effects resulting solely from interaction between the component grammars. Learning in such a theory could depend to a great extent on the proposals made here for learning the individual component grammars and would require further principles for working out the relations between the different strata as part of the process of learning them.

### 9.6 What has been accomplished

Phonologically interesting properties can be found at the level of the map. Map properties depend upon commitments about the representational structure of inputs, outputs, and the correspondences between them. However, map properties stand apart from any particular theory of how outputs are derived from inputs, providing independent landmarks for evaluating both data and theories. This book has proposed one such property, output drivenness, and argued that
it captures familiar intuitions about surface orientedness in phonological maps. Output drivenness makes it possible to show that chain shifts and derived environment effects are variations on the same basic phenomenon, at the level of the map. It is likely that other linguistically significant properties of maps wait to be discovered, perhaps defining less restrictive classes of maps.

From the definitions of output drivenness and of Optimality Theory itself, sufficient conditions for an Optimality Theoretic system to define only output-driven maps can be derived. The key conditions are that Gen must be correspondence uniform and that all of the constraints must be output-driven preserving. This also determines necessary conditions for getting non-output-driven behavior from OT systems. If Gen is correspondence uniform, then a system must include at least one non-ODP constraint in order to define a non-output-driven map. The definition of non-ODP constraint behavior unifies the understanding of a number of different proposals in OT, including proposals as disparate as local conjunction and sympathy theory.

Output drivenness imposes structure on the space of inputs. In Optimality Theory, due in part to the principle of Richness of the Base, that input structure proves especially useful in learning. Output drivenness figures very prominently in effective approaches to the learning of underlying forms, the simultaneous learning of underlying forms and constraint ranking information, and the enforcement of restrictiveness in the face of paradigmatic subset relations. Output drivenness may ultimately be superseded by other properties that better characterize phonological systems, but those other properties will need to impose some kind of analogous structure on the input if a plausible account of language learning is to be maintained.

A better understanding of the properties of phonological maps, and their implications for specific linguistic theories, is beneficial to an evaluation of the relative strengths of competing theories, constitutes linguistic insight on its own, and is indispensable to theories of language learning.
References


References


References


1993b. Prosodic morphology I: Constraint interaction and satisfaction. Ms., Linguistics Department, University of Massachusetts, Amherst, and Linguistics Department, Rutgers University, New Brunswick, NJ.


2006. Implication and impossibility in grammatical systems: What it is and how to find it. Ms., Linguistics Department, Rutgers University. ROA-880.


