

JOHNS HOPKINS
U N I V E R S I T Y

Department of Cognitive Science

☎ (410) 516-5250 Baltimore, MD 21218-2685 Fax: (410) 516-8020

**Learnability in Optimality Theory
(short version)**

Bruce Tesar & Paul Smolensky

October 1996

Technical Report
JHU-CogSci-96-2

<http://www.cogsci.jhu.edu/TechReports/>

TR-Request@cogsci.jhu.edu

Learnability in Optimality Theory

Bruce Tesar

The Center for Cognitive Science / Linguistics Department

Rutgers University

Piscataway, NJ 08855

tesar@ruccs.rutgers.edu

Paul Smolensky

Cognitive Science Department

Johns Hopkins University

Baltimore, MD 21218-2685

smolensky@cogsci.jhu.edu

Abstract

A central claim of Optimality Theory is that grammars may differ only in how conflicts among universal well-formedness constraints are resolved: a grammar is precisely a means of resolving such conflicts via a strict priority ranking of constraints. It is shown here how this theory of Universal Grammar yields a highly general Constraint Demotion principle for grammar learning. The resulting learning procedure specifically exploits the grammatical structure of Optimality Theory, independent of the content of substantive constraints defining any given grammatical module. The learning problem is decomposed and formal results are presented for a central subproblem, deducing the constraint ranking particular to a target language, given structural descriptions of positive examples and knowledge of universal grammatical elements. Despite the potentially large size of the space of possible grammars, the structure imposed on this space by Optimality Theory allows efficient convergence to a correct grammar. Implications are discussed for learning from overt data only, learnability of partially-ranked constraint hierarchies, and the initial state. It is argued that Optimality Theory promotes a goal which, while generally desired, has been surprising elusive: confluence of the demands of more effective learnability and deeper linguistic explanation.

How exactly does a theory of grammar bear on questions of learnability? Restrictions on what counts as a possible human language can restrict the search space of the learner. But this is a coarse observation: alone it says nothing about how data may be brought to bear on the problem, and further, the number of possible languages predicted by most linguistic theories is extremely large.¹ It would clearly be a desirable result if the nature of the restrictions imposed by a theory of grammar could contribute further to language learnability.

The central claim of this paper is that the character of the restrictions imposed by Optimality Theory (Prince and Smolensky 1991, 1993) have demonstrable and significant consequences for central questions of learnability. Optimality Theory explains linguistic phenomena through the complex interaction of violable constraints. The main results of this paper demonstrate that those constraint interactions are nevertheless restricted in a way that permits the correct grammar to be inferred from grammatical structural descriptions. These results are theorems, based on a formal analysis of the Optimality Theory framework; proofs of the theorems are contained in an appendix. The results have two important properties. First, they derive from central principles of the Optimality Theory framework. Second, they are nevertheless independent of the details of any substantive analysis of particular phenomena. The results apply equally to phonology, syntax, and any other domain admitting an Optimality Theoretic analysis. Thus, these theorems provide a learnability measure of the restrictiveness inherent in Optimality Theory's account of cross-linguistic variation *per se*: constraint reranking.

The structure of the paper is as follows. Section 1 formulates the Optimality Theoretic learning problem we address. Section 2 addresses this problem by developing the principle of Constraint Demotion, which is incorporated into an error-driven learning procedure in section 3. Section 4 takes up some issues and open questions raised by Constraint Demotion, and section 5 concludes. Section 6 is an appendix containing the formal definitions, theorems, and proofs.

1. Learnability and Optimality Theory

Optimality Theory (henceforth, ‘OT’) defines grammaticality by optimization over violable constraints. The defining reference is Prince and Smolensky 1993 (abbreviated ‘P&S’ here). Section 1.1 provides the necessary OT background, while section 1.2 outlines the approach to language learnability proposed here, including a decomposition of the overall problem; the results of this paper solve the subproblem involving direct modification of the grammar.

1.1 Optimality Theory

In this section, we present the basics of OT as a series of general principles, each exemplified within the Basic CV Syllable Theory of P&S.

1.1.1 Constraints and Their Violation

(1) Grammars specify functions.

A grammar is a specification of a function which assigns to each *input* a unique structural description or *output*. (A grammar *per se* does not provide an algorithm for computing this function, e.g., by sequential derivation.)

In Basic CV Syllable Theory (henceforth, ‘CVT’), an input is a string of Cs and Vs, e.g., /VCVC/. An output is a parse of the string into syllables, denoted as follows:

- (2)
- a. .V.CVC. = [_σ V] [_σ CVC]
 - b. ⟨V⟩.CV.⟨C⟩ = V [_σ CV] C
 - c. ⟨V⟩.CV.C□. = V [_σ CV] [_σ C□]
 - d. .□V.CV.⟨C⟩ = [_σ □V] [_σ CV] C

(These four forms will be referred to frequently in the paper, and will be consistently labeled a–d.)

Output *a* is an onsetless open syllable followed by a closed syllable; periods denote the boundaries of syllables (σ). Output *b* 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 $\langle \rangle$. These segments exemplify *underparsing*, and are not phonetically realized, so *b* is ‘pronounced’ simply as .CV. The form .CV. is the *overt form* contained in *b*. Parse *c* 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 \square , and exemplifies *overparsing*. The phonetic interpretation of this empty nucleus is an epenthetic vowel. Thus *c* has .CV.CV. as its overt form. As in *b*, the initial V of the input is unparsed in *c*. Parse *d* is also a pair of open syllables (phonetically, .CV.CV.), but this time it is the onset of the first syllable which is unfilled (notated \square ; phonetically, an epenthetic consonant), while the final C is unparsed.

(3) *Gen*: Universal Grammar provides a function *Gen* which, 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 in all possible parsings of the string into syllables, including the possible over- and underparsing structures exemplified above in (*b–d*). 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 positions 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) 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.

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

(4) *Con*: Universal Grammar provides a set *Con* of universal well-formedness constraints.

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 multi-set 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 *a* is more marked than a parse *b* with respect to *C* iff *C* assesses more marks to *a* than to *b*. (The theory recognizes the notions more- and less-marked, but not absolute numerical levels of markedness.)

The CVT constraints are given in (5).

(5) Basic CV Syllable Theory Constraints

- ONSET Syllables have onsets.
- NOCODA Syllables do *not* have codas.
- PARSE Underlying (input) material is parsed into syllable structure.
- FILL^{Nuc} Nucleus positions are filled with underlying material.
- FILL^{Ons} Onset positions (when present) are filled with underlying material.

These constraints can be illustrated with the candidate outputs in (*a–d*). The marks incurred by these candidates are summarized in table (6).

(6) Constraint Tableau for L_1

Candidates	ONSET	NOCODA	FILL ^{Nuc}	PARSE	FILL ^{Ons}
/VCVC/ →					
\Rightarrow <i>d.</i> .□V.CV.<C>				*	*
<i>b.</i> <V>.CV.<C>				**	
<i>c.</i> <V>.CV.C□.			*	*	
<i>a.</i> .V.CVC.	*	*			

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 ‘*’, one for each violation.

