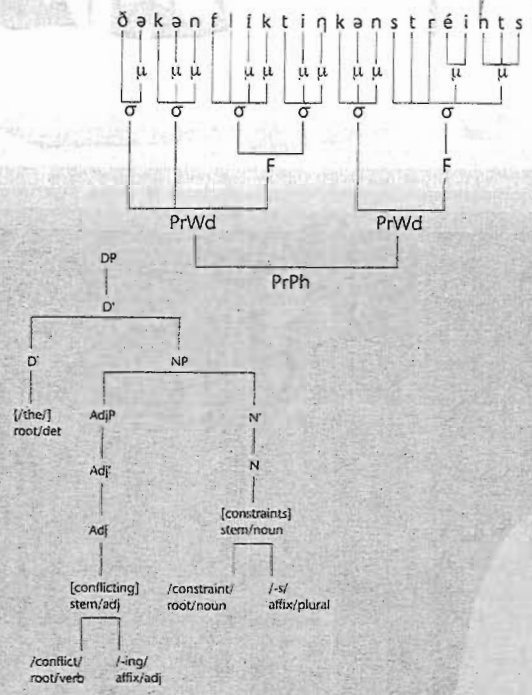
Learnability in Optimality Theory

MIT Press (2000)



Bruce Tesar and Paul Smolensky

1 Language Learning

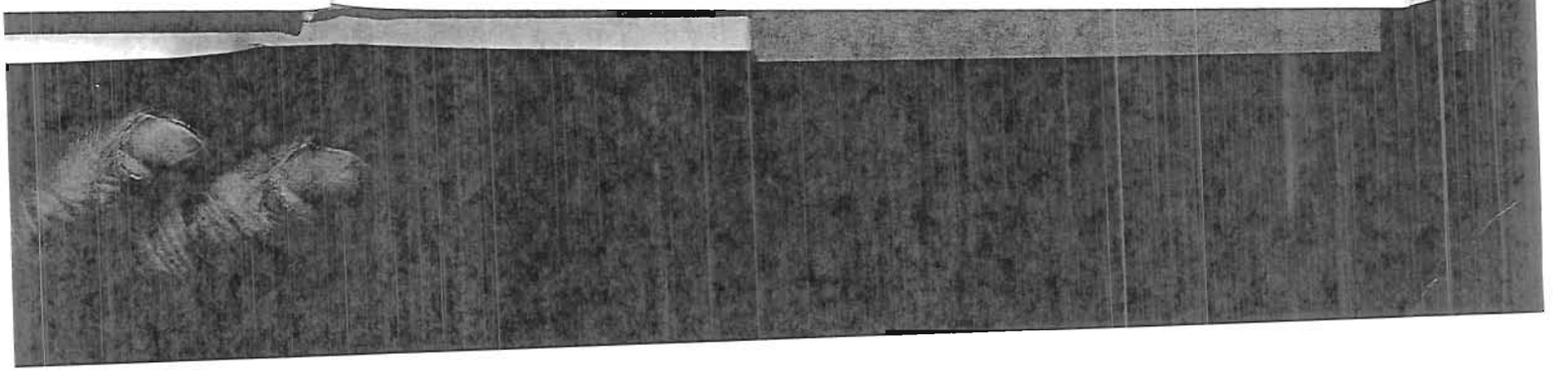
1.1 What This Book Is About

This book argues that the linguistic framework of *Optimality Theory* (OT) (Prince and Smolensky 1993) makes possible a particularly strong union of the interests of language learnability and linguistic theory. In support of this claim, a particular approach to language learning, *Robust Interpretive Parsing / Constraint Demotion* (RIP/CD), is presented and evaluated. This learning proposal is tightly bound to the central principles of OT, and the success of the learning proposal is evidence in favor of the main claim.

The language learning issue of primary concern in this book is the ambiguity of the overt information that constitutes the actual data received by a learner, and the resulting interdependence of the core grammar and the structural analysis of overt linguistic forms: which grammar a learner chooses depends on how they interpret the forms they hear, and which analysis they choose for a form depends on what grammar they are using. The RIP/CD proposal claims that this interdependence can be finessed by successive iteration: the learner can use a first guess at a grammar to estimate the structural analysis of the data, use the estimated analyses to improve the grammar, use the improved grammar to improve the analyses, and so forth. The learning procedure learns both the correct interpretations of the data and the correct grammar simultaneously. The viability of this "back-and-forth" strategy is heavily dependent on the use of OT to characterize the knowledge of language that the learner comes to possess.

The RIP/CD learning proposal is evaluated by a series of computer experiments, applying the proposal to overt data from a number of languages generated by an OT system for metrical stress. This system exhibits a nontrivial degree of ambiguity in the overt forms: most overt forms have several viable structural interpretations, with different interpretations favored by different grammars of the system. The performance is evaluated both on accuracy—whether or not the correct grammar was in fact learned—and computational efficiency—the amount of effort exerted during the process of learning the correct grammar.

The empirical results just mentioned are supported by stronger formal results concerning major parts of the proposal. It is not necessary to conduct any simulations to attempt to measure the amount of information required by the learner to determine the correct grammar, because



of a strong upper bound on the amount of data required. This result, which applies to all language systems defined within OT, is proved correct in chapter 7. This result is an important part of the proposal made here, for it demonstrates that the adoption of OT guarantees a strong solution to one of the major issues in language learning.

Chapter 1 is devoted to laying out the larger context of this work, including the nature of relationships between learnability and universal grammar, and the background work on general learning theory that has informed and inspired the specific language learning proposal made here. Readers who would prefer to skip the background on the first reading are advised to jump to section 1.4, which presents a top-level outline of the proposals made in this book, along with pointers to the location of each topic within the book.

1.2 Learnability and Universal Grammar

It has become commonplace in generative linguistics circles to see the logical problem of language acquisition as a driving force in shaping grammatical theory (Chomsky 1981). The basic logic is essentially as follows. Learning a grammar is difficult because there are so many conceivable grammars and the available data is so impoverished. Thus a crucial job of a theory of universal grammar is to *restrict* the space of possible grammars the learner must consider, so that impoverished data may suffice to determine a correct grammar. This notion of restrictiveness is often reduced to the criterion that a satisfactory grammatical theory will delimit a finite set of possible grammars—distinguished from one another by the values of a finite number of *parameters*, for example. The fewer the possible grammars, the more learnable the theory.

Or so it would seem. In fact, however, limiting the set of possible grammars to a finite number serves only to improve the worst-case performance of the *least informed* learning method of all: exhaustive search, in which every possible hypothesis is examined. For, with finitely many possible grammars, search for a correct one is guaranteed to terminate eventually: at worst, once all possible grammars have been examined. With infinitely many possible grammars, such search may continue forever.

But comfort from the finiteness of the space of possible grammars is tenuous indeed. For a grammatical theory with an infinite number of possible grammars might be well structured, permitting *informed* search that converges quickly to the correct grammar—even though uninformed, exhaustive search is infeasible. And it is of little value that exhaustive search is guaranteed to terminate eventually when the space of possible grammars is finite, if the number of grammars is astronomical. In fact, a well-structured theory admitting an infinity of grammars could well be feasibly learnable,[1] while a poorly structured theory admitting a finite, but very large, number of possible grammars might not.

And indeed, a *principles-and-parameters* (P&P) *universal grammar* (UG) with n parameters admits at least 2^n grammars; more, if the parameters are not binary. Such exponential growth in the number of parameters quickly leads to spaces much too large to search exhaustively. An OT UG with N constraints admits $N!$ grammars, which grows still faster.

Thus to achieve meaningful assurance of learnability from our grammatical theory, we must seek evidence that the theory provides the space of possible grammars with the *kind of structure* that learning can effectively exploit.

Consider P&P theory in this regard. Two types of learnability research are useful as contrasts to the results we offer in this book. The first is *cue learning*, exemplified by work such as Dresher and Kaye 1990. These authors adopt a particular parameterized space of grammars, and analyze in great detail the relationships between the parameter settings and the forms overtly available to the learner. They propose a *specific* learning algorithm to make use of the structure provided by a *specific* P&P theory. Their analysis is entirely limited to their particular parametric system for metrical stress; a cue learning approach to a parametric grammar for some other component of linguistic theory, or even to an alternative parametric analysis of metrical stress, would essentially require starting over from scratch.

Another approach to learnability within P&P, quite different from cue learning, is represented in the work of Gibson and Wexler (1994) and Niyogi and Berwick (1996). The *triggering learning algorithm* (and its variations) is designed to learn grammars from data overtly available to the learner. Like those developed in our work, these algorithms apply to any instance of a very general class of systems: in their case, the class of P&P systems. Further, Niyogi and Berwick (1996) provide formal analysis of the algorithms. However, this work differs from ours in a direction representing the opposite extreme from cue learning: these learning

algorithms are *minimally informed* by the grammatical theory. For triggering learning algorithms treat the grammar only as a black box evaluating learning data as either grammatically analyzable or not; the algorithms either randomly flip grammar parameters in order to render an input analyzable (Gibson and Wexler's *Triggering Learning Algorithm*), or randomly flip parameters without regard to immediate resulting analyzability (which, Niyogi and Berwick argue, can actually outperform the Triggering Learning Algorithm). These learning algorithms are equally appropriate as procedures for learning parameterized grammars and as procedures for, say, training a neural network[2] (with binary weights) to classify radar images of submarines: if flipping a parameter (connection in the network) gives better classification of a submarine, flip it. These are simply generic search algorithms that employ no properties of the grammatical theory per se.

