Doing Optimality Theory
Doing Optimality Theory

Applying Theory to Data

John J. McCarthy
## Contents

*Acknowledgments*  
viii  
*Read This First!*  
x  
*List of Abbreviations*  
xxii

### 1 An Introduction to Optimality Theory  
1.1 How OT Began  
1  
1.2 Why Must Constraints Be Violable?  
7  
1.3 The Nature of Constraints in OT  
13  
1.4 Candidate Sets: OT’s Gen Component  
16  
1.5 Candidate Evaluation: OT’s Eval Component  
19  
1.6 Constraint Activity  
22  
1.7 Differences between Languages  
26  
1.8 The Version of OT Discussed in This Book  
27  
1.9 Suggestions for Further Reading  
28  

### 2 How to Construct an Analysis  
2.1 Where to Begin  
30  
2.1.1 Choosing a problem to work on  
30  
2.1.2 Formulating a descriptive generalization  
33  
2.1.3 Getting from the generalization to an analysis  
37  
2.1.4 Summary  
39  
2.2 How to Rank Constraints  
41  
2.3 Working through an Analysis in Phonology  
53  
2.4 The Limits of Ranking Arguments  
65  
2.5 Candidates in Ranking Arguments  
72  
2.6 Harmonic Bounding  
80  
2.7 Constraints in Ranking Arguments  
83
Contents

2.8 Inputs in Ranking Arguments 87
2.9 Working through an Analysis in Syntax 95
2.10 Finding and Fixing Problems in an Analysis 103
  2.10.1 How to check an analysis for problems 103
  2.10.2 Problem 1: An invalid ranking argument 108
  2.10.3 Problem 2: A ranking paradox 109
  2.10.4 Problem 3: Dealing with richness of the base 113
2.11 Constraint Ranking by Algorithm and Computer 115
2.12 The Logic of Constraint Ranking and Its Uses 124

3 How to Write Up an Analysis
  3.1 Introduction 137
  3.2 How to Organize a Paper 138
  3.3 How to Present an OT Analysis 142
  3.4 The Responsibilities of Good Scholarship 152
  3.5 How to Write Clearly 157
  3.6 General Advice about Research Topics 162

4 Developing New Constraints
  4.1 Introduction 166
  4.2 When Is It Necessary to Modify Con? 167
  4.3 How to Discover a New Constraint 171
  4.4 How to Define a New Constraint 174
  4.5 Properties of Markedness Constraints
    4.5.1 How markedness constraints assign violations 176
    4.5.2 Constraints that are evaluated gradiently 181
    4.5.3 Constraints derived by harmonic alignment 186
  4.6 Properties of Faithfulness Constraints
    4.6.1 Correspondence theory 195
    4.6.2 Faithfulness to features 199
    4.6.3 Positional faithfulness 203
    4.6.4 Faithfulness constraints in the early OT literature 208
  4.7 Justifying Constraints
    4.7.1 The three ways of justifying a constraint 212
    4.7.2 Justifying constraints formally 213
    4.7.3 Justifying constraints functionally 220
  4.8 A Classified List of Common Phonological
    Markedness Constraints 223

5 Language Typology and Universals
  5.1 Factorial Typology 235
  5.2 Language Universals and How to Explain Them in OT 236
5.3 Investigating the Factorial Typology of a Constraint Set 239
5.4 Using Factorial Typology to Test New Constraints 247
5.5 Factorial Typology When Con Isn’t Fully Known 250
5.6 How to Proceed from Typology to Constraints 254

6 Some Current Research Questions 260
  6.1 Introduction 260
  6.2 How Does a Language Vary? 260
  6.3 How is Language Acquired? 264
  6.4 Does OT Need Derivations? 266
  6.5 How Is Ungrammaticality Accounted For? 271
  6.6 Is Faithfulness Enough? 274

Afterword 279
References 280
Constraint Index 298
Language Index 301
Subject Index 303
Acknowledgments

This book is the product of 15 years of teaching Optimality Theory to undergraduate and graduate students at the University of Massachusetts Amherst and at two Linguistic Institutes (Cornell in 1997 and MIT in 2005). Some of it was first circulated on CD-ROM as a collection of handouts for teaching OT (McCarthy 1999). The many students I have taught over the years have shaped this work profoundly.

My colleague Joe Pater, who shares responsibility for teaching introductory phonology at the University of Massachusetts, has been an important influence, as have John Kingston and Lisa Selkirk. Alan Prince and I have talked many times about the challenges of teaching and doing OT, and I have learned much from him. His comments on portions of chapter 2 were valuable and led to significant changes in the presentation.

I am very grateful for help from the members of our weekly phonology group, who devoted a meeting to helping me brainstorm about what should go in this book and another meeting to going over the results: Leah Bateman, Michael Becker, Tim Beechey, Ioana Chitoran (Dartmouth College), Emily Elfner, Kathryn Flack, Elena Innes, Gaja Jarosz, Karen Jesney, Mike Key, Wendell Kimper, Kathryn Pruitt, Ma Qiuwu (Nankai University), Nathan Sanders (Williams College), Ellen Simon (Ghent University), and Matt Wolf. In addition, I received useful written comments from Ubiratã Kickhöfel Alves, Ioana Chitoran, Gaja Jarosz, and Shigeto Kawahara. The students in a course I taught at the III Seminário Internacional de Fonologia in Porto Alegre, Brazil, gave me lots of good feedback through their comments and questions.
Acknowledgments

Kathryn Flack read the entire book and gave me literally hundreds of useful and insightful comments. I’ve used as many of them as I could. I’m also grateful to Anna Oxbury for her superb work copy-editing the manuscript.

This book is dedicated to my nephews and niece Michael, Jack, and Kennedy McCarthy.
Read This First!