1995. On the internal structure of the constraint component Con of UG. Ms., The Johns Hopkins University. ROA-86.


acquisition, 174, 210, 213, 254–255, 389, see also learning from positive evidence, 207–208
Akan, 56
Akers, Crystal, 396–398
akx (candidate), 44, 45
Albright, Adam, 248
Alderete, John, 125, 129, 259
alternation, 175, 211, 298, 305, 308–310, 317, 342, 356, 389, 391, see also morphemic alternation
Ambrazas, Vytautas, 39
Anderson, John M., 282
Anderson, Stephen R., 155
Angluin, Dana, 208
antifaithfulness and the rotation theorem, 165 conditioned, 145–148 conjoined with faithfulness, 158 input-output, 125 transderivational, 125 Anttila, Arto, 194–195 aoy (candidate), 70, 74–81 Apoussidou, Diana, 251, 397 Archangeli, Diana, 145, 387 assigning a feature value, 252, 260, 275, see also setting a feature autosegmental representation, 30, 60–61 Axininca, 56
Baayen, Harald, 389
Baker, C. Lee, 208
Baković, Eric, 39–41 Balari, Sergio, 145 basic alternant, 248–250 BCD. See Biased Constraint Demotion Becker, Michael, 258 Beckman, Jill, 130, 132
Benua, Laura, 88, 125, 149, 389
Bermudez-Otero, Ricardo, 170, 171, 399
Bernhardt, Barbara, 214
Biased Constraint Demotion, 213–221, 222, 225, 258, 330, 335, 358, 375, 382, 390–392
bmx (candidate), 44, 45
Boersma, Paul, 188, 195, 251, 397
bpy (candidate), 69, 70
Brasoveanu, Adrian, 69, 258 breaking (multiple correspondence), 60, 86
Broihier, Kevin, 187
Brown, R., 207
Browne, W., 155
Bybee, Joan L., 389
Camilli, A., 43 candidate, 20–21, 26, 29, 34, 35, 37, 44, 46, 64, see also akx; aoy; bmx; bpy
Catalan, 132
Cherry, Colin E., 282
Cho, Young-mee Yu, 194
Chomsky, Noam, 1, 5, 15, 21, 155, 170, 212, 387, 392 classical OT grammar, 150, 151, 153, 156, 159, 161, 162, 165, 166
Clements, George N., 282
coalescence (multiple correspondence), 60, 86 CON, 15, 70, 71, 82, 84, 120, 158, see also constraint
Conflicts Tie, 191–194, 220, 336
constraint. See faithfulness constraint; local
conjunction; local disjunction;
markedness constraint; output-driven
behavior, 117, 118, 127, 167, 400
conservative, 152, 153, 158, 162, 165, see also grammar, conservative
input-referring, 15, 89, 90, 171, 399
material implication, 145, 148, 156
non-stationary, 125, 149
value-independent, 89
value-restricted, 90, 137, 139, 168, 388
constraint demotion, 222, 255, 256, see also
Biased Constraint Demotion;
MultiRecursive Constraint Demotion;
Recursive Constraint Demotion; Robust
Interpretive Parsing / Constraint
Demotion
constraint promotion, 222
Contenders algorithm, 195, 202–203
contrast, 19, 33, 130, 235–236, 262, 274, 279,
280–284, 357, 358, 360, 363, 385–387,
393, see also contrast pair; morphemic
contrast; phonotactic contrast
contrast pair, 274, 275, 276, 299–302, 309,
310, 315, 322, 377, 379, 381, 383
contrast pair learning, 334–335, 345, 380
Contrast Pairs and Ranking algorithm, 246,
250, 274–277, 287, 288, 298, 299, 315,
336, 360, 381
contrastive feature, 284, 291, 295, 387, 388
for a morpheme, 282, 309, 385
for an input, 281, 282, 283, 284
phonotactically, 291–294, 299, 325
contrastive hierarchy, 282
contrastive segment, 283, 284
correspondence relation
base-reduplicant (BR), 88
between candidates. See correspondence
relation, input-input (II)
between disparities, 48, 50, 51–52
input-input (II), 50, 52–54, 57, 71, 72, 73,
76, 92, 283
input-output (IO), 20, 26, 30, 44, 47, 49, 52,
54, 57, 58, 60, 71, 85, 88, 152, 177, 323:
importance of, 33–37
output-output (OO), 88, 96, 125
correspondence uniform, 84–86, 87, 127, 284,
400
CPR. See Contrast Pairs and Ranking
algorithm
Crowhurst, Megan, 123, 145
CTie. See conflicts tie
cue learning, 186, 323
Culicover, Peter, 186
Dakota, 129, 132, 136, 168
Dambriunas, Leonardas, 39
data consistency, 175, 211
de Lacy, Paul, 90, 108, 249
demuth, Katherine, 214
Dependency Phonology, 282
derived environment effect, 14, 23, 24–25,
124, 128, 136, 156, 167–169, 400
derived environment exchange, 151, 155–159,
160, 163, 165, 166
Diola, 56
disparity, 1, 21, 59, 60, 176, 210, 283, 294,
302, see also correspondence relation,
between disparities
analogous, 74, 80, 92
classes of, 47
corresponding, 74
identical corresponding, 26, 30, 36, 54, 58,
94–95, 100
inventory of, 28–30, 47
non-identical corresponding, 100–101
role in relative similarity, 26–27
distinction conflict, 118, 125–127, 146, 158,
159, 168
distinction only at greater similarity, 118,
122–125, 168
distinction only at lesser similarity, 118,
119–122, 130, 131, 133, 136, 141, 142,
143, 145, 149, 168, 169
distinctive feature. See contrastive feature
Dresher, B. Elan, 186, 280, 282, 323, 397
Dutch 389, see also Flemish Brussels Dutch
EDCD. See Error-Driven Constraint Demotion
Eisner, Jason, 397
elementary ranking condition, 65
tenentment, 66–69, 82
trivial, 83, 257, 273
environment morpheme, 309, 317, 334, 345
ERC. See elementary ranking condition
Ernestus, Mirjam, 389
error detection, 186, 187, 188, 196, 201, 231,
329, 333, 391
Index