Candidate $a = .V.CVC.$ violates ONSET in its first syllable and NOCODA in its second; the remaining constraints are satisfied. The single mark which ONSET assesses $.V.CVC.$ is denoted *ONSET. This candidate is a *faithful* parse: it involves neither under- nor overparsing, and therefore satisfies the *faithfulness* constraints PARSE and FILL². By contrast, $b = \langle V \rangle . CV . \langle C \rangle$ violates PARSE, and more than once. This tableau will be further explained below.

1.1.2 Optimality and Harmonic Ordering

The central notion of optimality now makes its appearance. The idea is that by examining the marks assigned by the universal constraints to all the candidate outputs for a given input, we can find the least marked, or optimal, one; the only well-formed parse assigned by the grammar to the input is the optimal one (or optimal ones, if several parses should tie for optimality). The relevant notion of ‘least marked’ is not the simplistic one of just counting numbers of violations. Rather, in a given language, different constraints have

different strengths or priorities: they are not all equal in force. 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 well-formed structural description.

(7) *Constraint Ranking*: a grammar *ranks* the universal constraints in a *dominance hierarchy*.

When one constraint C_1 dominates another constraint C_2 in the hierarchy, the relation is denoted $C_1 \gg C_2$. The ranking defining a grammar is total: the hierarchy determines the relative dominance of every pair of constraints:

$$C_1 \gg C_2 \gg \dots \gg C_n$$

(8) *Harmonic Ordering*: a grammar's constraint ranking induces a *harmonic ordering* \prec of all structural descriptions. Two structures a and b are compared by identifying the highest-ranked constraint C with respect to which a and b are not equally marked: the candidate which is less marked with respect to C is the *more harmonic*, or the one with *higher Harmony* (with respect to the given ranking).

$a \prec b$ denotes that a is less harmonic than b . 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; all less harmonic candidates are ill-formed³.

A formulation of harmonic ordering that will prove quite useful for learning involves *Mark Cancellation*. Consider a pair of competing candidates a and b , with corresponding lists of violation marks $marks(a)$ and $marks(b)$. Mark Cancellation is a process applied to a pair of lists of marks, and it cancels violation marks in common to the two lists. Thus, if a constraint C assesses one or more marks $*C$ to both $marks(a)$ and $marks(b)$, an instance of

* \mathbb{C} is removed from each list, and the process is repeated until *at most one* of the lists still contains a mark * \mathbb{C} . (Note that if a and b are equally marked with respect to \mathbb{C} , the two lists contain equally many marks * \mathbb{C} , and all occurrences of * \mathbb{C} are eventually removed.) The resulting lists of *uncancelled marks* are denoted $marks'(a)$ and $marks'(b)$. If a mark * \mathbb{C} remains in the uncancelled mark list of a , then a is more marked with respect to \mathbb{C} . If the highest-ranked constraint assessing an uncancelled mark has a mark in $marks'(a)$, then $a < b$: this is the definition of harmonic ordering $<$ in terms of mark cancellation. Mark cancellation is indicated with diagonal shading in the tableau (9): one mark *PARSE cancels between the first two candidates of (6), d and b , and one uncancelled mark *PARSE remains in $marks'(b)$.

(9) Mark Cancellation

Candidates	ONSET	NoCODA	FILL ^{Nuc}	PARSE	FILL ^{Ons}
$d.$ $\langle \square V.CV.\langle C \rangle$				*	*
$b.$ $\langle V \rangle.CV.\langle C \rangle$				*	*

Defining grammaticality via harmonic ordering has an important consequence:

(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 parse p of an input I will best satisfy a constraint \mathbb{C} , unless all candidates that fare better than p on \mathbb{C} also fare worse than p on some constraint which is higher ranked than \mathbb{C} .

Harmonic ordering can be illustrated with CVT by reexamining the tableau (6) under the assumption that the universal constraints are ranked by a particular grammar, L_1 , with the ranking given in (11).

(11) Constraint hierarchy for L_1 : ONSET \gg NOCODA \gg FILL^{Nuc} \gg PARSE \gg FILL^{Ons}

The constraints (and their columns) are ordered in (6) left-to-right, reflecting the hierarchy in (11). The candidates in this tableau have been listed in harmonic order, from highest to lowest Harmony; the optimal candidate is marked manually⁴. Starting at the bottom of the tableau, $a \prec c$ 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, i.e., most violates the most highly ranked constraint: it is a , which violates ONSET. Therefore a is the less harmonic. In determining that $c \prec b$, first cancel the common mark *PARSE; c then earns the worst mark of the two, *FILL^{Nuc}. When comparing b to d , one *PARSE mark cancels, leaving $marks'(b) = \{*\text{PARSE}\}$ and $marks'(d) = \{*\text{FILL}^{\text{Ons}}\}$. The worst mark is the uncanceled *PARSE incurred by b , so $b \prec d$.

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 to provide missing onsets, and underparsing to avoid codas (it is the language denoted $\Sigma_{\text{ep,del}}^{\text{CV}}$ in P&S:§6.2.2.2).

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

(12) Constraint Hierarchy for L_2 : ONSET \gg NOCODA \gg FILL^{Ons} \gg PARSE \gg FILL^{Nuc}

Now the tableau corresponding to (6) becomes (13); the columns have been re-ordered to reflect the constraint reranking, and the candidates have been re-ordered to reflect the new harmonic ordering.

(13) Constraint Tableau for L_2

Candidates	ONSET	NoCODA	FILL ^{Ons}	PARSE	FILL ^{Nuc}
/VCVC/ →					
\Rightarrow c. ⟨V⟩.CV.C□.				*	*
b. ⟨V⟩.CV.⟨C⟩				**	
d. □V.CV.⟨C⟩			*	*	
a. .V.CVC.	*	*			

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

The relation between L_1 and L_2 illustrates a principle of Optimality Theory central to learnability concerns:

(14) Typology by Reranking

Systematic cross-linguistic variation is due entirely to variation in language-specific rankings of the universal constraints in *Con*. Analysis of the optimal forms arising from all possible rankings of *Con* gives the typology of possible human languages.

Universal Grammar 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:§6. In this typology, licit syllables may have required or optional onsets, and, independently, forbidden or optional codas.

One further principle of OT will figure in our analysis of learnability, richness of the

base. Discussion of this principle will be postponed until its point of relevance, section 4.3.

1.2 Decomposing the Learning Problem

The results presented in this paper address a particular subproblem of the overall enterprise of language learnability. That subproblem, and the corresponding results, are best understood in the context of an overall approach to language learnability. This section briefly outlines that approach. The nature of and motivation for the approach are further discussed in section 4.2.

To begin, three types of linguistic entities must be distinguished:

(15) Three Kinds of Linguistic Entities

Full structural descriptions: the candidate outputs of *Gen*, including overt structure and input.

Overt structure: the part of a description directly accessible to the learner.

The grammar: determines which structural descriptions are grammatical.