The goal of this book is to explain how to do analysis and research in Optimality Theory. Many of OT’s basic premises are very different from other linguistic theories. This means that OT requires new and often unfamiliar ways of developing analyses, arguing for them, and even writing them up. Furthermore, the research frontier in OT isn’t too distant from the core of basic knowledge. For this reason, any analyst, even a novice, may soon find herself or himself in the position of making proposals about universal grammar.

Throughout this book, there is plenty of practical advice – do this, but don’t do that! This advice is presented in a maximally general way, but it’s also illustrated with specific examples. The examples require minimal previous background, and explanatory notes (printed in a box) are included wherever necessary. The intended audience includes readers who are encountering OT for the first time as well as those who are more advanced. Because of its focus on practical matters, this book is a good companion to my Thematic Guide to Optimality Theory (McCarthy 2002), which has a more theoretical and polemical orientation.

Chapter 1 provides a succinct summary of the core concepts of OT. Readers who are new to OT will want to read this chapter closely. Chapter 2 explains how to construct, justify, and test an analysis. Chapter 3 is the expositional counterpart to chapter 2: it provides suggestions and a model for writing up OT analyses in a way that is clear and persuasive. A lot of chapter 3’s advice about writing in linguistics is independent of OT, so even readers who aren’t interested in OT might find it useful.

In OT, it’s sometimes necessary to posit new universal constraints or modify old ones. This special responsibility is the topic of chapter 4.
Positing a new constraint changes the typological predictions, and so chapter 5 addresses that topic. Finally, chapter 6 introduces some of the areas of recent research.

Every few pages, there are sections labeled “Exercises” or “Questions.” The primarily theoretical chapters (1 and 6) have more questions than exercises, and the primarily practical chapters (2–5) have more exercises than questions. The exercises call for pencil and paper (or computer and keyboard); they are good practice and suitable as homework assignments. A few of the questions could also be assigned as homework, but most of them are more open-ended; they are intended to stimulate thought and discussion. By dispersing the exercises and questions throughout the book rather than leaving them for the ends of chapters, I hope to encourage readers to master each concept or technique before going on and to engage in a continuing dialogue with the text and with their classmates and teachers.
Abbreviations

Abbreviations of constraint names can be found in the index of constraints.

C       Any consonant
CON     Constraint component of Optimality Theory
EVAL    Evaluator component of Optimality Theory
GB      Government Binding Theory (Chomsky 1981)
GEN     Generator component of Optimality Theory
OT      Optimality Theory
ROA     Rutgers Optimality Archive, http://roa.rutgers.edu
SPE     The Sound Pattern of English (Chomsky and Halle 1968)
t      A syntactic trace
UG      Universal grammar
V       Any vowel
XP      Any phrasal category
1

An Introduction to Optimality Theory

1.1 How OT Began

Around 1990, Alan Prince and Paul Smolensky began collaborating on a new theory of human language. This collaboration led in fairly short order to a book-length manuscript, \textit{Optimality Theory: Constraint Interaction in Generative Grammar}. Photocopies of the manuscript were widely distributed and had a terrific impact on the field of linguistics, even though it wasn’t formally published until more than a decade later (as Prince and Smolensky 1993/2004). OT had and continues to have its greatest effect on phonology, but it has also led to important work in syntax, semantics, sociolinguistics, historical linguistics, and other areas. OT belongs on anyone’s list of the top three developments in the history of generative grammar.

One of Prince and Smolensky’s goals for OT was to solve a long-standing problem in phonology. Phonological theory in the tradition of Chomsky and Halle’s (1968) \textit{The Sound Pattern of English (SPE)} was based on rewrite rules. The rewrite rule $A \rightarrow B/C\_\_D$ describes an input configuration CAD and an $A \rightarrow B$ transformation that applies to it. Rewrite rules can describe lots of phenomena, but they do a poor job of explaining how phonological systems fit together. (For a brief explanation of SPE’s main assumptions, see the boxed text at the end of this section.)

To illustrate, we’ll look at some data from Yawelmani, a nearly extinct dialect of the California Penutian language Yokuts (Newman 1944).\textsuperscript{1} In this language, syllables cannot be bigger than CVC (consonant-vowel-consonant). Various phonological processes are involved with this limit on syllable size. For instance, Yawelmani has a process that
deletes a vowel at the end of a word, as the data in (a) of (1) show. (The “.” marks the boundary between two syllables.) But the data in (b) show that final vowels do not delete when they are preceded by a consonant cluster. The explanation for the difference between (a) and (b) is that deletion after a cluster would require a syllable that is too big or leave a consonant that cannot be syllabified: *\([xat^k]\).²

(1) Yawelmani final vowel deletion

<table>
<thead>
<tr>
<th>Underlying</th>
<th>Surface</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>/taxaː-ka/</td>
<td>[ta.xaː] ‘bring!’</td>
<td></td>
</tr>
<tr>
<td>/taxaː-mi/</td>
<td>[ta.xam] ‘having brought’</td>
<td></td>
</tr>
<tr>
<td>/xat-ka/</td>
<td>[xat.ka] ‘eat!’</td>
<td></td>
</tr>
<tr>
<td>/xat-mi/</td>
<td>[xat.mi] ‘having eaten’</td>
<td></td>
</tr>
</tbody>
</table>

Yawelmani also has a process of vowel epenthesis that applies to a cluster of three consonants in the middle of a word. (See (2). The data in (a) show epenthesis into triconsonantal clusters, and the data in (b) show that there is no epenthesis in smaller clusters.) If there were no epenthesis process, then the result would again be a syllable that is too big or a consonant that cannot be syllabified: *\([ʔi.lik.hin]\).