Further, the learnability results relating to triggering learning algorithms assume the existence of overt data that directly reveal individual parameter values. Such an assumption limits how impoverished the learning data can be and has unclear relevance to realistic grammars (see Frank and Kapur 1996); we discuss this further in section 6.1. Finally, regardless of the availability of such "triggering" forms, these algorithms offer little justification for confidence in their tractability. In fact, the only result regarding the time required for learning is that the probability of learning the correct grammar increases toward 1 as the number of learning instances approaches infinity[3]—leaving open the possibility of doing even worse than exhaustive search.

In sum, these two approaches to learnability analysis within P&P either (1) use grammatical structure in the learning algorithm, but the structure of a *particular* parametric system, or (2) develop general algorithms applicable to any P&P system, but algorithms *so* general they apply just as well to any nongrammatical parameterized system. This dichotomy of approaches is likely a consequence of the nature of P&P. A particular P&P system, like one for stress, has sufficient structure to inform a learning procedure (option 1). But as a general theory of how grammars may differ (as opposed to how stress systems may differ), P&P provides little structure for a learner to exploit beyond the existence of a finite space for searching. In particular, P&P theory per se provides no characteristically *grammatical* structure for a language learner to exploit.

But the situation in OT is quite different. This theory is reviewed in chapter 2, but the immediately relevant claims of OT are these:

(1.1) OT in a nutshell

- What is it that all languages have in common? *A set of constraints on well-formedness.*
- How may languages differ? *Only in which constraints have priority in case of conflict.*
- Language-particular relative constraint priorities are characterized by a *ranking* of the universal well-formedness constraints into a *dominance hierarchy*, with each constraint having absolute priority over all lower-ranked constraints.
- The grammar of a particular language—its constraint hierarchy—is an evaluator of structural descriptions, assigning a (nonnumerical) *Harmony* value that assesses the degree to which the constraints are met, taking into account the language-particular priorities. This provides the *harmonic ordering of forms*, ordering structural descriptions from maximal to minimal Harmony.
- The grammatical forms of the language are the *optimal* ones: the well-formed structural description of an input is the one with maximal Harmony.

Note that the constraints mentioned in (1.1) are the same in all languages: they contain no parameters. Unlike P&P, this is a theory of crosslinguistic variation with sufficient structure to enable grammatically informed learning algorithms independent of substantive grammatical assumptions.

(1.2) Main claim of this book: OT is a theory of UG that provides sufficient structure at the level of the grammatical framework itself to allow general but grammatically informed learning algorithms to be formally defined. Further, the efficiency of the algorithms can be argued to follow in large part from the formal structure of the grammatical framework.

The algorithms we develop are procedures for learning the priority ranking of constraints that, by (1.1), is all that distinguishes the grammar of a particular language. These are unquestionably grammar learning algorithms, not generic search algorithms.[4] Yet the structure that makes

these algorithms possible is not the structure of a theory of stress, nor a theory of phonology: it is the structure defining any OT grammar, that given in (1.1).

Of course, if a grammatically *un*informed learning algorithm, such as the Triggering Learning Algorithm, is desired, it can be obtained as easily in OT as in P&P; in fact, Pulleyblank and Turkel (1995, 1998) have already formulated and studied the *Constraint-Ranking Triggering Learning Algorithm*. Indeed, we can apply any of a number of generic search algorithms to the space of OT grammars—for example, Pulleyblank and Turkel (1995, 1998) have also applied the genetic algorithm to learning OT grammars. But unlike P&P, with OT we have an alternative to grammatically uninformed learning: learning algorithms specially constructed to exploit the structure provided by OT's theory of crosslinguistic variation.

1.3 Decomposing the Learning Problem

1.3.1 Grammar Learning and Robust Interpretive Parsing

To begin our analysis of grammar learning, we must distinguish the following three types of linguistic structure:

(1.3) The players in order of their appearance

- *Overt part of grammatical forms*: directly accessible to the learner
- *Full structural descriptions*: combine overt and nonovert ("hidden") structure
- *The grammar*: determines which structural descriptions are well formed

These three elements are all intimately connected, yet we propose to distinguish two subproblems, as schematically shown in figure 1.1.

(1.4) Decomposition of the problem

- *Robust interpretive parsing*: mapping the overt part of a form into a full structural description, complete with all hidden structure—given a grammar
- *Learning the grammar*—given a (robust) parser

(An interpretive parser is "robust" if it can parse an overt structure with a grammar, even when that structure is not grammatical according to the grammar. The importance of robustness will be discussed shortly.)

2 An Overview of Optimality Theory

This chapter presents the fundamental principles of OT. The defining reference is by Prince and Smolensky (1993) (abbreviated *P&S* here). Sections 2.1 and 2.2 provide the basics of the linguistic theory, while section 2.3 formulates the precise grammar learning problem posed by OT. Readers familiar with OT may wish to move directly to section 2.3.

We present the basics of OT as a series of general principles. To underscore the generality of the grammatical theory and our learnability analysis, we exemplify these principles with two running examples, one in phonology and one in syntax. The phonological example is the Basic CV Syllable Theory of P&S (chapter 6); we abbreviate this *CVT*. Our syntactic example is the theory of the distribution of clausal subjects proposed in Grimshaw and Samek-Lodovici 1995 (see also Samek-Lodovici 1994, 1996; Grimshaw and Samek-Lodovici 1998); we dub this theory *GSL*. Both examples will be used in chapter 3 to illustrate the Constraint Demotion learning procedure.

2.1 Constraints and Their Violation

Our starting point is a very basic principle.

(2.1) Grammars specify functions.
A grammar specifies a function that assigns to each input a structural description or output. (A grammar per se does not provide an algorithm for computing this function, e.g., by sequential derivation.)

In CVT, an input is a string of C's and V's—for example, /VCVC/. An output is a parse of the string into syllables, denoted as in (2.2).

(2.2) Examples of the input string /VCVC/ parsed into syllables
a. .V.CVC. = $[_\sigma$ V] $[_\sigma$ CVC]
b. ⟨V⟩.CV.⟨C⟩ = V $[_\sigma$ CV] C
c. ⟨V⟩.CV.C□. = V $[_\sigma$ CV] $[_\sigma$ C□]
d. .□V.CV.⟨C⟩ = $[_\sigma$ □V] $[_\sigma$ CV] C

Output (2.2a) is an onsetless open syllable followed by a closed syllable (periods denote the boundaries of syllables). Output (2.2b) contains only one, open, syllable. The initial V and final C of the input are not parsed into syllable structure, as notated by the angle brackets ⟨⟩. These segments exemplify *underparsing* and are not phonetically realized, so

(2.2b) is "pronounced" simply as .CV. The form .CV. is the *overt form* contained in (2.2b). Output (2.2c) consists of a pair of open syllables, in which the nucleus of the second syllable is not filled by an input segment. This empty nucleus is notated □ and exemplifies *overparsing*. The phonetic interpretation of this empty nucleus is an epenthetic vowel. Thus (2.2c) has .CV.CV. as its overt form. As in (2.2b), the initial V of the input is unparsed in (2.2c). Output (2.2d) is also a pair of open syllables (phonetically, .CV.CV.), but this time it is the onset of the first syllable that is unfilled (notated □; phonetically, an epenthetic consonant), while the final C is unparsed.

In Grimshaw and Samek-Lodovici's theory, GSL, an input is "a lexical head with a mapping of its argument structure into other lexical heads, plus a tense specification . . . as in Grimshaw (1997). The . . . input also specifies which arguments are foci, and which arguments are coreferent with the topic" (Grimshaw and Samek-Lodovici 1995:590). The example we will use is shown in (2.3); it represents the predicate *sing*, in the present perfect tense, with a masculine singular argument that is the current discourse topic.

An output in GSL is an X′ structure, a possible extended projection for the lexical head in the sense of Grimshaw 1990.

(2.3) Some outputs for the input $I \equiv$ ⟨sing(x), x = topic, x = he;Tense = present perfect⟩

a. $[_{IP}$ has [sung]]
b. $[_{IP}$ he$_i$ has [t$_i$ sung]]
c. $[_{IP}$ has [[t$_i$ sung] he$_i$]]
d. $[_{IP}$ it has [[t$_i$ sung] he$_i$]]

In the following discussion, these outputs will consistently be labeled (a)–(d) as in (2.3). Output (a) is a clause with no subject: the highest projection of the verb, labeled IP, has no Spec position. Output (b) has *he* in SpecIP, co-indexed with a trace in SpecVP. Output (c) has no SpecIP position, and *he* right-adjoined to VP, co-indexed with a trace in SpecVP; output (d) is the same, but with an expletive subject in SpecIP.