In terms of CVT, full structural descriptions are exemplified by the descriptions listed in (2). Overt structure is the part of a structural description that actually is realized phonetically. For example, in $b = \langle V \rangle.CV.\langle C \rangle$, the overt structure is CV; the unparsed segments $\langle V \rangle$ and $\langle C \rangle$ are not included. Unparsed segments are present in the full structural description, but not the overt structure. The part of the grammar to be learned is the ranking of the constraints, as exemplified in (11).

It is important to keep in mind that the grammar evaluates full structural descriptions; it does not evaluate overt structure in isolation. This is, of course, hardly novel to Optimality Theory; it is fundamental to linguistic theory in general. The general challenge of language acquisition, under any linguistic theory, is that of inferring the correct grammar from overt data, despite the gap between the two arising from the hidden elements of structural

descriptions, absent from overt data.

It is also important to distinguish three processes, each of which plays an important role in the approach to language acquisition proposed here:

(16) Three Processes

Production-Directed Parsing: mapping an underlying form (input) to its optimal description—given a grammar.

Robust Interpretive Parsing: mapping an overt structure to its full structural description, complete with all hidden structure—given a grammar.

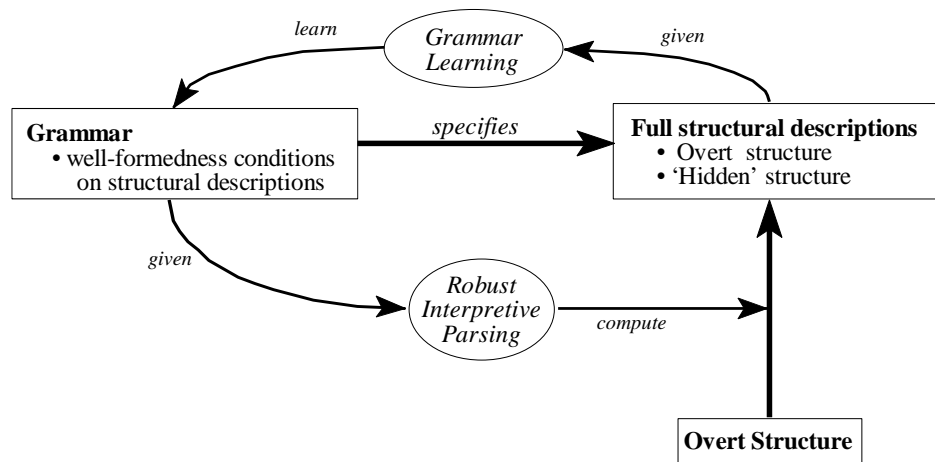
Learning the Grammar: determining a grammar from full grammatical descriptions.

Production-directed parsing is the computation of that structural description, among those candidates produced by *Gen* containing a given input, which is optimal with respect to a given ranking. Production-directed parsing takes a part of a structural description, the underlying form, and fills in the rest of the structure. Robust interpretive parsing also takes a part of a structural description and fills in the rest, but it starts with a different part, the overt structure. Robust interpretive parsing is closer to what many readers probably associate with the word “parsing.” “Robustness” refers to the fact that an overt structure not generated by the grammar currently held by the learner is not simply rejected: rather, it is assigned the most harmonic structure possible. The learner can, of course, tell that the assigned parse is not grammatical by her current grammar (by comparing it to the description her grammar assigns to the same underlying form); in fact, the learner will exploit that observation during learning. Both production-directed parsing and robust interpretive parsing make use of the same harmonic ordering of structural descriptions induced by the constraint ranking. They differ in the part of the structure they start from: production-directed parsing starts with an underlying form, and chooses among candidates with the same underlying form, while robust

interpretive parsing starts with an overt structure, and chooses among candidates with the same overt structure.

These entities and processes are all intimately connected, as schematically shown in (17).

(17) Decomposition of the Learning Problem



Any linguistic theory must ultimately be able to support procedures which are tractable performance approximations to both parsing and learning. Ideally, a grammatical theory should provide sufficient structure so that procedures for both parsing and grammar learning can be strongly shaped by grammatical principles.

In the approach to learning developed here, full structural descriptions bear not just a logical relationship between overt structures and grammars: they also play an active role in the learning process. We propose that a language learner uses a grammar to interpret overt forms by imposing on those overt forms the best structural descriptions, as determined by her current ranking. She then makes use of those descriptions in learning.

Specifically, we propose that a learner starts with an initial ranking of the constraints. As overt forms are observed, the learner uses the currently hypothesized ranking to assign

structural descriptions to those forms. These hypothesized full structures are treated by the grammar learning subsystem as the target parses to be assigned by the correct grammar: they are used to change the hypothesized ranking, yielding a new grammar. The new ranking is then used to assign new full descriptions to overt forms. This process continues, back and forth, until the correct ranking is converged upon. At that point, the ranking will assign the correct structural descriptions to each of the overt structures, and the overt structures will indicate that the ranking is correct, and should not be changed.

The process of computing optimal structural descriptions for underlying forms (production-directed parsing) has already been addressed elsewhere. Algorithms which are provably correct for significant classes of OT grammars have been developed, based upon dynamic programming (Tesar 1994, 1995ab, in press). For positive initial results in applying similar techniques to robust interpretive parsing, see Tesar, in preparation a.

At this point, we put aside the larger learning algorithm until section 4.2, for the present paper is devoted to the subproblem in (17) labelled “grammar learning”: inferring constraint rankings from full structural descriptions. The next two sections develop an algorithm for performing such inference. This algorithm has a property important for the success of the overall learning approach: when supplied with the correct structural descriptions for a language, it is guaranteed to find the correct ranking. Furthermore, the number of structural descriptions required by the algorithm is quite modest, especially when compared to the number of distinct rankings.

2. 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* 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. Call the observed grammatical structure the *winner*, and any competing structure a *loser*. The challenge faced by the learner is then, 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. The main principle is presented more precisely in this section, and an algorithm for learning constraint rankings from grammatical structural descriptions is presented in section 3.

2.1 The Basic Idea

In our CV language L_1 , the winner for input /VCVC/ is $\cdot\Box V.CV.\langle C \rangle$. Table (6) gives the marks incurred by the winner (labelled d) and by three competing *losers*. These may be used to form three *loser/winner* pairs, as shown in (18). A *mark-data pair* is the paired lists of constraint violation marks for a *loser/winner* pair.

(18) Mark-data pairs (L_1)

	<i>loser</i> < <i>winner</i>	<i>marks(loser)</i>	<i>marks(winner)</i>
$a < d$	$\cdot V.CVC. < \cdot\Box V.CV.\langle C \rangle$	*ONSET *NoCODA	*PARSE *FILL ^{Ons}
$b < d$	$\langle V \rangle.CV.\langle C \rangle < \cdot\Box V.CV.\langle C \rangle$	*PARSE *PARSE	*PARSE *FILL ^{Ons}
$c < d$	$\langle V \rangle.CV.C\Box. < \cdot\Box V.CV.\langle C \rangle$	*PARSE *FILL ^{Nuc}	*PARSE *FILL ^{Ons}

To make contact with more familiar OT constraint tableaux, the information in (18) will also be displayed in the format of (19).