Error-Driven Constraint Demotion, 195, 221, 255
error-driven learning, 19, 186, 195, 196, 321–328
Etxarri Basque, 4, 38, 119
evaluation metric, 392–394
everything idempotent, 150, 151, 153, 155, 166,Ewen, Colin J., 282

faithfulness. See faithfulness constraint
containment-based, 45, 120, 141
correspondence-based, 45, 88, 89, 120
theory of, 15
faithfulness constraint, 8, 11, 15, 88, 152, 154, 213, 216, 223, 227, 235, 387
positional, 130, 132
faithfulness low, 329, 330, 332, 353, 391, 393, 394
Fant, Gunnar, 281, 282
feature, see also contrastive feature
binary, 177, 248, 259, 281, 283, 285, 290
privative, 387
suprabinary, 283, 290
feature geometry, 282
Fewest Set Features, 369, 370, 375, 376, 383, 390
Fisher, R. A., 372
Flemish Brussels Dutch, 156, 161, 168
Flora, J., 249
Fromkin, Victoria, 56
fusion, 68, 227, 272

Gen, 69, 70, 84, 87, 141, 151, 158, 177, 397, see also correspondence uniform
freedom of analysis, 85
partitioned, 86, 177
GLA. See Gradual Learning Algorithm
Gnanadesikan, Amalia, 144, 214
Gold, E. Mark, 186
Goldsmith, John, 60
Government Phonology, 282
Gradual Learning Algorithm, 195, 221, 251
grammar
conservative, 152, 154, 155, 158, 159, 165, see also constraint, conservative
exact, 151, 152, 158
homogeneous, 151, 152, 158
inclusive, 152, 158
grammar space, 205, 253, 254
grammar subset, 211–213
greater internal similarity, 26, 27, 44, 45, 47, 54, 55, 69, 82, 84, 118, 285, 287
Hale, Kenneth, 2
Hale, Mark, 212–213, 248
Halle, Morris, 5, 21, 155, 170, 281, 282, 387, 392
Hanlon, C., 207
harmonic bounding, 185, 191, 192, 229
harmonic serialism, 84
Hayes, Bruce, 16, 172, 195, 210, 216
Hewitt, Mark, 123, 145
Horwood, Graham, 259
Hulst, Harry van der, 282
idempotency, 4, 13, 14, 16, 22, 23, 152, 163, 166, 170, 211, 288, see also identity map
property; eventually idempotent
identity map property, 4, see also idempotency
IDL. See Inconsistency Detection Learner
Idsardi, William, 7–8, 10, 41
Inconsistency Detection Learner, 397
initial lexicon construction, 274, 277, 278, 360
initial word evaluation, 333, 334, 335, 365, 371, 376, 383, 391
Inkelas, Sharon, 390
input-input correspondence. See correspondence relation, input-input (II)
internal similarity, 26, 27, 45, 58, 61, 159, see also greater internal similarity
Itô, Juniko, 135, 140, 170, 171, 221, 387
Jakobson, Roman, 281, 282
Jarosz, Gaja, 251, 373–375
join, 270–272, 299

Kager, Rene, 120, 259
Kaye, Jonathan, 186, 282, 323, 397
Kenstowicz, Michael, 39, 248, 249, 389
Kerkhoff, Annemarie, 389
Kiparsky, Paul, 1, 5, 10, 37, 40, 117, 170, 387, 399
Kirchner, Robert, 2, 4, 38, 43, 119–120
Kisseberth, Charles, 1, 2, 3, 39, 248, 249
Klimas, Antanas, 39
Klokeid, Terry Jack, 2
Kruskal, Joseph, 30, 34
Index 413

language space, 205, 253, 254
language subset, 211–213
Lardil, 2, 3
learning. See acquisition and restrictiveness, 207–209, 358–361, 373, 376, 382, 390–392, see also learning, phonotactic; Fewest Set Features non-phonotactic, 246, 274, 277, see also non-phonotactic ranking information of rankings. See chapters 5–7 of underlying forms. See chapters 6–8 paradigmatic, 360, 391 phonotactic, 16, 209–211, 221, 222, 224, 226, 235, 254, 323, 329, 336, 382, 391
Levelt, Clara, 214
Levenshtein distance, 30, 32
Levenshtein, V. I., 30, 32, 34
Odden, David, 40 ODL. See Output-Driven Learner ODM. See output-driven map ODP. See output-driven preserving Ohno, Kazutoshi, 145 Oostendorp, Marc van, 214 opacity, 5, 7, 14, 15, 19, 37, 41, 43, 170, see also opaque process; surface-apparent process; surface-true process opaque process, 14, 38, 39, 40 Optimality Theory opacity in, 6–13 output-driven maps and. See chapters 3, 4, 7 processes in, 6 order-preserving bijection, 33, 44, 176 OT with candidate bijection (OT-CC), 28
Output-Driven Learner, 334, 335, 359, 375, 376, 377, 382, 383, 385, 390, 394, 395, 397
output-driven map, 13–15, 20–26
output-driven preserving, 83, 86, 88, 118
output drivenness. See output-driven map