The second principle of OT is a prerequisite for the competition involved in the determination of optimality.

(2.4) *Gen*: UG provides a function *Gen* that, given any input I, generates *Gen*(I), the set of candidate structural descriptions for I.

The input I is an identified substructure contained within each of its candidate outputs in *Gen*(I). The domain of *Gen* implicitly defines the space of possible inputs.

In CVT, for any input I, the candidate outputs in *Gen*(I) consist of all possible parses of the string into syllables, including the possible over- and underparsing structures exemplified in (2.2b)–(2.2d). All syllables are assumed to contain a nucleus position, with optional preceding onset and following coda positions. CVT adopts the simplifying assumption (true of many languages) that the syllable position's onset and coda may each contain at most one C, and the nucleus position may contain at most one V. The four candidates of /VCVC/ in (2.2) are only illustrative of the full set *Gen*(/VCVC/). Since the possibilities of overparsing are unlimited, *Gen*(/VCVC/) in fact contains an infinite number of candidates.

For the syntactic input $I \equiv$ ⟨sing(x), . . .⟩ given in (2.3), *Gen*(I) includes the four X′-structure outputs (2.3a)–(2.3d), along with others such as the entirely empty *null parse*, Ø. Each structural description of I in *Gen*(I) should be understood to include I itself as a subpart, along with the output X′ structure. Following McCarthy and Prince 1995, we may assume that each structural description includes a *correspondence relation* linking the lexical heads in I with their correspondents in the output. Output (2.3a), $[_{IP}$ has [sung]], displays *underparsing*: an element of the input, x, has no correspondent in the output. Output (2.3d), $[_{IP}$ it has [[t$_i$ sung] he$_i$]], displays *overparsing*: an element of the output, *it*, has no correspondent in the input.

The next principle identifies the formal character of substantive grammatical principles.

(2.5) *Con*: UG provides a set *Con* of universal well-formedness constraints[1].

The constraints in *Con* evaluate the candidate outputs for a given input in parallel (i.e., simultaneously). Given a candidate output, each constraint assesses a multiset of *marks*, where each mark corresponds to one violation of the constraint. The collection of all marks assessed a candidate parse p is denoted *marks*(p). A mark assessed by a constraint C is denoted *C. A parse p_x is more marked than a parse p_y with respect to C if and only if C assesses more marks to p_x than to p_y. (The theory recognizes the notions more and less marked, but not absolute numerical levels of markedness.)

The CVT constraints are given in (2.6).

(2.6) Basic CV Syllable Theory constraints[2]
a. ONSET: Syllables have onsets.
b. NOCODA: Syllables do *not* have codas.
c. PARSE: Underlying (input) material is parsed into syllable structure.
d. FILL^{Nuc}: Nucleus positions are filled with underlying material.
e. FILL^{Ons}: Onset positions (when present) are filled with underlying material.

These constraints can be illustrated with the candidate outputs in (2.2a)–(2.2d). The marks incurred by these candidates are summarized in table 2.1. This is an OT *constraint tableau*. The competing candidates are shown in the left column. The other columns are for the universal constraints, each indicated by the label at the top of the column. Constraint violations are indicated with asterisks, one for each violation.

Candidate (2.2a) = .V.CVC. violates ONSET in its first syllable and NOCODA in its second; the remaining constraints are satisfied. The single mark that ONSET assesses .V.CVC. is denoted *ONSET. This candidate is a *faithful* parse: it involves neither underparsing nor overparsing, and therefore satisfies the *faithfulness* constraints PARSE and FILL[3]. By contrast, (2.2b) = ⟨V⟩.CV.⟨C⟩ violates PARSE, and more than once. This tableau will be further explained later.

The GSL constraints are given in (2.7) (Grimshaw and Samek-Lodovici 1995:590).

(2.7) Constraints of the GSL theory of subjects
a. SUBJ(ECT): The highest A-specifier in an extended projection must be filled (Grimshaw 1997).

Table 2.1
Constraint tableau for L_1

Candidates	ONSET	NOCODA	FILL^{Nuc}	PARSE	FILL^{Ons}
/VCVC/ →					
☞ (d) .□V.CV.⟨C⟩				*	*
(b) ⟨V⟩.CV.⟨C⟩				* *	
(c) ⟨V⟩.CV.C□.			*	*	
(a) .V.CVC.	*	*			

b. FULL-INT(ERPRETATION): Elements of the output must be interpreted (Grimshaw 1997).
c. DROP-TOP(IC): Arguments coreferent with the topic are structurally unrealized.
d. AL(IGN)-FOC(US): The left edge of a focused constituent is aligned with the right edge of a maximal projection.
e. PARSE: Input constituents are parsed (have a correspondent in the output).

These constraints can be illustrated with the candidate outputs in (2.3), as shown in table 2.2. (In all candidates, ALIGN-FOCUS is vacuously satisfied, because this input has no focus.) We can interpret PARSE and FULL-INTERPRETATION as members of the FAITHFULNESS family of constraints, which play the important role in OT of requiring that an output faithfully parse its input: each input element has one output correspondent with identical featural content, and vice versa. (Relative to OT phonology, the technical details of FAITHFULNESS in OT syntax are more obviously an open question for research. In phonology the "vocabulary" of the input and output are more nearly identical, so requiring one-to-one correspondence between input and output is more straightforward.)

2.2 Optimality and Harmonic Ordering

In OT, each underlying form is assigned a structural description, selected from the set of possible candidates. The selected candidate is, by definition, the grammatical candidate. The basis for this selection is the constraint violations assessed to each candidate. Intuitively, the grammatical candidate should be the one "least offensive" to the constraints. However, constraints can conflict, and the case shown in table 2.1, with

Table 2.2
Constraint violations in GSL

⟨sing(x), x = topic, x = he; T = pres perf⟩	PARSE	SUBJ	FULL-INT	DROP-TOP	AL-FOC
(b) $[_{IP}$ he$_i$ has $[t_i$ sung$]]$				*	
(d) $[_{IP}$ it has $[[t_i$ sung$]$ he$_i$ $]]$			*	*	
(c) $[_{IP}$ has $[[t_i$ sung$]$ he$_i$ $]]$		*		*	
(a) $[_{IP}$ has [sung$]]$	*	*			

every candidate violating at least one constraint, is by far the most common. Thus, the grammar needs a basis for resolving such conflicts.

OT gives a quite specific and restrictive theory of how constraint conflict is resolved. In a given language, different constraints are assigned different priority levels. When a choice must be made between satisfying one constraint or another, the stronger must take priority. The result is that the weaker will be violated in a grammatical structural description.

(2.8) *Constraint ranking*: A grammar *ranks* the universal constraints in a *dominance hierarchy*.

When one constraint $\mathbb{C}_1$ dominates another constraint $\mathbb{C}_2$ in the hierarchy, the relation is denoted $\mathbb{C}_1 \gg \mathbb{C}_2$. The ranking defining a grammar is total; the hierarchy determines the relative dominance of every pair of constraints: $\mathbb{C}_1 \gg \mathbb{C}_2 \gg \ldots \gg \mathbb{C}_n$.

(2.9) *Harmonic ordering (H-eval)*: A grammar's constraint ranking induces a harmonic ordering $\prec$ of all structural descriptions. Two candidate structural descriptions D_1 and D_2 are compared by identifying the highest-ranked constraint $\mathbb{C}_x$ with respect to which D_1 and D_2 are not equally marked. The candidate less marked with respect to $\mathbb{C}_x$ is the more harmonic, or the one with higher Harmony (with respect to the given ranking).

$D_1 \prec D_2$ denotes that D_1 is less harmonic than D_2. The harmonic ordering $\prec$ determines the relative Harmony of every pair of candidates. For a given input, the most harmonic of the candidate outputs provided by *Gen* is the optimal candidate: it is the one assigned to the input by the grammar. Only this optimal candidate is well formed (grammatical); all less harmonic candidates are ill formed.

Given the definition of grammaticality in terms of relative Harmony, along with the requirement that grammars be defined by total rankings of the constraints, there is only one possible way that more than one competing candidate can be simultaneously grammatical: both candidates must have identical constraint violations. Two candidates assessed exactly the same marks by all the constraints cannot be distinguished on the basis of *any* constraint ranking relation and will always be equally harmonic. If two (or more) candidates have equal Harmony, and both

are more harmonic than all the other candidates, the two candidates are both optimal, with the interpretation of free alternation. If candidates D_1 and D_2 have equal Harmony, that relationship is denoted $D_1 \sim D_2$. In practice, it is quite rare to have more than one optimal candidate for any given input.