(19) Initial data

<i>loser/winner pairs</i>	<i>not-yet-ranked</i>				
	FILL ^{Nuc}	FILL ^{Ons}	PARSE	ONSET	NOCODA
<i>d</i> ✓ .□V.CV.<C>		⊗	⊗		
<i>a</i> .V.CVC.				*	*
<i>d</i> ✓ .□V.CV.<C>		⊗	⊗		
<i>b</i> <V>.CV.<C>			*	*	
<i>d</i> ✓ .□V.CV.<C>		⊗	⊗		
<i>c</i> <V>.CV.C□.	*		*		

At this point, the constraints are unranked; the dotted vertical lines separating constraints in (19) convey that no relative ranking of adjacent constraints is intended. The winner is indicated with a ✓; \mathbb{E} 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, *marks(winner)*, are distinguished by the symbol ⊗. Diagonal shading denotes mark cancellation, as in tableau (9).

Now 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 (8), 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 (18).

(20) Mark-data pairs after cancellation (L_1)

<i>loser/winner pairs</i>		<i>marks'(loser)</i>	<i>marks'(winner)</i>
$a < d$.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}

The cancelled marks have been ~~struck out~~. Note that the cancellation operation which transforms *marks* to *marks'* is defined only on *pairs* of sets of marks; e.g., *PARSE is cancelled in the pairs $b < d$ and $c < d$, but not in the pair $a < d$. Note also that cancellation of marks is done token-by-token: in the row $b < d$, one but not the other mark *PARSE in *marks(b)* is cancelled.

The table (20) of mark-data after cancellation is the data on which Constraint Demotion operates. Another representation in tableau form is given in (19), where common marks in each loser/winner pair of rows are indicated as ‘cancelled’ by diagonal shading. This table also reveals what successful learning must accomplish: the ranking of the constraints must be adjusted so that, for each pair, all of the uncanceled winner marks \otimes 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 \otimes 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) which is further to the left (dominant in the hierarchy) than all of the columns containing uncanceled \otimes (winner marks).

The algorithm to accomplish this is based upon the principle in (21).

(21) The 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

uncancelled loser mark.

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

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

2.2 Stratified Domination Hierarchies

Optimality Theory 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:

(22) Stratified Domination Hierarchy

$$\{C_1, C_2, \dots, C_3\} \gg \{C_4, C_5, \dots, C_6\} \gg \dots \gg \{C_7, C_8, \dots, C_9\}$$

The constraints C_1, C_2, \dots, 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 C_4, C_5, \dots, 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 (8) needs to be elaborated slightly for stratified hierarchies. When C_1 and C_2 are in the same stratum, two marks $*C_1$ and $*C_2$ 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 (23) gives a simple illustration.

(23) Harmonic ordering with a stratified hierarchy: $C_1 \gg \{C_2, C_3\} \gg C_4$

	C_1	C_2	C_3	C_4
p_1	*!		*	
p_2			*	*!
$\rightarrow p_3$		*		
p_4			* *!	

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 ‘cancelling,’ because they do not decide between the candidates. It is crucial, though, that the marks not be cancelled for the purposes of learning. The term Mark Cancellation, as used in the rest of this paper, should be understood to only cancel marks assessed by the same constraint to competing candidates, independent of the constraint hierarchy.

2.3 An Example: Basic CV Syllable Theory

Constraint Demotion (abbreviated CD) will now be illustrated using CVT; specifically, with the target language L_1 of (6,11). The initial stratified hierarchy is set to

$$(24) \quad \mathcal{H} = \mathcal{H}_0 = \{\text{FILL}^{\text{Nuc}}, \text{FILL}^{\text{Ons}}, \text{PARSE}, \text{ONSET}, \text{NOCODA}\}$$

Suppose that the first loser/winner pair is $b \prec d$ of (18). Mark Cancellation is applied to the corresponding pair of mark lists, resulting in the mark-data pair shown in (25).

(25) Mark-data pair, Step 1 (L_1)

<i>loser</i> \prec <i>winner</i>		<i>marks'(loser)</i>	<i>marks'(winner)</i>
$b \prec d$	$\langle V \rangle.CV.\langle C \rangle \prec \cdot \square V.CV.\langle C \rangle$	*PARSE *PARSE	*PARSE *FILL ^{Ons}

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 *FILL^{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 (26).

$$(26) \quad \mathcal{H} = \{\text{FILL}^{\text{Nuc}}, \text{PARSE}, \text{ONSET}, \text{NOCODA}\} \gg \{\text{FILL}^{\text{Ons}}\}$$

This demotion is shown in tableau form in (27); recall that strata are separated by solid vertical lines, whereas dotted vertical lines separate constraints in the same stratum; diagonal

shading denotes mark cancellation. The uncanceled winner mark \otimes 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 \otimes (which is therefore greyed out).

(27) First Demotion

<i>loser/winner pair</i>	FILL ^{Nuc}	FILL ^{Ons}	PARSE	ONSET	NOCODA	FILL ^{Ons}
<i>d</i> ✓.□V.CV.<C>		\otimes	\otimes			\otimes
<i>b</i> <V>.CV.<C>			*	*!		

Now another loser/winner pair is selected. Suppose this is $a < d$ of (18):

(28) Mark-data pair for CD, Step 2 (L_1)

<i>loser < winner</i>	<i>marks'(loser)</i>	<i>marks'(winner)</i>
$a < d$.V.CVC. < .□V.CV.<C>	*ONSET *NOCODA	*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 dominated by ONSET. FILL^{Ons} is so dominated. PARSE is not, however, so it is demoted to the stratum immediately below that of ONSET.

$$(29) \quad \mathcal{H} = \{ \text{FILL}^{\text{Nuc}}, \text{ONSET}, \text{NOCODA} \} \gg \{ \text{FILL}^{\text{Ons}}, \text{PARSE} \}$$

In tableau form, this demotion is shown in (30). (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.)

(30) Second Demotion

<i>loser/winner pair</i>	FILL ^{Nuc}	PARSE	ONSET	NOCODA	FILL ^{Ons}	PARSE
<i>d</i> $\langle \square V.CV.\langle C \rangle$		⊗			⊗	⊗
<i>a</i> V.CVC.			*!	*!		



Suppose now that the next *loser/winner* pair is:

(31) Mark-data pair for CD, Step 3 (L_1)

<i>loser</i> < <i>winner</i>	<i>marks'(loser)</i>	<i>marks'(winner)</i>
<i>c</i> < <i>d</i> $\langle V \rangle.CV.C\square.$ < $\langle \square V.CV.\langle C \rangle$	*PARSE *FILL ^{Nuc}	*PARSE *FILL ^{Ons}

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.

Suppose the next *loser/winner* pair results from a new input, /VC/, with a new optimal parse, $\langle \square V.\langle C \rangle$.

(32) Mark-pair for CD, Step 4 (L_1)

<i>loser</i> < <i>winner</i>	<i>marks'(winner)</i>	<i>marks'(winner)</i>
$\langle VC \rangle$ < $\langle \square V.\langle C \rangle$	*PARSE *PARSE	*PARSE *FILL ^{Ons}

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 (33).