(2) Yawelmani vowel epenthesis

<table>
<thead>
<tr>
<th>Underlying</th>
<th>Surface</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>/ʔi.lik-hin/</td>
<td>[ʔi.lik.hin] ‘sing (nonfuture)’</td>
<td></td>
</tr>
<tr>
<td>/lihm-hin/</td>
<td>[li.him.hin] ‘run (nonfuture)’</td>
<td></td>
</tr>
<tr>
<td>/ʔi.lik-al/</td>
<td>[ʔi.lik.al] ‘sing (dubitative)’</td>
<td></td>
</tr>
<tr>
<td>/lihm-al/</td>
<td>[lih.mal] ‘run (dubitative)’</td>
<td></td>
</tr>
</tbody>
</table>

It is certainly possible to state SPE-style rewrite rules to account for these two processes in Yawelmani – V → Ø/VC___# and Ø → i/C___CC will do the job nicely. But, as Kisseberth (1970) first argued, those rewrite rules are missing an important generalization about the special role of surface-structure constraints in both rules. Final vowel deletion cannot create bad syllables in surface forms, and epenthesis exists to eliminate them. Adopting a suggestion from Haj Ross, Kisseberth called this situation a conspiracy.

When two or more rewrite rules are involved in a conspiracy, they directly or indirectly support some constraint on surface forms. In Yawelmani, the relevant constraints are a CVC limit on syllable size and a prohibition on unsyllabified consonants. Final vowel deletion is
blocked from applying when it would produce a surface form like *[xatk*], which cannot be fully parsed into maximally CVC syllables. Epenthesis is triggered to apply by the need to fix sequences that cannot be parsed into CVC syllables. In every conspiracy there is a constraint on surface forms, which we can refer to more succinctly as an output constraint, since it evaluates the output of the grammar. There is also some mixture of processes that are blocked by that output constraint and/or processes that are triggered by it.

Conspiracies are common in the languages of the world, and so it was a matter of some concern that the SPE theory of rewrite rules couldn’t explain them. A rewrite rule, by its very nature, describes an input configuration and an operation that applies to it. A conspiracy is completely different: it refers to an output configuration, it involves several different operations, and those operations may participate in the conspiracy by applying or failing to apply, depending on the circumstances. When analysts try to describe conspiratorial behavior in terms of rewrite rules, they have to start using counterfactuals, as I did in the preceding paragraph: “blocked from applying when it would produce,” “to fix sequences that could not be parsed.” Statements like these show that the analyst understands what’s really going on in the language, but counterfactual conditions have no place in SPE’s theory of how to apply rules. When a phonologist says something like “The epenthesis rule ensures that the language has only unmarked syllables,” he or she is describing an intuition about how the system works. But that intuition has to be expressed formally, in the theory itself. Otherwise, we are just telling ourselves stories.

At around the same time that phonologists were beginning to grasp the importance of output constraints, syntacticians were having a similar revelation. For example, clitic movement in Spanish is triggered by an output constraint requiring that second person clitics precede first person clitics (Perlmutter 1971). That is why the direct and indirect objects appear in different orders in Te_{IO} me_{DO} presento ‘I introduce myself to you’ versus Te_{DO} me_{IO} presentas ‘You introduce yourself to me.’ Another example: in English, movement of wh-question words is blocked when it would leave the trace of wh immediately after the complementizer that (Chomsky and Lasnik 1977): *Who did you say that t left? (cf. Who did you say t left?). These syntactic examples have parallels in Yawelmani phonology. The triggering of clitic movement by an output constraint in Spanish is like the triggering of epenthesis in Yawelmani. And the blocking of wh-movement in English is like the blocking of final vowel deletion in Yawelmani.
Chomsky and Lasnik (1977) proposed a theory of output constraints and their function that had (and continues to have) a great deal of influence in syntax. Their main idea is that all of the rewrite rules – that is, the syntactic transformations – are technically optional. An input to the grammar freely undergoes any, all, or none of the transformations. The result of freely applying transformations is a set of candidate surface structures. These candidate surface structures are checked by the output constraints, which are called filters, and some of them are marked as ungrammatical by the filters. For instance, the *wh*-movement transformation applies optionally, producing both *Who did you say that t left?* and *You said that who left?* as candidate surface structures. The *that*-trace filter marks the first of these as ungrammatical, so only the second is well-formed. Henceforth, I’ll refer to Chomsky and Lasnik’s proposal as the filters model.

The filters model does a good job of explaining how output constraints can seem to trigger or block transformations. Because the transformations are strictly optional, if there is a candidate surface structure that has undergone a transformation T, there is also a candidate derived from the same deep structure that has not undergone T. If a filter marks the result of applying T as ungrammatical, then the filter has in effect blocked T, since the derivation in which T has applied does not lead to a well-formed output. If a filter marks the result of *not* applying T as ungrammatical, then the filter has in effect triggered T, since the derivation in which T has failed to apply does not lead to a well-formed output. The filter isn’t literally triggering or blocking T – it cannot, since the filter doesn’t even apply in the same grammatical component as T – but the filter appears to be blocking or triggering T by ruling out the surface structure where T has or has not applied.

A goal of the filters model was to shift most of the burden of explaining syntactic patterns from the theory of transformations to the theory of filters. Transformations could be made much simpler and more general. In Government-Binding Theory (GB) (Chomsky 1981), the theory of transformations withered away almost entirely, leaving just the transformation Move α.

Although the filters model in syntax emerged not long after the discovery of the conspiracy problem in phonology, the filters model had surprisingly little influence on the field of phonology at that time. There are two main reasons for this, in my opinion. One of them, which I will explain in the next section, is that the filters model fails as an explanation for phonological conspiracies like Yawelmani’s if output
constraints are inviolable, and constraints of that era were always assumed to be inviolable.

The other reason is that the field of phonology was so strongly influenced by SPE. SPE’s central hypothesis is that rules with simpler formulations are more natural, in the sense that they are more likely to occur in languages and to express linguistically significant generalizations. In accordance with this hypothesis, the SPE theory supplies abbreviatory devices that allow putatively natural rules to be formulated more simply. In a conspiracy, the output constraint is what makes the rules natural – the output constraint is the generalization that unites the disparate rules. Therefore, a theory of conspiracies embedded in the overall SPE research program would have to use the output constraint to simplify the statement of the rules that participate in the conspiracy.