A couple of properties are worth stressing. First, the harmonic evaluation of candidates is purely relative, with no significance attached to the absolute number or distribution of constraint violations assessed to a candidate. A candidate with 150 constraint violations is no less grammatical for it, provided it better satisfies the ranked constraints than any of its competitors. Second, the only comparisons directly relevant to the grammaticality of forms are those between the optimal candidate and its competitors. The relative Harmony, with respect to each other, of two suboptimal candidates is assigned no significant interpretation. One suboptimal competitor is not "closer to grammatical" than another even though it is more harmonic than the other.

A formulation of harmonic ordering that will prove quite useful for learning involves *Mark Cancelation*. Consider a pair of competing candidates D_a and D_b, with corresponding lists of violation marks *marks*(D_a) and *marks*(D_b). Mark Cancelation is a process applied to a pair of lists of marks: it cancels violation marks in common to the two lists. Thus, if a constraint $\mathbb{C}_x$ assesses one or more marks $*\mathbb{C}_x$ in both *marks*(D_a) and *marks*(D_b), an instance of $*\mathbb{C}_x$ is removed from each list, and the process is repeated until at most one of the lists still contains a mark $*\mathbb{C}_x$. (Note that if D_a and D_b are equally marked with respect to $\mathbb{C}_x$, the two lists contain equally many marks $*\mathbb{C}_x$, and all occurrences of $*\mathbb{C}_x$ are eventually removed.) The resulting lists of *uncanceled marks* are denoted *marks'* (D_a) and *marks'*(D_b). If a mark $*\mathbb{C}_x$ remains in the uncanceled mark list of D_a, then D_a is more marked with respect to $\mathbb{C}_x$. If the highest-ranked constraint assessing an uncanceled mark has a mark in *marks'*(D_a), then $D_a \prec D_b$: this is the definition of harmonic ordering $\prec$ in terms of Mark Cancelation. Mark Cancelation is indicated by "crossing out the marks" in the tableau in table 2.3: one mark *PARSE cancels between the candidates (d) and (b) of table 2.1, and one uncanceled mark *PARSE remains in *marks'*(b).

Defining grammaticality via harmonic ordering has an important consequence.

Table 2.3
Mark Cancelation

Candidates	ONSET	NOCODA	FILL^{Nuc}	PARSE	FILL^{Ons}
(d) .□V.CV.⟨C⟩				*~~*~~	*
(b) ⟨V⟩.CV.⟨C⟩				*~~*~~ *	

(2.10) *Minimal violation*: The grammatical candidate minimally violates the constraints, relative to the constraint ranking.

The constraints of UG are *violable*: they are potentially violated in well-formed structures. Such violation is *minimal*, however, in the sense that the grammatical candidate D for an input I will best satisfy a constraint $\mathbb{C}$, unless each candidate that fares better than D on $\mathbb{C}$ also fares worse than D on some constraint that is higher ranked than $\mathbb{C}$.

Harmonic ordering can be illustrated with CVT by reexamining the tableau in table 1.1 under the assumption that the universal constraints are ranked by a particular grammar, L_1, with the ranking given in (2.11).

(2.11) Constraint hierarchy for L_1:
ONSET ≫ NOCODA ≫ FILL^{Nuc} ≫ PARSE ≫ FILL^{Ons}

The constraints (and their columns) are ordered in table 2.1 left to right, reflecting the hierarchy in (2.11). The candidates in this tableau have been listed in harmonic order, from highest to lowest Harmony; the optimal candidate is marked manually[4]. Starting at the bottom of the tableau, (a) ≺ (c) can be verified as follows. The first step is to cancel common marks: here, there are none. The next step is to determine which candidate has the worst uncanceled mark—that is, most violates the most highly ranked constraint: it is (a), which violates ONSET. Therefore (a) is the less harmonic. In determining that (c) ≺ (b), first cancel the common mark *PARSE; (c) then earns the worst remaining mark of the two, *FILL^{Nuc}. When comparing (b) to (d), one *PARSE mark cancels, leaving *marks'*(b) = {*PARSE} and *marks'*(d) = {*FILL^{Ons}}. The worst mark is the uncanceled *PARSE incurred by (b), so (b) ≺ (d).

L_1 is a language in which all syllables have the overt form .CV.: onsets are required, codas are forbidden. In case of problematic inputs such as /VCVC/ where a faithful parse into CV syllables is not possible, this language uses overparsing (of consonants) to provide missing onsets for vowels, and underparsing (of consonants) to avoid codas (it is the language denoted $\Sigma^{CV}_{ep.del}$ in P&S section 6.2.2.2).

Table 2.4
Constraint tableau for L_2

Candidates	ONSET	NOCODA	FILL^{Ons}	PARSE	FILL^{Nuc}
/VCVC/ →					
☞ (c) ⟨V⟩.CV.CṍC.				*	*
(b) ⟨V⟩.CV.⟨C⟩				* *	
(d) .□V.CV.⟨C⟩			*	*	
(a) .V.CVC.	*	*			

Exchanging the two FILL constraints in L_1 gives the grammar L_2.

(2.12) Constraint hierarchy for L_2:
ONSET ≫ NOCODA ≫ FILL^{Ons} ≫ PARSE ≫ FILL^{Nuc}

Now the tableau corresponding to table 2.1 becomes table 2.4; the columns have been reordered to reflect the constraint reranking, and the candidates have been reordered to reflect the new harmonic ordering.

Like L_1, all syllables in L_2 are CV; /VCVC/ gets syllabified differently, however. In L_2, underparsing (of vowels) is used to avoid onsetless syllables, and overparsing (of vowels) to avoid codas (L_2 is P&S's language $\Sigma^{CV}_{del,ep}$).

The relation between L_1 and L_2 illustrates a principle of OT central to learnability concerns.

(2.13) *Typology by reranking*: Systematic crosslinguistic variation is due entirely to variation in language-specific total rankings of the universal constraints in *Con*. Analysis of the optimal forms arising from all possible total rankings of *Con* gives the typology of possible human languages. UG may impose restrictions on the possible rankings of *Con*.

Analysis of all rankings of the CVT constraints reveals a typology of basic CV syllable structures that explains Jakobson's typological generalizations (Jakobson 1962, Clements and Keyser 1983): see P&S chapter 6. In this typology, licit syllables may have required or optional onsets, and, independently, forbidden or optional codas.

These principles may also be illustrated with the GSL theory. To begin, Mark Cancelation is illustrated in table 2.5.

Harmonic ordering can be illustrated with GSL by reexamining the tableau in table 2.3, reproduced here as table 2.6, under the assumption that the universal constraints are ranked by a particular grammar of a language L_1' with the ranking given in (2.14). (This is a language like English with respect to the distribution of subjects.)

(2.14) Constraint hierarchy for (English-like) L_1':
PARSE ≫ SUBJ ≫ FULL-INT ≫ DROP-TOP ≫ AL-FOC

The constraints are ordered in table 2.6 left to right, reflecting the hierarchy in (2.14). The candidates in this tableau have been listed in harmonic order, from highest to lowest Harmony. Starting at the bottom of the tableau, (a) ≺ (c) can be verified as follows. The first step is to cancel common marks: here, *SUBJ. Then (c) has an uncanceled *DROP-TOP mark, *marks'*(c) = {*DROP-TOP}, while *a* has an uncanceled *PARSE mark, *marks'*(a) = {*PARSE}; so (a) is less harmonic. Next we verify that (c) ≺ (d): the uncanceled mark of (c), *SUBJ, is assessed by a constraint that is higher ranked in L_1 than that assessing the uncanceled mark of (d), FULL-INT. Finally, (d) ≺ (b) holds because (d) has an uncanceled mark while (b) does not.

As shown in the tableau in table 2.6, L_1 is a language in which unfocused topic-referring subjects are parsed into subject position (SpecIP). This English-like behavior changes to Italian-like behavior when the ranking of PARSE and SUBJ are lowered to their positions in the ranking defining language L_2'.

(2.15) Constraint hierarchy for (Italian-like) L_2':
FULL-INT ≫ DROP-TOP ≫ PARSE ≫ AL-FOC ≫ SUBJ

As shown in the tableau of table 2.7, now an unfocused topic-referring subject is not parsed.

Analysis of all rankings of the GSL constraints derives a typology of subject distribution relating the presence or absence of expletive subjects, the preverbal or postverbal positioning of focused subjects, and the presence or absence of topic-referring subjects (see Grimshaw and Samek-Lodovici 1995, 1998; Samek-Lodovici 1996).

A final central principle of OT is given in (2.16).