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

This demotion is shown in tableau (34).

(34) Third Demotion

<i>loser/winner</i> pair	FILL ^{Nuc}	ONSET	NOCODA	FILL ^{Ons}	PARSE	FILL ^{Ons}
☞✓ .□V.(C)				⊗	⊗	⊗
⟨VC⟩					⊗	⊗!

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.

2.4 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 towards the correct hierarchy? The answer is no, and understanding why reveals the logic behind Constraint Demotion.

Consider the tableau shown in (35), 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, $a < d$.

(35) The Disjunction Problem

<i>loser/winner pair</i>	FILL ^{Ons}	ONSET	FILL ^{Nuc}	NOCODA	PARSE
<i>d</i> ✓ .□V.CV.<C>	⊗				⊗
<i>a</i> V.CVC.		*		*	

There are no marks in common, so no marks are cancelled. For the winner to be more harmonic than the loser, at least one of the loser's marks must dominate all of the winner's marks. This relation is expressed in (36).

$$(36) \quad (\text{ONSET or NOCODA}) \gg (\text{FILL}^{\text{Ons}} \text{ and PARSE})$$

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* of 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 isn't 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 detangling 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 (35), that is ONSET). Thus, if other data have already determined that $\text{ONSET} \gg \text{NOCODA}$, that relationship is preserved. The constraints

violated by the winner are only demoted as far as necessary.

2.5 The Initial Hierarchy

The illustration of Constraint Demotion given in section 2.3 started with initial hierarchy \mathcal{H}_0 , given in (24), having all the constraints in one stratum. Using that as an 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 in sections 6.1 to 6.3 assumes \mathcal{H}_0 as the initial hierarchy, and proves the following result, as (56, 65):

(37) Theorem: Correctness of Constraint Demotion

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.

However, using \mathcal{H}_0 as the initial hierarchy is not required by CD. In fact, convergence is obtained no matter what initial hierarchy is used; this is proven in section 6.4. 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.

This is not all there is to be said about the initial hierarchy; the issue is discussed further in section 4.3.

3. Selecting Competing Descriptions: Error-Driven Constraint Demotion

Having developed the basic principle of Constraint Demotion, we now show how it can be incorporated into a procedure for learning a grammar from correct structural descriptions.

3.1 Parsing Identifies Informative Competitors

CD operates on loser/winner pairs, deducing consequences for the grammar from the fact that the winner must be more harmonic than the loser. The winner is a positive example provided externally to the grammar learner: a parse of some input (e.g., an underlying lexical form in phonology; a predicate/argument structure in syntax), a parse taken to be optimal according to the target grammar. The loser is an alternative parse of the same input, which must be suboptimal with respect to the target grammar (unless it happens to have exactly the same marks as the winner). Presumably, such a loser must be generated by the grammar learner. Whether the loser/winner pair is informative depends both on the winner and on the loser.

An antagonistic learning environment can of course always deny the learner necessary informative examples, making learning the target grammar impossible. We consider this uninteresting and assume that as long as there remain potentially informative positive examples, these are not maliciously withheld from the learner (but see section 4.3 for a discussion of the possibility of languages underdetermined by positive evidence). This still leaves a challenging problem, however. Having received a potentially informative positive example, a winner, the learner needs to find a corresponding loser which forms an informative

loser/winner pair. In principle, if the winner is a parse of an input I , then any of the competing parses in $Gen(I)$ can be chosen as the loser; typically, there are an infinity of choices, not all of which will lead to an informative loser/winner pair. What is needed is a procedure for choosing a loser which is guaranteed to be informative, as long as any such competitor exists.

The idea (Tesar, in press) is simple. Consider a learner in the midst of learning, with current constraint hierarchy \mathcal{H} . A positive example p is received: the target parse of an input I . It is natural for the learner to compute her own parse p' for I , optimal with respect to her current hierarchy \mathcal{H} . If the learner's parse p' is different from the target parse p , learning should be possible; otherwise, it isn't. For if the target parse p equals the learner's parse p' , then p is already optimal according to \mathcal{H} ; no demotion occurs, and no learning is possible. On the other hand, if the target parse p is *not* the learner's parse p' , then p is suboptimal according to \mathcal{H} , and the hierarchy needs to be modified so that p becomes optimal. In order for a loser to be informative when paired with the winner p , the Harmony of the loser (according to the current \mathcal{H}) must be greater than the Harmony of p : only then will demotion occur to render p more harmonic than the loser. The obvious choice for this loser is p' : it is of maximum Harmony according to \mathcal{H} , and if any competitor to the winner has higher Harmony according to \mathcal{H} , then p' must. The type of parsing responsible for computing p' is production-directed parsing, as defined in (16): given an input I and a stratified hierarchy \mathcal{H} , compute the optimal parse(s) of I . This is the problem solved in a number of general cases by Tesar (1995b), as discussed in section 1.2.

If the optimal parse given the current \mathcal{H} , *loser*, should happen to equal the correct parse *winner*, the execution of CD will produce no change in \mathcal{H} : no learning can occur. In fact, CD need be executed only when there is a mismatch between the correct parse and the optimal parse assigned by the current ranking. This is an *error-driven* learning algorithm (Wexler and Culicover 1980). Each observed parse is compared with a computed parse of

the input. If the two parses match, no error occurs, and so no learning takes place. If the two parses differ, the error is attributed to the current hypothesized ranking, and so CD is used to adjust the hypothesized ranking. The resulting algorithm is called *Error-Driven Constraint Demotion* (EDCD).

(38) The Error-Driven Constraint Demotion Algorithm (EDCD)

Given: a hierarchy \mathcal{H} and a set *PositiveData* of grammatical structural descriptions.

For each description *winner* in *PositiveData*:

Set *loser* to be the optimal description assigned by \mathcal{H} to *I*, the underlying form of *winner*.

If *loser* is identical to *winner*, keep \mathcal{H} ;

Else:

- apply Mark Cancellation, getting (*marks'*(*loser*), *marks'*(*winner*))
- apply Constraint Demotion to (*marks'*(*loser*), *marks'*(*winner*)) and \mathcal{H}
- adopt the new hierarchy resulting from demotion as the current hierarchy

This algorithm demonstrates that using the familiar strategy of error-driven learning does not require inviolable constraints or independently evaluable parameters. Because Optimality Theory is defined by means of optimization, errors are defined with respect to the relative Harmony of several entire structural descriptions, rather than particular diagnostic criteria applied to an isolated parse. Constraint Demotion accomplishes learning precisely on the basis of the comparison of entire structural descriptions.⁵

3.2 Data Complexity: The Amount of Data Required to Learn the Grammar

The data complexity of a learning algorithm is the amount of data that needs to be supplied to the algorithm in order to ensure that it learns the correct grammar. For ED CD, an opportunity for progress towards the correct grammar is presented every time an error

occurs (a mismatch between a positive datum and the corresponding parse which is optimal with respect to the current hypothesized grammar). Any such error 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 errors that could occur before the correct grammar is reached.