Kisseberth (1970) proposed a theory of blocking effects along exactly these lines. He assumed for yawelmani an output constraint */CC[C, #] that is violated by medial clusters of three consonants or final clusters of two consonants. By assumption, a rule is blocked from applying if its immediate output violates this constraint. That assumption allows the formulation of the final vowel deletion rule to be simplified from $V \rightarrow \emptyset/VC___#$ to $V \rightarrow \emptyset/___#$. And since simpler rules are more natural under SPE’s assumptions, the existence of the output constraint has in some sense explained why final vowel deletion is blocked after consonant clusters.

Kiparsky (1973b: 77–78) presents several criticisms of this proposal. One problem is that the rule $V \rightarrow \emptyset/___#$ is just as as simple, and therefore should be just as natural, in a language without the */CC[C, #] output constraint. But a language without the output constraint is a language without the conspiracy, and if conspiracies contribute to naturalness, then the language without one should be less natural. Another problem with this proposal is that it only works for blocking effects. Rules that are triggered by the output constraint won’t receive simpler formulations. For instance, there is no way of using the output constraint to simplify the statement of the epenthesis rule, replacing $\emptyset \rightarrow i/C___CC$ with, say, $\emptyset \rightarrow i/___C$. The problem with $\emptyset \rightarrow i/___C$ is that it would epenthesize [i] before every single consonant. The theory at that time lacked any sort of economy mechanism to ensure that epentheses applies only when it’s needed and not otherwise.

Starting in the mid-1970s, phonological research moved toward richer theories of representation that included syllables and other
structures. As phonological representations became more elaborated, it became possible to imagine an almost ruleless phonology in which automatic satisfaction of universal constraints on representations was all that mattered. Goldsmith (1976a, 1976b) and Prince (1983) worked on proposals along these lines for autosegmental and metrical phonology, respectively. This work ran headlong into another problem, however: the proposed universal constraints did not hold in every language all of the time. That is why the subsequent literature on autosegmental and metrical phonology, such as Pulleyblank (1986) and Hayes (1995), returned to language-particular rewrite rules as the central analytic mechanism.

By the end of the 1980s, there was certainly a consensus about the importance of output constraints, but there were also major unresolved questions about the nature and activity of these constraints. That "conceptual crisis at the center of phonological thought," as Prince and Smolensky (1993/2004: 2) refer to it, was not very widely acknowledged at the time, but in hindsight it’s hard to miss. It’s a major feature of the intellectual context in which OT was developed.

Explanation: The SPE theory and its relation to OT phonology

In SPE, every morpheme is assumed to have a unique underlying representation that is stored in the lexicon. The underlying representation includes all of the unpredictable phonological properties of a morpheme. For example, the Yawelmani imperative suffix has surface alternants [-k’a] and [-k’], and the nonfuture suffix has alternants [-mi] and [-m] (see (1)). Their underlying representations are /-k’a/ and /-mi/. (The underlying representations couldn’t be /-k’/ and /-m/, because there would be no good way of explaining why [a] is epenthesized in one suffix and [i] in the other.)

The mapping from underlying to surface representations is accomplished by applying a series of ordered rewrite rules. For instance, the path from /taxa-k’a/ to [ta.xak’] requires two rules: first, the final vowel is deleted, yielding [ta.xak’], and then the vowel is shortened to produce the surface form [ta.xak’]. As the text mentions, a rewrite rule is an expression A \rightarrow B/C\_D that changes any CAD sequence into a CBD sequence. OT does not have rewrite rules or anything that resembles them.

SPE also includes a theory of representations. Every speech sound consists of a bundle of values for certain universal, binary distinctive features: [nasal], [round], and so on. In the 1970s and 1980s, SPE’s rather
simple representational theory was greatly enhanced. For instance, SPE does not include syllables in its representations, but later work would analyze the \([\text{ta}\.\text{xa}\.\text{k}^2] \rightarrow [\text{ta}\.\text{xak}^2]\) mapping as a process of vowel shortening in a syllable that is closed by a consonant.

Most work in OT phonology presupposes SPE’s view of underlying representations, its theory of distinctive features, and many of the subsequent representational enhancements. It’s important to realize, however, that OT itself does not require a commitment to any of these ideas.

QUESTIONS

1. How will the filters model work when several different transformations are applicable? What about when a transformation is applicable at several different places in a sentence? What about when a transformation is applicable to its own output?

2. The text promises that the next section will give an argument that the filters model cannot explain phonological conspiracies if constraints are inviolable. Try to figure out the argument before reading the section. (Hint: An output constraint is needed to ensure that final vowel deletion occurs in \(/\text{taxa}:k^2\alpha/ \rightarrow [\text{ta}\.\text{xak}^2]\).)

EXERCISE

3. Yawelmani has output constraints that limit syllables to a CVC maximum and require exhaustive syllabification. In Yawelmani, these constraints trigger epenthesis and block final vowel deletion. Can you imagine a different language that has the same output constraints but which block and/or trigger other processes? Hypothetical examples are fine; it isn’t necessary to identify actual languages.