Table 2.5
Mark Cancelation

⟨sing(x), x = topic, x = he; T = pres perf⟩	PARSE	SUBJ	FULL-INT	DROP-TOP	AL-FOC
☞ (b) [IP hei has [ti sung]]				*	
(c) [IP has [[ti sung] hei]]		*		*	

Table 2.6
Constraint tableau for (English-like) L_1'

⟨sing(x), x=topic, x=he; T=pres perf⟩	PARSE	SUBJ	FULL-INT	DROP-TOP	AL-FOC
☞ (b) [IP hei has [ti sung]]				*	
(d) [IP it has [[ti sung] hei]]			*	*	
(c) [IP has [[ti sung] hei]]		*		*	
(a) [IP has [sung]]	*	*			

Table 2.7
Constraint tableau for (Italian-like) L_2'

⟨sing(x), x = topic, x = he; T = pres perf⟩	FULL-INT	DROP-TOP	PARSE	AL-FOC	SUBJ
☞ (a) [IP has [sung]]			*		*
(b) [IP hei has [ti sung]]		*			*
(c) [IP has [[ti sung] hei]]		*			*
(d) [IP it has [[ti sung] hei]]	*	*			*

(2.16) *Richness of the base*: The set of possible inputs to the grammars of all languages is the same. The grammatical inventories of languages are defined as the forms appearing in the outputs that emerge from the grammar when it is fed the universal set of all possible inputs (P&S section 9.3).

Systematic differences in inventories arise from different constraint rankings, not different inputs. The lexicon of a language is a sample from the inventory of possible inputs; all systematic properties of the lexicon arise indirectly from the grammar, that delimits the inventory from which the lexicon is drawn. There are no morpheme structure constraints on phonological inputs; no lexical parameter that determines whether a language has *pro*. In language L_1, all syllables have the form.CV. not because the possible *inputs* are restricted ahead of time to consist only of forms with strict CV alternation, but because the grammar so restricts the *outputs*. Richness of the base extends this style of explanation to all inventory phenomena. If any language lacks a particular structure, it is because any input containing the structure will have, as its optimal candidate, a structural description with an output that does not contain that structure: the optimal candidate will always change it into something else.

As our last issue concerning OT fundamentals, we return to the question of infinity. In the CVT, and quite typically in OT phonology, at least, *Gen*(*I*) contains an infinite number of candidate structural descriptions of each input *I*. In the face of this infinity, is the theory well defined? Of course, the overwhelming majority of formal systems in mathematics involve an infinity of structures; the mere fact of infinity means only that the most primitive conceivable method, listing all the possibilities and checking each one, is infeasible. But even in finite cases, this method is commonly infeasible anyway. For an OT grammar to be well defined, it must be that for any input, which structure is optimal is formally determinate. The necessary formal definitions are provided in P&S chapter 5. To show that a given structure is the optimal parse of *I*, we need to provide a proof that none of the (infinitely many) other parses in *Gen*(*I*) has higher Harmony. A general technique for such demonstration, the *Method of Mark Eliminability* (P&S section 7.3), proceeds by showing that any attempt to avoid the marks incurred by the putatively optimal output leads to alternatives that incur worse marks.[5]

Thus the infinite candidate set has a perfectly well-defined optimum (or optima, if multiple outputs incur exactly the same, optimal, set of marks). Yet it might still be the case that the task of actually computing the optimal candidate cannot be performed efficiently. But as Tesar (1995, 1996) has shown, computational feasibility is not a problem either, at least in the general cases studied to date. One reason is that the infinity of candidates derives from the unbounded potential for empty structure. But empty structure is always penalized by constraints of the FILL family: these militate against empty syllable positions in phonology (FILL^{Ons}, FILL^{Nuc}), empty X^0 positions in syntax (OBLIGATORY-HEADS of Grimshaw 1993, 1997), uninterpretable elements (FULL-INT), and the like. Optimal structures may have empty structure, in violation of FILL, only when that is necessary to avoid violation of higher-ranking constraints. This will not be the case for unbounded quantities of empty structure. It follows that finite inputs will only have a finite number of structural descriptions that are potentially optimal, under some constraint ranking. Thus a parser constructing an optimal parse of a given input *I* need only have access to a finite part of the infinite space *Gen*(*I*).

The parsing algorithms developed by Tesar construct optimal parses from increasingly large portions of the input, requiring an amount of computational time and storage space that grows with the size of the input only as fast as for parsers of conventional, rewrite-rule grammars of corresponding complexity. The structure in the space of candidates allows for efficient computation of optimal parses, even though the grammar's specification of well-formedness makes reference to an infinite set of parses.

2.3 The Grammar Learning Problem

Having provided the necessary principles, we can now insert the crucial grammatical structure of OT into the learning problem schematically formulated in section 1.3.

(2.17) Our grammar learning problem (including relevant grammatical structure).

Given: Learning data in the form of full grammatical structural descriptions.

The universal components of any OT grammar:

- The set of possible inputs
- The function *Gen* generating the candidate outputs for any possible input
- The constraints *Con* on well-formedness

Find: a language-particular OT grammar, consisting of a ranking (or set of rankings) of the constraints in *Con*, consistent with all the given data.

The initial data for the learning problem are well-formed candidate structural descriptions; each consists of an input together with the output declared optimal by the target grammar. For example, the learner of the CV language L_1 might have as an initial datum .□V.CV.⟨C⟩, candidate (d) of table 2.1, the parse assigned to the input I = /VCVC/; the learner of the Italian-like language L_2' might have as an initial datum the input I = ⟨sing(x), x = topic, x = he; T = pres perf⟩ together with its grammatical parse, $p = [_{IP}$ has [sung]] of table 2.7(a).

3 Constraint Demotion

3.1 The Principle of Constraint Demotion

Optimality Theory is inherently comparative; the grammaticality of a structural description is determined not in isolation, but with respect to competing candidates. Therefore, the learner is not informed about the correct ranking by positive data in isolation; the role of the competing candidates must be addressed. This fact is not a liability, but an advantage: a comparative theory gives comparative structure to be exploited. Each piece of positive evidence, a grammatical structural description, brings with it a body of implicit negative evidence in the form of the competing descriptions. Given access to *Gen* (which is universal) and the underlying form (contained in the given structural description), the learner has access to these competitors. Any competing candidate, along with the grammatical structure, determines a data pair related to the correct ranking: the correct ranking must make the grammatical structure more harmonic than the ungrammatical competitor.

This can be stated more concretely in the context of Basic CV Syllable Theory. Suppose the learner receives a piece of explicit positive evidence like p = .□V.CV.⟨C⟩. Now consider any other parse of p's input I = /VCVC/; e.g., p' = .V.CVC. In the general case, there are two possibilities. Either an alternative parse p' has exactly the same marks as p, in which case p' has the same Harmony as p (no matter what the unknown ranking) and must be tied for optimality: p' too then is a grammatical parse of I. This case is unusual, but possible. In the typical case, a competitor p' and p will not have identical marks. In this case the harmonic ordering of forms determined by the unknown ranking will declare one more harmonic than the other; it must be p that is the more harmonic, since it is given as well-formed learning data and is thus optimal.

For each well-formed example p a learner receives, therefore, every other parse p' of the same input must be suboptimal—that is, ill formed—unless p' happens to have exactly the same marks as p. Thus a single positive example, a parse p of an input I, conveys a body of implicit negative evidence: all the other parses p' in $Gen(I)$—with the exception of those parses that the learner can recognize as tied for optimality with p in virtue of having the same marks.

In our CV example, a learner given the positive datum p = .□V.CV.⟨C⟩ knows that, with respect to the unknown constraint hierarchy of the

language being learned, the alternative parse of the same input, p' = .V.CVC., is less harmonic:

(3.1) .V.CVC. ≺ .□V.CV.⟨C⟩

Furthermore, corresponding harmonic comparisons must hold for every other parse p'' in *Gen*(/VCVC/).

The implicit negative evidence provided by the structure of OT can also be illustrated with the GSL theory. Suppose the learner receives a piece of explicit positive evidence such as the form: p = ⟨sing(x), x = topic, x = he; T = pres perf⟩; [$_{IP}$ has [sung]]. (Recall that in OT, full structural descriptions consist of an "input", an "output", and a correspondence between their elements. This example informs the learner that an unfocused, topic-referring subject is not overtly realized in the target language.) Now consider any other parse p' of p's input I = ⟨sing(x), x = topic, x = he; T = pres perf⟩; e.g., the parse p' with output [$_{IP}$ he$_i$ has [t$_i$ sung]]. Having received the positive datum p, the learner knows that, with respect to the unknown constraint hierarchy of the language being learned, the alternative parse of the same input, p', is less harmonic:

(3.2) for I = ⟨sing(x), x = topic, x = he; T = pres perf⟩, [$_{IP}$ he$_i$ has [t$_i$ sung]] ≺ [$_{IP}$ has [sung]].

Thus each single piece of positive initial data conveys a large amount of inferred comparative data of the form outlined in (3.3).

(3.3) [suboptimal parse of input I, "*loser*"] ≺ [optimal parse of input I, "*winner*"]

Such pairs are what feed our learning algorithm. Each pair carries the information that the constraints violated by the suboptimal parse loser must outrank those violated by the optimal parse winner. That is, in some sense, we must have *marks*(*loser*) ≫ *marks*(*winner*).

(3.4) The key: *loser-marks* ≫ *winner-marks*

The learning procedure we now develop is nothing but a way of making this observation precise and deducing its consequences. The challenge faced by the learner is: given a suitable set of such *loser/winner pairs*, to find a ranking such that each *winner* is more harmonic than its corresponding *loser*. Constraint Demotion solves this challenge, by

demoting the constraints violated by the winner down in the hierarchy so that they are dominated by the constraints violated by the loser.