With EDCD, an error 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 two stratified hierarchies is $N(N-1)$, where N is the number of constraints in the grammar; this then is the maximum number of errors made prior to learning the correct hierarchy. This result is proved in the appendix as (74):

(39) Theorem: Computational complexity of Constraint Demotion

Starting with an arbitrary initial hierarchy, the number of informative loser/winner pairs required for learning is at most $N(N-1)$, where N = number of constraints in *Con*.

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 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 just given for EDCD 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 EDCD and the number of total rankings: when $N=10$, the EDCD data complexity is 90, while the number of total rankings is over 3.6 million. With 20 constraints, the EDCD data complexity is 380, while the number of total rankings is over 2 billion billion (2.43×10^{18}). This reveals the restrictiveness of the structure imposed by Optimality Theory 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.

4. Issues for the Constraint Demotion Approach

We close by considering a number of implications and open questions arising from the learnability results of the preceding two sections.

4.1 Learnability and Total Ranking

The discussion in this paper assumes that the learning data are generated by a UG-allowed grammar, which, by (14), 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 (14) 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 Error-Driven Constraint Demotion depend critically on the assumption that the target language is given by a totally-ranked hierarchy. This is a consequence of a principle implicit in EDCD. This principle states that the learner should assume that the observed description is optimal for the corresponding input, and that it is the *only* optimal description. This principle resembles other proposed learning principles, such as Clark's Principle of Contrast (E. Clark 1987) and Wexler's Uniqueness Principle (Wexler 1981). EDCD makes vigorous use of this learning principle.

In fact, it is possible for the algorithm to run endlessly when presented data from a non-totally-ranked stratified hierarchy. For the minimal illustration, suppose that there are two constraints \mathbb{C} and \mathbb{C}' , and two candidate parses p and p' , where p violates only \mathbb{C} and p' violates only \mathbb{C}' . Suppose \mathbb{C} and \mathbb{C}' are both initially top-ranked. Assume the target hierarchy also ranks \mathbb{C} and \mathbb{C}' in the same stratum, and that the two candidates tie for optimality. Both p and p' will therefore be separately observed as positive evidence. When p is observed, EDCD will assume the competitor p' to be suboptimal, since its marks are not identical to those of p . EDCD will therefore demote \mathbb{C} , the constraint violated by the observed optimal parse p , below \mathbb{C}' . Later, when the other optimal candidate p' is observed, EDCD will reverse the rankings of the constraints. This will continue endlessly, and learning will fail to converge. Notice that this instability occurs even though the initial hierarchy correctly had the constraints in the same stratum. Not only does the algorithm fail to converge on the non-fully-ranked target hierarchy: when given the correct hierarchy, in time EDCD rejects it.

In understanding this somewhat unusual state of affairs, it is important to carefully distinguish the space of target grammars being learned from the space of hypotheses being explored during learning. It is often assumed in learnability theory that language acquisition

operates within the limits imposed by UG: that hypothesized grammars are always fully-specified grammars admitted by UG. This has the advantage that learning can never terminate in a UG-disallowed state; such a learning process makes it obvious why adult grammars lie in the UG-allowed space. The learning approach presented here provides a different kind of answer: UG-disallowed grammars contained in the working hypothesis space cannot be learned by the learning algorithm. Consistent with a theme of recent work in Computational Learning Theory (e.g., Pitt and Valiant 1988, Kearns and Vazirani 1994; for a tutorial, see Haussler 1996), learning a member of the target space is greatly aided by allowing a learning algorithm to search within a larger space: the space of stratified hierarchies.

How does the learner get to a totally-ranked hierarchy? At the endpoint of learning, the hierarchy may not be fully ranked. The result is a stratified hierarchy with the property that it *could* be further refined into typically several fully-ranked hierarchies, each consistent with all the learning data. Lacking any evidence on which to do so, the learning algorithm does not commit to any such refinement; it is error-driven, and no further errors are made. In human terms, one could suppose that by adulthood, a learner has taken the learned stratified hierarchy and refined it to a fully-ranked hierarchy. It is not clear that anything depends upon which fully-ranked hierarchy is chosen.

It is currently an open question whether the Constraint Demotion approach can be extended to learn languages generated by stratified hierarchies in general, including those which are inconsistent with any total ranking. In such languages, some inputs may have multiple optimal outputs that do not earn identical sets of marks. In such a setting, the learner's primary data might consist of a set of underlying forms, and for each, *all* its optimal structural descriptions, should there be more than one. Much of the analysis might extend to this setting, but the algorithm would need to be extended with an additional step to handle pairs $opt_1 \sim opt_2$ of tying optima. In this step, each mark in $marks'(opt_1)$ must be placed in

the same stratum as a corresponding mark in $marks'(opt_2)$: a somewhat delicate business. Indeed, achieving ties for optimality between forms which incur different marks is always a delicate matter. It appears likely to us that learning languages which do not derive from a totally-ranked hierarchy is in general much more difficult than the totally-ranked case. If this is indeed true, demands of learnability could ultimately explain a fundamental principle of OT: UG admits only (adult) grammars defined by totally-ranked hierarchies.

While learnability appears to be problematic in the face of ties for optimality between outputs with *different* marks (impossible given a totally-ranked hierarchy), recall that EDCD has no problems whatever coping with ties for optimality between outputs with the *same* marks (possible given a totally-ranked hierarchy).

4.2 Iterative Approaches to Learning Hidden Structure

The learner can't deduce the hidden structure for overt structures until she has learned the grammar; but she can't learn the grammar until she has the hidden structure. This feature of the language learning problem is challenging, but not at all special to language, as it turns out. Even in such mundane contexts as a computer learning to recognize handwritten digits, this same problem arises. This problem has been extensively studied in the learning theory literature (often under the name 'unsupervised learning,' e.g., Hinton 1989). Much of the work has addressed automatic speech recognition (mostly under the name 'Hidden Markov Models,' e.g., Baum and Petrie 1966, Bahl, Jelinek and Mercer 1983, Brown et al. 1990); these speech systems are simultaneously learning (i) when the acoustic data they are 'hearing' is an example of, say, the phone [f], and (ii) what makes for a good acoustic realization of [f].

This problem has been addressed, in theory and practice, with a fair degree of success. The formulation is approximately as follows. A parametrized system is assumed which, given the values of hidden variables, produces the probabilities that overt variables will have various values: this is the *model* of the relation between hidden and overt variables. Given the hidden

variables constituting [f] within a sequence of phones, such a model would specify the probabilities of different acoustic values in the portion of the acoustic stream corresponding to the hidden [f]. The learning system needs to learn the correct model parameters so that hidden [f]s will be associated with the correct acoustic values, at the same time as it is learning to classify all acoustic tokens of [f] as being of type [f].