Padgett, Jaye, 132, 387
Palauan, 249–250
Pali, 249
paradigm uniformity, 389
paradigmatic information, 247, 282
paradigmatic subset, 276, 356, 376, 392
Pater, Joe, 90, 108, 112, 258
Payne, David L., 56
phonotactic contrast, 224–233, 294, 329
phonotactic learning. See learning, phonotactic
phonotactic ranking information, 231, 232, see also learning, phonotactic
p-map hypothesis, 32
Polish, 134–137
Pool. See mark pooling
process, 5–6, 7, see also opaque process; surface-apparent process; surface-true process; transparent process
production-directed parsing, 187, 195, 200, 330
Pulleyblank, Douglas, 397
Radical CV Phonology, 282
ranking conservatism, 221, 222
RCD. See Recursive Constraint Demotion
Recursive Constraint Demotion, 178, 213, 256
reference representation space, 44, 47, 84
Reiss, Charles, 33, 42–43, 212–213, 248
relative similarity lattice, 285, 287, 289, 310
relative similarity order, 27, 285
joint, 310, 312
relative similarity relation, 27, 45, 47, 57, 158, 166, 285, 324
restrictiveness, 19, 207, 210, 211, 256, see also Biased Constraint Demotion; learning, and restrictiveness; paradigmatic subset; r-measure; subset problem
Richness of the Base, 209, 212, 280, 283, 360, 374, 387, 393
Riggle, Jason, 195, 202, 258
Rf. See correspondence relation, input-input (II)
Ristad, E. S., 392
r-measure, 215–216, 363, 392, 393
Robust Interpretive Parsing / Constraint Demotion, 259
Rosenblatt, Frank, 186
Rosenthal, Sam, 176
rotation theorem, 161, 162–167
RRS. See reference representation space
Rubach, Jerzy, 135
Sagey, Elizabeth, 282
Samek-Lodovici, Vieri, 258
Sankoff, David, 34
Sapir, David J., 56
Schachter, Paul, 56
Schane, Sanford, 249
Schmalstieg, William, 39
Servigliano, 43
setting a feature, 247, 252, 259–260, see also assigning a feature value
Shaw, Patricia, 129
Sherer, Tim, 214
similarity. See internal similarity; relative similarity lattice; relative similarity order; relative similarity relation and numeric distance metrics, 30 perceptually-based, 32 relational, 30, 33
simulation (computer), 251, 352, 370, 377–381
single form learning, 333, 334, 369, 382
Smolensky, Paul, 2, 6, 45, 84, 88, 120, 137, 141, 174, 178, 187, 195, 214, 221, 248, 255, 256, 259, 387, 389, 390, 397
Sound Pattern of English (SPE), 5, 10, 14, 34, 40, 392
Stemberger, Joseph, 214
Steriade, Donca, 32, 387
Stratal Optimality Theory, 117, 170, 399
stratified constraint hierarchy, 181 monostratal, 188, 192, 197
refinement of, 183, 194, 195, 200
stress/length system, 175–177, 206, 284
learning simulations, 377
typology of, 237
Index

structural ambiguity, 396
subset principle, 212
subset problem, 208, 372, see also restrictiveness
support, 196, 197, 200, 204, 221, 222, 328, 332, 377, 394
surface orientedness, 1–6, 13, 19, 23–26, 43, 56, 58
and Optimality Theory, 6–11
surface-apparent process, 6
surface-true process, 5
sympathy constraint, 148–150
Szász, Gábor, 4

Target-Output Closeness, 42–43
Tiberian Hebrew, 149
TOC. See Target-Output Closeness
total ranking, 181, 183, 184, 185, 187, 202, 204, 205, 395, see also stratified constraint hierarchy

transparent process, 5, 14, 39, 62
Turkel, William J., 397
underspecification, 387–388
Vallverdú, Teresa, 145
Variationist EDCD, 188, 194–195
Vergnaud, Jean-Roger, 282
viable (candidate / input / node / sublattice), 290
Wexler, Kenneth, 186
W-extension, 67, 271, 303
Wilkinson, Karina, 2
Wilson, Colin, 149
winner–loser pair, 64, 66, 177–181, 204, 395, 397, see also support
non-phonotactic, 298, 330–331
phonotactic, 224, 226–233
selecting, 185–186, 195
Wolf, Matthew, 145
Zoll, Cheryl, 61
Zonneveld, W., 156, 161