1.2 Why Must Constraints Be Violable?

In the previous section, I alluded to a second reason why phonology did not develop an optional-rules-plus-output-constraints theory, similar to the filters model in syntax. The main impediment was the assumption, standard at the time, that output constraints are never violated.
Suppose we try to apply the filters model to Yawelmani. (It may be helpful to follow the chart in (3) as you read the rest of this paragraph.) Since there is epenthesis in the language, the transformational component must contain an optional epenthesis rule. Given /ʔilk-hin/ as the input to the transformational component, the output of that component will include [ʔi.lik.hin], with epenthesis, and various ways of syllabifying the word without epenthesis, such as *[ʔi.lik.hin] and *[ʔi.l.k.hin]. (I will use the notation “.k.” to indicate that the [k] is outside the syllable on its left and right. It’s unsyllabified.) These three forms are then checked by the filters. One filter, which I’ll call *C\textsubscript{unsyl}, prohibits unsyllabified consonants. It marks *[ʔi.l.k.hin] as ungrammatical. Another filter marks *[ʔi.lk.hin] as ungrammatical because it contains a syllable that exceeds the CVC limit. (I’ll call this filter *Complex-Syllable.) Since *[ʔi.l.k.hin] and *[ʔi.lk.hin] are ruled out by the two filters, [ʔi.lik.hin] is the only grammatical output from this input. From the perspective of someone looking at the output of the grammar, it looks as if the filters *C\textsubscript{unsyl} and *Complex-Syllable are triggering the epenthesis process. (For a brief explanation of the role of syllable structure in phonological processes, see the boxed text at the end of this section.)

(3) Filters model applied to Yawelmani – input /ʔilk-hin/ → [ʔi.lik.hin]

<table>
<thead>
<tr>
<th>Input</th>
<th>Transformational component (all optional)</th>
<th>Output of transformational component</th>
<th>Filter component</th>
<th>Output of filter component</th>
</tr>
</thead>
<tbody>
<tr>
<td>/ʔilk-hin/</td>
<td>syllabification epenthesis</td>
<td>[ʔi.lik.hin] [ʔi.l.k.hin] [ʔilk.hin]</td>
<td>*C\textsubscript{unsyl} *Complex-Syllable</td>
<td>*[ʔi.lk.hin] vs. *ʔi.lk.hin *ʔi.lk.hin</td>
</tr>
</tbody>
</table>

Since Yawelmani also has final vowel deletion, the transformational component would also have to contain an optional rule that deletes final vowels. As a result of this rule, the output of the transformational component will include both [ta.xak’] and *[ta.xa:k’a]. Since *[ta.xa:k’a] is ungrammatical, some filter must rule it out. What filter? The obvious move is to posit a filter that forbids word-final vowels. We can call this filter *V#.
Why Must Constraints Be Violable?

When we try to extend this analysis to the input /xat-k’a/, however, we run into trouble. (Follow along in (4).) Among the outputs of the transformational component are [xat.k’a] (which is correct), *[xat.k’] (with an unsyllabified consonant), and *[xatk’] (with a syllable that is too big). Unfortunately, all of these forms, including the correct one, violate some filter. The forms *[xat.k’] and *[xatk’] are marked as ungrammatical because they violate the filters *Cunsyll and *Complex-Syllable, respectively. The form that we want, [xat.k’a], is marked as ungrammatical by the filter *V#, which was needed to make final vowel deletion obligatory in [ta.xak’]. The only form that isn’t marked as ungrammatical is *[xa.tik’], which is wrong.

(4) Filters model applied to Yawelmani – input /xat-k’a/ → wrong output

<table>
<thead>
<tr>
<th>Input</th>
<th>Transformational component (all optional)</th>
<th>Output of transformational component</th>
<th>Filter component</th>
<th>Output of filter component</th>
</tr>
</thead>
<tbody>
<tr>
<td>/xat-k’a/</td>
<td>syllabification epenthesis final vowel deletion</td>
<td>[xat.k’a] [xat.k’] [xatk’] [xa.tik’]</td>
<td>*Cunsyll *Complex-Syllable *V#</td>
<td>[xa.tik’] vs. *[xat.k’a] *[xat.k’] *[xatk’]</td>
</tr>
</tbody>
</table>

This analysis fails because it’s based on a wrong assumption, the assumption that filters are never violated. If filters are inviolable constraints on outputs, Yawelmani cannot possibly have a filter *V# – obviously, since it has vowel-final words like [xat.k’a]! We could get around this problem by replacing *V# with a more specific filter, *VCV#, but this would be admitting defeat. The filter *VCV# stipulates something that our analysis really should explain: final vowel deletion is blocked in [xat.k’a] because letting it apply would produce an unsyllabified consonant or a syllable that is too big. If we haven’t explained that, then we haven’t really accounted for Yawelmani’s conspiracy.

A real explanation needs to derive the failure of final vowel deletion in [xat.k’a] from the independently necessary filters *Cunsyll and *Complex-Syllable. The idea goes something like this. Even though [xat.k’a] violates *V#, the alternative *[xat.k’] is even worse, since it violates *Cunsyll, and *Cunsyll has a higher priority than *V#. To say the
same thing in a different way, *V# triggers final vowel deletion, but
the constraint *C\text{unsyll} sometimes blocks satisfaction of *V#. The same
goes for *\text{COMPLEX-SYLLABLE}: it too has higher priority than *V#, so it
too can block satisfaction of *V#. (You will be dealing with the
*\[xa.tik\] problem in exercise 17 in chapter 2.)

Although constraint priority relationships were occasionally mentioned
in the pre-OT literature (e.g., Burzio 1994), the standard assumption
was that all output constraints are inviolable and therefore unpriorit-
ized. The central thesis of OT, on the other hand, is that constraints
are ranked and violable. Constraint prioritization is fundamental to the
theory (Prince and Smolensky 1993/2004). The comparison between
\[xat.k'\text{a}] and *\[xat.k']\ reveals a type of constraint conflict between *V#
and *C\text{unsyll}: a form that obeys one violates the other (see (5)). If *V#
takes precedence, then the result is *\[xat.k']\, which obeys *V# at the
expense of violating *C\text{unsyll}. If *C\text{unsyll} takes precedence, then the result
is \[xat.k'\text{a}\], which obeys *C\text{unsyll} but violates *V#. Since \[xat.k'\text{a}\] is what
we want, priority goes to *C\text{unsyll}.