3.1.1 The Basic Idea

In our CV language L_1, the winner for input /VCVC/ is .□V.CV.⟨C⟩. Table 2.1 gave the marks incurred by the winner (labeled (d)) and by three competing losers. These may be used to form three *loser/winner* pairs, as shown in table 3.1. A *mark-data pair* is the paired lists of constraint violation marks for a *loser/winner* pair.

To make contact with more familiar OT constraint tableaux, the information in table 3.1 will also be displayed in the format of table 3.2.

At this point, the constraints are unranked; the dotted vertical lines separating constraints in table 3.2 convey that no relative ranking of adjacent constraints is intended. The winner is indicated with a ✓; ☞ will denote the structure that is optimal according to the current grammar, which may not be the same as the winner (the structure that is grammatical in the target language). The constraint violations of the winner,

Table 3.1
Mark-data pairs (L_1)

	loser	≺	*winner*	*marks*(*loser*)	*marks*(*winner*)
(a) ≺ (d)	.V.CVC.	≺	.□V.CV.⟨C⟩	*ONSET *NOCODA	*PARSE *FILL^{Ons}
(b) ≺ (d)	⟨V⟩.CV.⟨C⟩	≺	.□V.CV.⟨C⟩	*PARSE *PARSE	*PARSE *FILL^{Ons}
(c) ≺ (d)	⟨V⟩.CV.C□.	≺	.□V.CV.⟨C⟩	*PARSE *FILL^{Nuc}	*PARSE *FILL^{Ons}

Table 3.2
Initial data

		not-yet-ranked				
loser/winner pairs		FILL^{Nuc}	FILL^{Ons}	PARSE	ONSET	NOCODA
(d) ✓	.□V.CV.⟨C⟩		⊛	⊛		
(a)	.V.CVC.				*	*
(d) ✓	.□V.CV.⟨C⟩		⊛	⊛		
(b)	⟨V⟩.CV.⟨C⟩			*	*	
(d) ✓	.□V.CV.⟨C⟩		⊛	⊛		
(c)	⟨V⟩.CV.C □.	*		*		

Table 3.3
Mark-data pairs after cancelation (L_1)

loser/winner pairs				*marks'(loser)*	*marks'(winner)*
(a) ≺ (d)	.V.CVC.	≺	.□V.CV.⟨C⟩	*ONSET *NOCODA	*PARSE *FILL^{Ons}
(b) ≺ (d)	⟨V⟩.CV.⟨C⟩	≺	.□V.CV.⟨C⟩	~~*PARSE~~ *PARSE	~~*PARSE~~ *FILL^{Ons}
(c) ≺ (d)	⟨V⟩.CV.C□.	≺	.□V.CV.⟨C⟩	~~*PARSE~~ *FILL^{Nuc}	~~*PARSE~~ *FILL^{Ons}

marks(*winner*), are distinguished by the symbol ⊛. Mark cancelation is denoted by diagonal crossing, as in table 2.3.

In order that each loser be less harmonic than the winner, the marks incurred by the former, *marks*(*loser*), must collectively be worse than *marks*(*winner*). According to (2.9), what this means more precisely is that *loser* must incur the worst uncanceled mark, compared to *winner*. This requires that uncanceled marks be identified, so the first step is to cancel the common marks in table 3.1, as shown in table 3.3.

The canceled marks have been ~~struck-out~~. Note that the cancelation operation that transforms *marks* to *marks'* is defined only on pairs of sets of marks—for example, *PARSE is canceled in the pairs (b) ≺ (d) and (c) ≺ (d), but not in the pair (a) ≺ (d). Note also that cancelation of marks is done token by token: in the row (b) ≺ (d), one but not the other mark *PARSE in *marks*(b) is canceled.

The mark-data after cancelation are the data on which Constraint Demotion operates. The representation in tableau form, given in table 3.2, reveals what successful learning must accomplish: the ranking of the constraints must be adjusted so that, for each pair, all the uncanceled winner marks ⊛ are dominated by at least one loser mark *. Using the standard tableau convention of positioning the highest-ranked constraints to the left, the columns containing uncanceled ⊛ (winner marks) need to be moved far enough to the right (down in the hierarchy) so that, for each pair, there is a column (constraint) containing an uncanceled * (loser mark) further to the left (dominant in the hierarchy) than all the columns containing uncanceled ⊛ (winner marks).

The algorithm to accomplish this is based on the principle in (3.5).

(3.5) *Principle of Constraint Demotion*: For any constraint $\mathbb{C}$ assessing an uncanceled winner mark, if $\mathbb{C}$ is not dominated by a constraint assessing an uncanceled loser mark, demote $\mathbb{C}$ to immediately below the highest-ranked constraint assessing an uncanceled loser mark.

Constraint Demotion works by demoting the constraints with uncanceled winner marks down far enough in the hierarchy so that they are dominated by a constraint with an uncanceled loser mark, ensuring that each winner is more harmonic than its competing losers.

Notice that it is not necessary for *all* uncanceled loser marks to dominate all uncanceled winner marks: one will suffice. However, given more than one uncanceled loser mark, it is often not immediately apparent which one needs to dominate the uncanceled winner marks (the pair (a) ≺ (d) above is such a case). This is the challenge successfully overcome by Constraint Demotion.

3.1.2 Stratified Domination Hierarchies

OT grammars are defined by rankings in which the domination relation between any two constraints is specified. The learning algorithm, however, works with a larger space of hypotheses, the space of stratified hierarchies. A stratified domination hierarchy has the form in (3.6).

(3.6) Stratified domination hierarchy
$\{\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_3\} \gg \{\mathbb{C}_4, \mathbb{C}_5, \ldots, \mathbb{C}_6\} \gg \ldots \gg \{\mathbb{C}_7, \mathbb{C}_8, \ldots, \mathbb{C}_9\}$

The constraints $\mathbb{C}_1, \mathbb{C}_2, \ldots, \mathbb{C}_3$ comprise the first stratum in the hierarchy: they are not ranked with respect to one another, but they each dominate all the remaining constraints. Similarly, the constraints $\mathbb{C}_4$, $\mathbb{C}_5$, $\ldots$, $\mathbb{C}_6$ comprise the second stratum: they are not ranked with respect to one another, but they each dominate all the constraints in the lower strata. In tableaux, strata will be separated from each other by solid vertical lines, while constraints within the same stratum will be separated by dotted lines, with no relative ranking implied.

The original notion of constraint ranking, in which a domination relation is specified for every pair of candidates, can now be seen as a special case of the stratified hierarchy, where each stratum contains exactly one constraint. That special case will be labeled here a total ranking. Henceforth, "hierarchy" will mean stratified hierarchy; when appropriate, hierarchies will be explicitly qualified as "totally ranked".

The definition of harmonic ordering (2.9) needs to be elaborated slightly for stratified hierarchies. When $\mathbb{C}_1$ and $\mathbb{C}_2$ are in the same

Table 3.4
Harmonic ordering with a stratified hierarchy: $\mathbb{C}_1 \gg \{\mathbb{C}_2, \mathbb{C}_3\} \gg \mathbb{C}_4$

	$\mathbb{C}_1$	$\mathbb{C}_2$	$\mathbb{C}_3$	$\mathbb{C}_4$
p_1	*!		*	
p_2			*	*!
☞ p_3		*		
p_4			* *!	

stratum, two marks $*\mathbb{C}_1$ and $*\mathbb{C}_2$ are equally weighted in the computation of Harmony. In effect, all constraints in a single stratum are collapsed together, and treated as though they were a single constraint, for the purposes of determining the relative Harmony of candidates. Minimal violation with respect to a stratum is determined by the candidate incurring the smallest sum of violations assessed by all constraints in the stratum. The tableau in table 3.4 gives a simple illustration.

Here, all candidates are compared to the optimal one, p_3. In this illustration, parses p_2 and p_3 violate different constraints, which are in the same stratum of the hierarchy. Therefore, these marks cannot decide between the candidates, and it is left to the lower-ranked constraint to decide in favor of p_3. Notice that candidate p_4 is still eliminated by the middle stratum because it incurs more than the minimal number of marks to constraints in the middle stratum. (The symbol *! indicates a mark fatal in comparison with the optimal parse.)

With respect to the comparison of candidates, marks assessed by different constraints in the same stratum can be thought of as "canceling", because they do not decide between the candidates. It is crucial, though, that the marks not be canceled for the purposes of learning. The term *Mark Cancelation*, as used in the rest of this book, should be understood to only cancel marks assessed by the same constraint to competing candidates; this is valid independent of the target constraint hierarchy, which, during learning, is unknown.