(40) The Problem of Learning Hidden Structure

Given: a set of overt learning data (e.g., acoustic data)

a parametrized model which relates overt information to hidden structure
(e.g., abstract phones)

Find: a set of model parameters such that the hidden structure assigned to the data
by the model makes the overt data most probable (this model ‘best
explains’ the data)

There is a class of algorithms for solving this type of problem, the Expectation-Maximization or *EM* algorithms (Dempster, Laird and Rubin 1977; for recent tutorial introductions, see Nádas and Mercer 1996, Smolensky 1996a). The basic idea common to this class of algorithms may be characterized as in (41).

(41) EM-type solution to the Problem of Learning Hidden Structure

0. Adopt some initial model of the relation between hidden and overt structure; this can be a random set of parameter values, or a more informed initial guess.
1. Given this initial model, and given some overt learning data, find the hidden structure that makes the observed data most probable according to the model. Hypothesizing this hidden structure provides the best explanation of the overt data, assuming the current (generally poor) model.
2. Using the hidden structure assigned to the overt data, find new model parameter

values that make the complete (hidden and overt) data most probable.

3. Now that the model has been changed, it will assign different (generally more correct) hidden structure to the original overt data. The algorithm executes steps 1 and 2 repeatedly, until the values of the model and the hidden structure converge (stop changing).

This kind of algorithm can be proven to converge for a number of classes of statistical learning problems.

In Optimality Theory, the Harmony of structural descriptions is computed from the grammar non-numerically, and there is no probabilistic interpretation of Harmony. But the approach in (41) could still be applied. Whether this iterative algorithm can be proven to converge, whether it converges in a reasonable time—these and other issues are all open research problems at the moment. But initial positive experimental results learning stress systems (Tesar, in preparation b) and extensive previous experience with EM-type algorithms in related applications suggests that there are reasonable prospects for good performance, as long as algorithms can be devised for the subproblems in steps 1 and 2 of (41) which satisfy a ‘correctness’ criterion: they give the respective correct answers when given the correct respective input. In other words, given the correct model, the correct hidden structure is assigned the overt data, and vice-versa. The corresponding OT subproblems are precisely those addressed by the three processes in (16): production-directed parsing, robust interpretive parsing, and grammar learning. Significant progress has already been made on parsing algorithms. The work in this paper completely satisfies this criterion for learning the grammar: EDCD finds the correct ranking, given the correct full descriptions (including the hidden structure).

4.3 Implications of Richness of the Base

A relevant central principle of Optimality Theory not yet considered is this:

(42) 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 descriptions which emerge from the grammar when it is fed the universal set of all possible inputs.

Thus, 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 properties of the lexicon arise indirectly from the grammar, which delimits the inventory from which the lexicon is drawn. There are no morpheme structure constraints on phonological inputs; no lexical parameter which determines whether a language has *pro*.

As pointed out to us by Alan Prince (1993), richness of the base has significant implications for the explanatory role of the grammar, in particular the relationship between the *faithfulness* constraints (e.g., PARSE and FILL) and the *structural* constraints. Recall that the faithfulness constraints require the overt structure of a description to match the underlying form. In order for marked structures to appear in overt structures, one or more of the faithfulness constraints must dominate the structural constraints violated by the marked structure. Conversely, a language in which a marked structure never appears is properly explained by having the relevant structural constraints dominate the faithfulness constraints.

Consider CVT. A language like L_1 , all of whose lexical items surface as sequences .CV. syllables, has a systematic property. This cannot be explained by stipulating special structure in the lexicon, namely, a lexicon of underlying forms consisting only of CV sequences. It is not sufficient that the grammar yield .CV. outputs when given only CV inputs: it must give .CV. outputs even when the input is, say, /VCVC/, as shown in (6). This can only be achieved by rankings in which faithfulness constraints are dominated by the structural constraints. (8) is such a ranking.

What kind of evidence could lead the learner to select the correct hierarchy? One possibility is grammatical alternations. Alternations occur precisely because the underlying form of an item is altered in some environments in order to satisfy high-ranked structural constraints, at the expense of faithfulness. When learning the underlying forms, the learner could use the alternations as evidence that faithfulness constraints are dominated.

Another proposal, suggested by Prince, is that the initial ranking has the faithfulness constraints lower-ranked than the structural constraints. The idea is that structural constraints will only be demoted below the faithfulness constraints in response to the appearance of marked forms in observed overt structures. This proposal is similar in spirit to the Subset Principle (Angluin 1978, Berwick 1986, Pinker 1986, Wexler and Manzini 1987). Because .CV. syllables are unmarked, i.e., they violate no structural constraints, all languages include them in their syllable structure inventory; other, marked, syllable structures may or may not appear in the inventory. Starting the faithfulness constraints below syllable structure constraints means starting with the smallest syllable inventory: only the unmarked syllable. If positive evidence is presented showing that marked syllables must also be allowed, the constraint violations of the marked syllables will force demotions of structural constraints below faithfulness so that underlying structures like /CVC/ can surface as .CVC. But if no positive evidence is provided for admitting marked syllables into the inventory, the initial, smallest, inventory will remain.

One notable advantage of the latter proposal is that it accords well with recent work in child phonological acquisition (Demuth 1995, Gnanadesikan 1995, Levelt 1995). This work has argued that a range of empirical generalizations concerning phonological acquisition can be modelled by constraint reranking. This work proceeds from two assumptions: (a) the child's input is the correct adult form; (b) the initial ranking is one in which the faithfulness constraints are dominated by the structural constraints. (For further discussion of the relation

of these assumptions to the learnability theory developed here, see Smolensky 1996bc.)

4.4 Learning Underlying Forms

One aspect of acquisition not yet discussed is acquisition of the underlying forms contained in the lexical entries. According to the principle of richness of the base (42), the set of possible underlying forms is universal; since we are assuming here that knowledge of universals need not be learned, in a sense there is no learning problem for *possible* underlying forms. For interesting aspects of syntax, this is pretty much all that need be said. In OT analyses of grammatical voice systems (Legendre, Raymond and Smolensky 1993), inversion (Grimshaw 1993, to appear), *wh*-questions (Billings and Rudin 1994; Legendre et al. 1995, Ackema and Neeleman, in press; Legendre, Smolensky and Wilson, in press), and null subjects (Grimshaw and Samek-Lodovici 1995, Samek-Lodovici 1995, Grimshaw and Samek-Lodovici 1996), the set of underlying forms is universal, and all cross-linguistic variation arises from the grammar: the constraint ranking is all that need be learned. The inputs in these syntactic analyses are all some kind of predicate/argument structure, the kind of semantic structure that has often been taken as available to the syntactic learner independently of the overt data (e.g., Hamburger and Wexler 1973).

In phonology, however, there is nearly always an additional layer to the question of the underlying forms. While it is as true of phonology as of syntax that richness of the base entails a universal input set, there is the further question of which of the universally available inputs is paired with particular morphemes: the problem of learning the language-dependent underlying forms of morphemes.⁶

This problem was addressed in P&S:§9, where the following principle was developed:

(43) **Lexicon Optimization:** Suppose given an overt structure φ and a grammar. Consider all structural descriptions (of all inputs) with overt part equal to φ ; let the one with

maximal Harmony be p , a parse of some input I . Then I is assigned as the underlying form of φ .⁷

The principle of Lexicon Optimization casts the learning of underlying forms as an optimization problem. This permits the problem to be approached with optimization strategies similar to those already proposed here for the learning of the constraint rankings. An iterative approach would involve an algorithm which computes the optimal underlying forms given the current ranking, and then uses those hypothesized underlying forms when computing the hypothesized interpretive parses of overt learning data; these parses are then used to determine a new ranking, and the process repeats until convergence.