\begin{center}
\begin{tabular}{|c|c|}
\hline
& \text{*C\text{unsyll}} & \text{*V#} \\
\hline
\[xat.k'\text{a}\] & obeyed & violated \\
\hline
*\[xat.k'] & violated & obeyed \\
\hline
\end{tabular}
\end{center}

In OT terms, the higher-priority constraint dominates the lower-
priority constraint. *C\text{unsyll} must dominate *V# in the grammar of
Yawelmani. We write this as *C\text{unsyll} >> *V#. *\text{COMPLEX-SYLLABLE} also
dominates *V#. This means that *V# will be satisfied only when this
doesn’t require an output with an unsyllabified consonant or a too-
big syllable. With the input \[xat-k'\text{a}\], these constraints impose conflict-
ing demands and the higher-ranking ones are decisive, blocking
vowel deletion. With the input /taxa:k'\text{a}/, however, the final vowel
can be deleted with no danger of leaving a consonant unsyllabified
or creating a syllable that is too big (see (6)). In this case, there is
no conflict between *C\text{unsyll} and *V#, so both of them can and must
be satisfied. Constraints are violable in OT, but violation is never
gratuitous; it must always be compelled by some higher-ranking,
conflicting constraint.
The goal in this discussion of Yawelmani was to explain away a conspiracy by deriving the failure of final vowel deletion in VCCV# words from independently necessary constraints on syllabification. The OT analysis that I’ve just sketched does exactly that: there is no final vowel deletion in [xat.k?a] because the alternatives, *[xat.k?] and *[xatk?], leave a consonant unsyllabified or require a syllable that exceeds the language’s limit on size. The most important and novel elements of this explanation are constraint ranking and violability, which allow *V# to be active in Yawelmani but not always satisfied. This seemingly modest change in how to think about output constraints is in reality quite profound, with important implications that are still being explored more than a decade later. In the rest of this chapter we will see some of those implications.

### Syllable structure and phonological processes

One of the most important developments in phonology during the 1970s and 1980s was the realization that syllable structure affects many phonological processes. Vowel epenthesis, for example, is often motivated by the need to fit consonants into restrictive syllable templates. Yawelmani /?ilk-hin/ → [?i.lïk.hin] is an example of this; because of epenthetic [i], the [k] can fit into Yawelmani’s maximally CVC syllables, whereas without the [i] it couldn’t (*[?i.lk.hin] or *?[ilk.hin]). Syllable structure requirements can also block processes, such as final vowel deletion in Yawelmani /xat-k?/ → [xat.k?].

Syllable structure offered some help with the conspiracy problem, but not enough. Selkirk (1981) proposed to solve the problem of how epenthesis is triggered by assuming that the initial pass of syllabification is able to create “degenerate” syllables that lack a vowel nucleus: [?i.l∆k.hin], with ∆ standing for an empty nucleus constituent in the second syllable. In this way, the language’s syllable structure template determines where and when epenthetic vowels are required.
The epenthesis process itself is just a matter of spelling out the empty nucleus as [i].

There were intractable problems in trying to extend this sort of approach to blocking effects, however. The /xat-k’a/ → [xat.k’a] mapping tells us that final vowel deletion is blocked because its output cannot be exhaustively syllabified. But when final vowel deletion is applied to /ta.xa:k’a/, the immediate output is [ta.xa:k’], which also cannot be exhaustively syllabified. Presumably the difference is that Yawelmani also has a process of closed syllable shortening that changes [ta.xa:k’] into the final output [ta.xak’], which can be exhaustively syllabified. The derivation, then, is /ta.xa:k’a/ → [ta.xa:k’] → [ta.xak’]. By the same logic, then, what’s wrong with the derivation /xat-k’a/ → [xat.k’] → *[xa.tik’], since Yawelmani also has a process of vowel epenthesis? Clearly, there were difficult problems in sorting out when languages block processes and when they allow them to apply but fix up the results. (See Goldsmith (1990: 319ff.), Myers (1991), Paradis (1988a, 1988b), and Prince and Smolensky (1993/2004: 238–257) for discussion of this and related issues.)

The importance of syllable structure in phonology continues to be recognized in most OT work. There is nothing in OT per se, however, that requires a commitment to any particular theory of syllable structure or even to the existence of syllables.

**QUESTIONS**

4 “[T]he standard assumption was that all output constraints are inviolable and therefore unprioritized.” Why “therefore”? Explain the connection between constraint violability and constraint prioritization.

5 “*Closed-Syllable must dominate the constraint *V# in the grammar of Yawelmani. . . Likewise *Complex-Syllable dominates *V#. This means that *V# will be satisfied only when this doesn’t require an output with an unsyllabified consonant or a too-big syllable.” Why does it mean that?

**EXERCISE**

6 The following Three Laws of Robotics are cited by Asimov (1950) from the *Handbook of Robotics* (56th edition, published 2058). Restate the laws as ranked constraints.
The Nature of Constraints in OT 13

1. A robot may not injure a human being, or, through inaction, allow a human being to come to harm.
2. A robot must obey the orders given to it by human beings except where such orders would conflict with the First Law.
3. A robot must protect its own existence as long as such protection does not conflict with the First or Second Law.

1.3 The Nature of Constraints in OT

In OT, constraints on output forms are called markedness constraints to distinguish them from constraints of a very different sort, faithfulness constraints. Faithfulness constraints prohibit differences between input and output. When underlying /taxa:-k’a/ maps to surface [ta.xak’], faithfulness constraints against vowel deletion and vowel shortening are violated. When underlying /?ilk-hin/ maps to surface [?i.lik.hin], there is a violation of a different faithfulness constraint, one that prohibits vowel epenthesis.