3.1.3 An Example: Basic CV Syllable Theory

Constraint Demotion (abbreviated CD) will now be illustrated using CVT—specifically, with the target language L_1 of table 2.1 and (2.11). The initial stratified hierarchy is set to

(3.7) $\mathcal{H} = \mathcal{H}_0 = \{\text{FILL}^{\text{Nuc}}, \text{FILL}^{\text{Ons}}, \text{PARSE}, \text{ONSET}, \text{NOCODA}\}$

Suppose that the first loser/winner pair is (b) $\prec$ (d) of table 3.1. Mark Cancelation is applied to the corresponding pair of mark lists, resulting in the mark-data pair shown in table 3.5.

Now CD can be applied. The highest ranked (in $\mathcal{H}$) uncanceled loser mark—the only one—is *PARSE. The *marks'(winner)* are checked to see if they are dominated by *PARSE. The only winner mark is $*\text{FILL}^{\text{Ons}}$, which is *not* so dominated. CD therefore calls for demoting FILL^{Ons} to the stratum immediately below PARSE. Since no such stratum currently exists, it is created. The resulting hierarchy is (3.8).

(3.8) $\mathcal{H} = \{\text{FILL}^{\text{Nuc}}, \text{PARSE}, \text{ONSET}, \text{NOCODA}\} \gg \{\text{FILL}^{\text{Ons}}\}$

This demotion is shown in tableau form in table 3.6; recall that strata are separated by solid vertical lines, whereas dotted vertical lines separate constraints in the same stratum. The uncanceled winner mark ⊛ is demoted to a (new) stratum immediately below the stratum containing the highest uncanceled winner mark *, which now becomes a fatal violation *! rendering irrelevant the dominated violation (which is therefore grayed out).

Now another loser/winner pair is selected. Suppose this is (a) $\prec$ (d) of table 3.1, as shown in table 3.7.

Table 3.5
Mark-data pair, step 1 (L_1)

	loser	$\prec$	*winner*	*marks'(loser)*	*marks'(winner)*
(b) $\prec$ (d)	⟨V⟩.CV.⟨C⟩	$\prec$	.□V.CV.⟨C⟩	~~*PARSE~~ *PARSE	~~*PARSE~~ $*\text{FILL}^{\text{Ons}}$

Table 3.6
First demotion

loser/winner pair	FILL^{Nuc}	FILL^{Ons}	PARSE	ONSET	NOCODA	FILL^{Ons}
(d) ✓☞ .□V.CV.⟨C⟩		⊛	⊛			⊛
(b) ⟨V⟩.CV.⟨C⟩			*!			

Table 3.7
Mark-data pair for CD, step 2 (L_1)

	loser	≺	*winner*	*marks'(loser)*	*marks'(winner)*
(a) ≺ (d)	.V.CVC.	≺	.□V.CV.⟨C⟩	*Onset *NoCoda	*Parse *Fill^{Ons}

Table 3.8
Second demotion

loser/winner pair	Fill^{Nuc}	Parse	Onset	NoCoda	Fill^{Ons}	Parse
(d) ✓☞ .□V.CV.⟨C⟩		⊛			⊛	⊛
(a) .V.CVC.			*!	*!		

Table 3.9
Mark-data pair for CD, step 3 (L_1)

	loser	≺	*winner*	*marks'(loser)*	*marks'(winner)*
(c) ≺ (d)	⟨V⟩.CV.CV́.	≺	.□V.CV.⟨C⟩	~~*Parse~~ *Fill^{Nuc}	~~*Parse~~ *Fill^{Ons}

There are no common marks to cancel. CD calls for finding the highest-ranked of the *marks'(loser)*. Since Onset and NoCoda are both top ranked, either will do; choose, say, Onset. Next, each constraint with a mark in *marks'(winner)* is checked to see if it is dominated by Onset. Fill^{Ons} is so dominated. Parse is not, however, so it is demoted to the stratum immediately below that of Onset.

(3.9) $\mathcal{H}$ = {Fill^{Ons}, Onset, NoCoda} ≫ {Fill^{Ons}, Parse}

In tableau form, this demotion is shown in table 3.8. (Both the Onset and NoCoda violations are marked as fatal, *!, because both are highest-ranking violations of the loser: they belong to the same stratum.)

Suppose now that the next *loser/winner* pair is as shown in table 3.9.

Since the uncanceled loser mark, *Fill^{Nuc}, already dominates the uncanceled winner mark, *Fill^{Ons}, no demotion results, and $\mathcal{H}$ is unchanged. This is an example of an *uninformative* pair, given its location in the sequence of training pairs: no demotions result.

Table 3.10
Mark-data pair for CD, step 4 (L_1)

	loser	≺	*winner*	*marks'(loser)*	*marks'(winner)*
	⟨VC⟩	≺	.□V.⟨C⟩	~~*Parse~~ *Parse	~~*Parse~~ *Fill^{Ons}

Table 3.11
Third demotion

loser/winner pair	Fill^{Nuc}	Onset	NoCoda	Fill^{Ons}	Parse	Fill^{Ons}
✓☞ .□V.⟨C⟩				⊛	⊛	⊛
⟨VC⟩					*!	

Suppose the next *loser/winner* pair results from a new input, /VC/, with a new optimal parse, .□V.⟨C⟩, as shown in table 3.10.

Since the winner mark * Fill^{Ons} is not dominated by the loser mark *Parse, it must be demoted to the stratum immediately below Parse, resulting in the hierarchy in (3.10).

(3.10) $\mathcal{H}$ = {Fill^{Nuc}, Onset, NoCoda} ≫ {Parse} ≫ {Fill^{Ons}}

This demotion is shown in table 3.11.

This stratified hierarchy generates precisely L_1, using the interpretation of stratified hierarchies described above. For any further *loser/winner* pairs that could be considered, *loser* is guaranteed to have at least one uncanceled mark assessed by a constraint dominating all the constraints assessing uncanceled marks to *winner*. Thus, no further data will be informative: L_1 has been learned.

A parallel example could be developed using the GSL theory. An example of a demotion step resulting from the loser/winner pair of (27) is shown in table 3.12. Prior to demotion, all constraints are unranked. With respect to this stratified hierarchy, the loser (b) is more harmonic than the desired winner (a), an error. The two uncanceled winner marks (assessed by Subj and Parse) must be demoted to a stratum just below that of the sole loser mark, *Drop-Top. This requires creation of such a stratum; after demotion, the learner's stratified hierarchy has two strata (separated by the heavy vertical line).

Table 3.12
Constraint Demotion for (Italian-like) L'_2

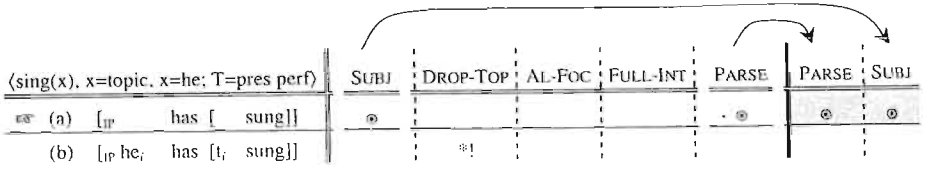


Table 3.13
The disjunction problem

loser/winner pair		FILL^{Ons}	ONSET	FILL^{Nuc}	NOCODA	PARSE
(a)	V.CVC.		*		*	
(d) ☞	.□V.CV.⟨C⟩	⊛				⊛

3.1.4 Why Not Constraint Promotion?

Constraint Demotion is defined entirely in terms of *demotion*; all movement of constraints is downward in the hierarchy. One could reasonably ask if this is an arbitrary choice; couldn't the learner just as easily promote constraints toward the correct hierarchy? The answer is no, and understanding why reveals the logic behind Constraint Demotion.

Consider the tableau shown in table 3.13, with (d) the winner and (a) the loser. The ranking depicted in the tableau makes the loser, (a), more harmonic than the winner, (d), so the learner needs to change the hierarchy to achieve the desired result, (a) $\prec$ (d).

There are no marks in common, so no marks are canceled. For the winner to be more harmonic than the loser, at least one of the loser's marks must dominate all the winner's marks. This relation is expressed in (3.11).

(3.11) (ONSET or NOCODA) $\gg$ (FILL^{Ons} and PARSE)

Demotion moves the constraints corresponding to the winner's marks. They are contained in a conjunction (*and*); thus, once the highest-ranked loser mark is identified, *all* the winner marks need to be dominated by it, so all constraints with winner marks are demoted if not already so dominated. A hypothetical *promotion* operation would move the constraints corresponding to the *loser's* marks up in the hierarchy. But notice that the loser's marks are contained in a *disjunction* (*or*). It is not clear which of the loser's violations should be promoted; perhaps all of them, or perhaps just one. Other data might require one of the constraints violated by the loser to be dominated by one of the constraints violated by the winner. This *loser/winner* pair gives no basis for choosing.

Disjunctions are notoriously problematic in general computational learning theory. Constraint Demotion solves the problem of disentangling the disjunctions by demoting the constraints violated by the winner; there is no choice to be made among them—all must be dominated. The choice between the constraints violated by the loser is made by picking the one highest ranked in the current hierarchy (in table 3.13, that is ONSET). Thus, if other data have already determined that ONSET $\gg$ NOCODA, that relationship is preserved. The constraints violated by the winner are only demoted as far as necessary.