Faithfulness constraints are one of Prince and Smolensky’s cleverest and least obvious ideas. No other theory of language has anything like OT’s faithfulness constraints. Faithfulness constraints only make sense in a theory like OT that allows constraints to be violated. The reason: phonology and syntax are full of examples of unfaithful mappings like /taxa:-k’a/ → [ta.xak’] and /?ilk-hin/ → [?i.lik.hin], so faithfulness constraints have to be violable if they are going to be at all useful.

The job of a constraint is to assign violation marks to candidates. (Violation marks are conventionally written as asterisks.) Depending on how the constraint is defined and what the candidate is, a constraint can assign any number of marks from zero upwards. For example, *V# assigns no marks to [ta.xak’], since [ta.xak’] ends in a consonant. It assigns one mark to *[ta.xa:-k’a], however, since *[ta.xa:-k’a] ends in a vowel. The anti-epenthesis faithfulness constraint assigns one violation mark for every epenthesized segment. This constraint is called D[ep], because it requires the output to depend on the input as the source of all its segments. As (7) shows, D[ep] assigns no violation marks to *[?i.il.k.hi.n], one mark to *[?i.lik.hin], two marks to *[?i.li.ki.hi.ni], and so on. Each constraint’s definition tells us how to determine the number of violation marks that it assigns to a given candidate.
14 An Introduction to Optimality Theory

Violation marks assigned by Dep

<table>
<thead>
<tr>
<th></th>
<th>Dep</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.</td>
<td>𝑖.𝑙.𝑘.ℎ𝑖𝑛</td>
</tr>
<tr>
<td>b.</td>
<td>𝑖.𝑙𝑖.𝑘𝑖.ℎ𝑖𝑛</td>
</tr>
<tr>
<td>c.</td>
<td>𝑖.𝑙𝑖.𝑘𝑖.ℎ𝑖.𝑛𝑖</td>
</tr>
<tr>
<td>d.</td>
<td>𝑖.𝑙𝑖.𝑘𝑖.ℎ𝑖.𝑛𝑖</td>
</tr>
</tbody>
</table>

Dep favors *[𝑖.𝑙.𝑘.ℎ𝑖𝑛] over *[𝑖.𝑙𝑖.𝑘𝑖.ℎ𝑖.𝑛𝑖], and so on (Samek-Lodovici and Prince 1999). Furthermore, Dep favors *[𝑖.𝑙𝑖.𝑘𝑖.ℎ𝑖.𝑛𝑖] over *[𝑖.𝑙𝑖.𝑘𝑖.ℎ𝑖.𝑛𝑖], and so on. Likewise, Dep favors *[𝑖.𝑙.𝑘.ℎ𝑖.𝑛𝑖] over *[𝑖.𝑙.𝑘.ℎ𝑖.𝑛𝑖], and so on. These preferences are Dep’s favoring relation over this set of candidates. If a constraint assigns n violation marks to some candidate, then it favors that candidate over all of the candidates to which it assigns more than n marks. The candidates that totally obey a constraint are just one aspect of the constraint’s favoring relation. Because constraints are violable in OT, it often happens that all viable candidates violate some constraint. In that case, it’s important to know which candidates the constraint favors among those that violate it. For example, *[𝑖.𝑙.𝑘.ℎ𝑖𝑛] is ruled out because of its unsyllabified [k], so violation of Dep is unavoidable. The form *[𝑖.𝑙𝑖.ℎ𝑖.𝑛𝑖] is optimal because it is most favored among the Dep-violating candidates, as we can see from (7).

In general, the candidates that are most favored by some constraint C have two things in common: they receive the same number of violation marks from C, and no other candidate receives fewer violation marks from C. There is always at least one candidate that is most favored by C. At the other extreme, it’s possible for all of the candidates to be most favored by C, if all candidates violate C equally.

Constraints are a major focus of research effort in OT, and that is why this book devotes an entire chapter (chapter 4) just to the problems of discovering, defining, and improving constraints. Furthermore, as we will see in chapter 5, most explanations and predictions in OT derive from specific assumptions about which constraints exist. The activities of modifying or rejecting old constraints and positing new ones are important research tools and important responsibilities of researchers working in OT.
Although research on constraints is central to OT, OT itself does not say much about the nature of constraints, beyond distinguishing between markedness and faithfulness. OT is a theory of how constraints interact with one another; it isn’t a theory of what the constraints are, nor is it a theory of representations. For example, OT does not commit the analyst to any particular approach to syllable structure or phrase structure. Instead, OT supplies a framework for applying the constraints and evaluating the representations that are a necessary part of any theory of syllable structure or phrase structure. This is the reason why it has been possible to apply OT to phonology, syntax, and semantics, despite their different subject matter.

Prince and Smolensky put forward two very strong hypotheses about the universality of constraints. First, the constraints themselves are universal. Universal Grammar (UG) includes a constraint component Con that contains the entire repertoire of constraints. (There are separate Cons for phonology and syntax, with some overlap in their formal properties.) Second, all constraints are present in the grammars of all languages. These hypotheses follow from the more general assumption that constraint ranking is the only systematic difference between languages. (More about this in the next section.)

In actual practice, the hypothesis of absolute constraint universality is usually somewhat weakened. It may be necessary to admit language-particular limitations on the domains of constraints in the lexicon to deal with exceptions, loan words, and the like. There may also be formal schemas for constructing language-particular constraints, such as alignment or constraint conjunction. I will say more about these issues in chapter 4.

QUESTIONS

7 Chomsky (1995: 380) says this: “In Prince and Smolensky 1993, there seems to be no barrier to the conclusion that all lexical inputs yield a single phonetic output, namely, whatever the optimal syllable might be (perhaps /ba/).” This is sometimes known as the “ba objection” to OT. Respond to it.

8 Chomsky (1995: 380) criticizes faithfulness constraints on the grounds that identity between input and output is “a principle that is virtually never satisfied.” Respond to this criticism.

9 “Because constraints are violable in OT, it often happens that all viable candidates violate some constraint. In that case, it’s important to know which candidates the constraint favors among those that violate it.” In light of this
statement, would you describe the presentation of constraint conflict in (5) as somewhat misleading? How would you correct this?

1.4 Candidate Sets: OT’s GEN Component

In Chomsky and Lasnik’s filters model, the transformations are all optional, so the transformational component produces a variety of possible outputs in which transformations have and have not applied. The filter component marks some of these possible outputs as ungrammatical. In OT, the equivalent of the transformational component is called the Generator, or GEN for short. The list of possible outputs supplied by GEN for a given input is called the candidate set for that input. The relationship among the input, GEN, and the candidate set is diagrammed in (8).

(8) Partial flowchart for OT

\[ \text{/input/} \rightarrow \text{GEN} \rightarrow \{\text{cand}_1, \text{cand}_2, \ldots\} \]

Details of the input and of GEN, like details of the constraints, depend on our theory of representations and whether we are analyzing phonology, syntax, or semantics.

In phonology, where there is the widest agreement on such matters, the input is usually taken to be identical with the underlying representation of generative phonology. This is a level of representation in which every morpheme that alternates regularly has a unique form, such as plural /-z/ in /bæg-z/, /bʊk-z/, and /nʊz-z/ (bags, books, and noses). The phonological GEN performs various operations on the input, deleting segments, epenthesizing them, and changing their feature values. These operations apply freely, optionally, and repeatedly to derive the members of the candidate set. For example, the candidate set from input /bʊk-z/ will include the results of rightward and leftward voice assimilation ([bʊks], [bʊgz]), epenthesis ([bʊkəz]), deletion ([bʊk]), and various combinations of these processes (e.g., [bʊkəs]). It will also include a faithful candidate, where nothing has happened: [bʊkz]. These diverse candidates, nearly all of which are ungrammatical, aren’t the final output of the grammar; the final output is determined by how the constraint component filters the candidate set.

Candidates compete to be realizations of some input. For example, [bʊks], [bʊgz], [bʊkəz], [bʊk], [bʊkəs], [bʊkz], etc. all compete to be
the surface realization of the input /buk-z/. Candidates from different inputs do not compete; there is no comparison of the mapping /buk-z/ → [buk] with the mapping /noz-z/ → [nozaz]. Therefore, Gen defines the range of competitors for a given input. This range must include at least all of the ways that the input could be realized in any possible human language. In phonology, the candidate set typically contains much more than that – perhaps even every possible sequence of segments. In syntax, the nature of the candidate set is more of an open question, though see §2.9 and Legendre, Smolensky, and Wilson (1998) for discussion of how to go about answering this question, starting from OT’s basic premises about competition.

It makes sense to assume that the operations in Gen are extremely general. The epenthesis operation, for example, does not specify certain contexts for epenthesis or certain segments to be epenthesized. Instead, it can insert any segment in any context. Of course, there are all sorts of limits on what can be epenthesized and where epenthesis can happen in actual output forms. But Gen isn’t the place to impose these limits. Instead, an important goal of research in OT is to derive the language-particular and universal properties of linguistic processes from a specific theory of Con and the assumption that grammars are rankings of Con. A similar goal was articulated for the filters model: to show “that the consequences of ordering, obligatoriness, and contextual dependency can be captured in terms of surface filters . . . and further, that these properties can be expressed in a natural way at this level” (Chomsky and Lasnik 1977: 433). Government-Binding theory (Chomsky 1981) was an attempt to follow through on this goal by reducing the transformational component of the grammar to a single optional context-free operation, Move α. This highly general transformation is in the same spirit as OT’s Gen.

If Gen is so unrestricted in its effects, then the candidate set is infinite. There are infinitely many candidates if Gen includes context-free structure-building operations like epenthesis in phonology or phrase-structure projection in syntax. These operations are allowed to apply indefinitely many times in candidate formation. For example, the candidates based on the input /noz-z/ will include not only [nozaz] but also [nozəaz], [nozəəaz], and so on.

The diversity and infinity of candidates is a source of worry to many people when they first encounter OT, and I will try to lay these worries to rest now.
The diversity of candidates can be troubling because it means that any candidate set will include forms that couldn’t possibly be the output in any language. Presumably, no human language could possibly map underlying /noːz-z/ to surface [noːzːəːz]. But if [noːzːəːz] is never optimal, what is it doing in /noːz-z/’s candidate set? The answer to this worry is that the output of GEN isn’t the final output of the grammar. The grammar as a whole does not overgenerate because the constraints filter the contents of the candidate set. Any decent theory of Con will explain why mappings like /noːz-z/ → [noːzːəːz] are impossible. That is where such explanations belong, in accordance with the overall goals of OT research that were discussed a couple of paragraphs above. This matter is the topic of chapter 5.

Another source of worry is mental or electronic computation: GEN will require infinite time to produce a candidate set, and the constraint component will require infinite time to evaluate the candidates. This worry starts from a wrong assumption: the formal definition of a theory of language is also its computational implementation. Since the very beginning, generative grammar has made a distinction between models of language competence and models of language processing or use. “If these simple distinctions are overlooked, great confusion must result,” according to Chomsky (1968: 117). There is a lot of good work on computational modeling of OT, and none of this work stumbles over the infinity of candidates because all of it recognizes the distinction between theory and implementation. See the suggestions for further reading at the end of the chapter.

QUESTIONS

10 “Any decent theory of Con will explain why mappings like /noːz-z/ → [noːzːəːz] are impossible.” How? [Hint: Think about markedness, since for [noːzːəːz] to win it must be less marked than its more faithful competitors [noːzz], [noːzaːz], and [noːzaːz].]

11 Why not put a limit on the number of epenthesis operations that GEN can perform? Would this ensure that the phonological candidate set is finite, or does the phonological GEN include other potential sources of an infinity of candidates?

12 What are some hypotheses about the input in syntactic theory? How would we go about determining which hypothesis is best?