3.2 Analysis of Constraint Demotion

3.2.1 Learnability Results: Convergence and Efficiency

The illustration of Constraint Demotion given in section 3.1.3 started with initial hierarchy $\mathcal{H}_0$, given in (3.7), having all the constraints in one stratum. Using this initial hierarchy is convenient for demonstrating some formal properties. By starting with all constraints at the top, CD can be understood to demote constraints down toward their correct position. Because CD only demotes constraints as far as necessary, a constraint never gets demoted below its target position, and will not be demoted further once reaching its target position. The formal analysis that assumes $\mathcal{H}_0$ as the initial hierarchy proves the following results.

(3.12) THEOREM Correctness of Constraint Demotion (initial hierarchy $\mathcal{H}_0$)
Starting with all constraints in *Con* ranked in the top stratum, and applying Constraint Demotion to informative positive evidence as long as such exists, the process converges on a stratified hierarchy such that all totally ranked refinements of that hierarchy correctly account for the learning data.

(3.13) THEOREM Data complexity of Constraint Demotion (initial hierarchy $\mathcal{H}_0$)
Starting with all constraints in *Con* ranked in the top stratum, the number of informative data pairs required for learning is at most $N(N-1)/2$, where N is the number of constraints in *Con*.

The data complexity of a learning algorithm is the amount of data that needs to be supplied to the algorithm to ensure that it learns the correct grammar. For Constraint Demotion, each informative data pair results in a demotion, and the convergence results ensure that each demotion brings the hypothesized grammar ever closer to the correct grammar. Therefore, it is convenient to measure data complexity in terms of the maximum number of informative data pairs needed before the correct grammar is reached.

In Constraint Demotion, an informative pair can result in the demotion of one or several constraints, each being demoted down one or more strata. The minimum amount of progress resulting from a single error is the demotion of one constraint down one stratum. The worst-case data complexity thus amounts to the maximum distance between a possible starting hierarchy and a possible target hierarchy to be learned, where the distance between the two hierarchies is measured in terms of one-stratum demotions of constraints. The maximum possible distance between the initial hierarchy $\mathcal{H}_0$ and a target hierarchy is $N(N-1)/2$, where N is the number of constraints in the grammar; this then is the maximum number of informative data pairs needed to learn the correct hierarchy.

The significance of this result is perhaps best illustrated by comparing it to the number of possible grammars. Given that any target grammar is consistent with at least one total ranking of the constraints, the number of possible grammars is potentially as large as the number of possible total rankings, $N!$. This number grows very quickly as a function of the number of constraints N, and if the amount of data required for learning scaled with the number of possible total rankings, it would be cause for concern indeed. Fortunately, the data complexity of CD is quite reasonable in its scaling. In fact, it does not take many universal constraints to give a drastic difference between the data complexity of CD and the number of total rankings: when $N = 10$, the CD data complexity is 45, while the number of total rankings is over 3.6 million. With 20 constraints, the CD data complexity is 190, while the number of total rankings is over 2 billion billion (2.43×10^{18}). This reveals the restrictiveness of the structure imposed by OT on the space of grammars: a learner can efficiently home in on any target grammar, managing an explosively sized grammar space with quite modest data requirements by fully exploiting the inherent structure provided by strict domination.

The power provided by strict domination for learning can be further underscored by considering that CD uses as its working hypothesis space not the space of total rankings, but the space of all stratified hierarchies, which is much larger and contains all total rankings as a subset. The disparity between the size of the working hypothesis space and the actual data requirements is that much greater.

As argued in chapter 1, the number of grammars made available by a grammatical framework is a rather crude measure of its explanatory power. A more significant measure is the degree to which the *structure* of UG allows rich grammars to be learned with realistically few positive examples. The crude number-of-grammars measure may be the best one can do given a theory of UG that does not enable the better learnability measure to be determined. In OT, however, we do have a quantitative and formally justified measure of learnability available in our $N(N-1)/2$ limit on the number of informative examples needed to solve our grammar learning problem. And we can see precisely how large the discrepancy can be between the number of grammars made available by a UG and the efficiency of learning that its structure enables.

This dramatic difference between the size of the OT grammar space and the number of informative examples needed to learn a grammar is due to the well-structured character of the space of fully ranked constraint hierarchies. It is useful to consider a set of parameters in the grammar space that suffice to specify the $N!$ grammars: these parameters state, for each pair of different constraints $\mathbb{C}_i$ and $\mathbb{C}_j$, which is dominant—that is, whether $\mathbb{C}_i \gg \mathbb{C}_j$ or $\mathbb{C}_j \gg \mathbb{C}_i$. There are in fact $N(N-1)/2$ such dominance parameters,[1] and this is the maximum number of informative examples needed to learn a correct hierarchy when starting from the $\mathcal{H}_0$ initial hierarchy.[2] Efficient learning via Constraint Demotion is possible because the enlarged hypothesis space allows these dominance parameters to be unspecified (in the initial state, they are *all* unspecified), and because evidence for adjusting these dominance parameters can be assessed independently (via the key idea (3.4): *loser-marks* $\gg$ *winner-marks*). A single adjustment may not irrevocably set a correct value for any dominance parameter, each adjustment brings the

hierarchy closer to the target, and eventually the adjustments are guaranteed to produce a correct set of parameter values. Note that what is independently adjustable here is not the substantive *content* of individual grammatical principles: it is the *interaction* of the principles, as determined by their relative rankings.

3.2.2 Arbitrary Initial Hierarchies

While the use of initial hierarchy $\mathcal{H}_0$ is convenient for purposes of illustration, it is by no means necessary for the success of Constraint Demotion. The formal proof of the correctness of Constraint Demotion can be extended to arbitrary initial constraint hierarchies, without any change whatsoever to the algorithm itself. Further, no significant change in the data complexity occurs: it is still a quadratic function of the number of constraints.

(3.14) THEOREM Correctness of Constraint Demotion
Starting with an arbitrary constraint hierarchy, and applying Constraint Demotion to informative positive evidence as long as such exists, the process converges on a stratified hierarchy such that all totally ranked refinements of that hierarchy correctly account for the learning data.

(3.15) THEOREM Data complexity of Constraint Demotion
Starting with an arbitrary constraint hierarchy, the number of informative data pairs required for learning is no more than $N(N-1)$, where N is the number of constraints in *Con*.

In fact, this upper bound is a significant overestimate, even more so than the one given for the special case with initial hierarchy $\mathcal{H}_0$.

With arbitrary initial hierarchies, CD can lead to empty strata; this can be seen as follows. Because the data observed must all be consistent with some total ranking, there is at least one constraint never assessing an uncanceled winner mark: the constraint top ranked in the total ranking. It is possible to have more than one such constraint (there are three for L_1); there will always be at least one. These constraints will never be demoted for any loser/winner pair, because only constraints assessing uncanceled winner marks for some loser/winner pair get demoted. Therefore, these constraints will stay put, no matter where they are in the initial hierarchy. If $\mathcal{H}_0$ is used, these constraints start at the top and stay there. For other initial hierarchies, these constraints stay put, and the other constraints eventually get demoted below them. This may leave some empty strata at the top, but that is of no consequence; all that matters is the relative position of the strata containing constraints.

Another learnability result now easily follows as well. In OT, a standard treatment of markedness scales is to posit in UG that certain constraints are universally ranked in a particular subhierarchy. For example, in P&S chapter 9, the markedness scale of place of articulation, according to which Coronal is less marked than, for example, Labial, is achieved via the UG requirement that the constraints violated by Cor(onal) and Lab(ial) PLace are universally ranked as in (3.16).

(3.16) Coronal unmarkedness universal subhierarchy
*PL/Lab ≫ *PL/Cor

In syntax, Legendre and others (Legendre et al. 1995; Legendre, Smolensky, and Wilson 1998) have proposed a universal hierarchy MINLINK, which realizes the "Shortest Link" principle. We now see that having such UG rankings in the initial state does not jeopardize learnability. The Constraint Demotion algorithm is easily adapted so that whenever a constraint that is part of a universal markedness subhierarchy is demoted, the constraints below it in the hierarchy are also demoted if necessary to preserve the universal subhierarchy.

3.2.3 Learnability and Total Ranking

In this subsection we take up some rather subtle issues concerning the roles of fully ranked and stratified constraint hierarchies in these learning algorithms.

The discussion here assumes that the learning data are generated by a UG-allowed grammar, which, by (2.13), is a totally ranked hierarchy. When learning is successful, the learned stratified hierarchy, even if not totally ranked, is completely consistent with at least one total ranking. The empirical basis for (2.13) is the broad finding that correct typologies of adult languages do not seem to result when constraints are permitted to form stratified hierarchies. Generally speaking, allowing constraints to have equal ranking produces empirically problematic constraint interactions.

From the learnability perspective, the formal results given for Constraint Demotion depend critically on the assumption that the target language is given by a totally ranked hierarchy. This is a consequence of a