4.5 Parametric Independence and Linguistic Explanation

It can be instructive to compare the learning approach presented here with recent learnability work conducted within the Principles and Parameters (P&P) framework. In the P&P framework, cross-linguistic variation is accounted for by a set of parameters, where a specific grammar is determined by fixing each parameter to one of its possible values. Because OT and P&P both use a finite space of possible grammars, the correct grammar in either framework can be found, in principle, by brute-force enumeration of the space of possible grammars.⁸

Two types of learnability research within P&P are useful as contrasts. The first is the Cue Learning approach. This is exemplified by Dresher and Kaye (1990), which adopts a well-defined parametrized space of grammars for a limited part of linguistic phenomena, metrical stress, and analyzes it in great detail. The goal of the analysis is to identify, for each setting of each parameter, some surface pattern, a ‘cue’, that is diagnostic for that parameter setting. The learner then monitors overt data looking for these cues, sometimes in a particular order. Dresher and Kaye’s cues are entirely specific to their particular parametric system. A modification to the parameter system could invalidate some of the proposed cues, requiring

that new ones be sought. Any attempt to apply cue learning to other areas of linguistic phenomena essentially start from scratch; the effort will be dictated entirely by the details of the chosen particular analysis of the phenomena.

A quite different tack is represented in the work of Gibson and Wexler (1994). They propose a learning algorithm, the Triggering Learning Algorithm (TLA) which can be applied to any instance of the general class of P&P theories. TLA is a form of error-driven random search. In response to an error, a parameter is selected at random and its value is changed; if the change renders the input analyzable, the new parameter setting is kept. The possible success of the algorithm is analyzed in terms of the existence of “triggers.” A trigger is a datum which indicates the appropriate value for a specific parameter. The learner is not assumed to be endowed with prior knowledge of the triggers, as is assumed with cues; success depends upon the learner occasionally guessing the right parameter in response to an error on a trigger, so that the parameter is set properly. This approach uses a hypothesized grammar as a kind of black box, issuing accept/reject judgements on overt structures, but nothing more.

A related algorithm makes even less use of the grammar. The algorithm of Niyogi and Berwick (1993) responds to errors by flipping parameters randomly, regardless of the resulting (un)analyzability. The algorithm uses the grammar only to detect errors. It is of course possible to apply algorithms resembling these to OT grammar spaces (in fact, Pulleyblank and Turkel (in press) have already formulated and studied a ‘Constraint-Ranking Triggering Learning Algorithm’). Indeed, any of a number of generic search algorithms could be applied to the space of OT grammars (e.g., Pulleyblank and Turkel 1995 have also applied a genetic algorithm to learning OT grammars).

These approaches to learnability analysis within the P&P theory either: (i) use the structure of a *particular* substantive theory, or (ii) make no use of the structure of the theory

beyond its ability to accept/reject overt structures. The approach advocated in this paper falls in between these two extremes, taking advantage of structure (strict domination of violable constraints) provided by the grammatical theory, but not restricted to any particular set of linguistic phenomena (e.g., metrical stress, or even phonology).

It is significant that a trigger provides information about the value of a single parameter, rather than relationships between the values of several parameters.⁹ This property is further reinforced by a proposed constraint on learning, the Single Value Constraint (R. Clark 1990, Gibson and Wexler 1994): successive hypotheses considered by a learner may differ by the value of at most one parameter. The result is that learnability concerns in the P&P framework favor parameters which are independent: they interact with each other as little as possible, so that the effects of each parameter setting can be distinguished from the effects of the other parameters. In fact, this property of independence has been proposed as a principle for grammars (Wexler and Manzini 1987). Unfortunately, this results in a conflict between the goals of learnability, which favor independent parameters with restricted effects, and the goals of linguistic theory, which favor parameters with wide-ranging effects and greater explanatory power (see Safir 1987 for a discussion of this conflict).

Optimality Theory may provide the opportunity for this conflict to be avoided. In Optimality Theory, interaction between constraints is not only possible but explanatorily crucial. Cross-linguistic variation is explained not by variation in the substance of individual constraints, but by variation in the relative ranking of the same constraints. Cross-linguistic variation is thus only possible to the extent that constraints interact. The Constraint Demotion learning algorithm not only tolerates constraint interaction, but is based entirely upon it. Informative data provide information not about one constraint in isolation, but about the results of interaction between constraints. Constraints which have wide-ranging effects benefit learnability. Thus the results presented here provide evidence that in Optimality

Theory, linguistic explanation and learnability work together: they both favor interacting constraints with wide-ranging effects and explanatory power.

This attractive feature arises from the fact that Optimality Theory defines grammaticality in terms of optimization over violable constraints. This central principle makes constraint interaction the main explanatory mechanism. It provides the implicit negative data used by Constraint Demotion precisely because it defines grammaticality in terms of the comparison of candidate descriptions, rather than in terms of the structure of each candidate description in isolation. Constraint Demotion proceeds by comparing the constraint violations assessed candidate structural descriptions. This makes constraint interaction the basis for learning.

By making constraint interaction the foundation of both linguistic explanation and learning, Optimality Theory creates the opportunity for the full alignment of these two goals. The discovery of sets of constraints which interact strongly in ways that participate in diverse linguistic phenomena represents progress for both theoretical explanation and learnability. Clearly, this is a desirable property for a theoretical framework.

5. Summary and Conclusions

This paper advocates an approach to language learning in which the grammar and analyses of the observed data are simultaneously iteratively approximated via optimization. This approach is motivated in part by similarities to work in statistical and computational learning theory. The approach is fundamentally based on the structure of Optimality Theory, in particular the definition of grammaticality in terms of optimization over violable constraints, and the resolution of conflicting constraints via strict domination.

The algorithm presented, Error-Driven Constraint Demotion, solves a critical part of the learning problem as construed by the proposed approach. EDCD disentangles the constraint interactions to find a constraint ranking making each of the given structural

descriptions optimal. The success of the algorithm on this task is guaranteed; the correctness is a theorem. Further, the algorithm succeeds with quite modest time and data requirements, in the face of the possibly huge number of possible human grammars. These modest resource requirements contribute significantly to the overall goal of a learnability account with requirements realistic for that of a human child. The formal properties are cause for optimism that formal results may be obtained for other parts of the overall problem of language learning, stronger formal results than previously obtained within any linguistic framework.

EDCD succeeds by exploiting the implicit negative evidence made available by the structure of Optimality Theory. Because a description is grammatical only in virtue of being more harmonic than all of its competitors, the learner may select informative competitors for use as negative evidence. Because it uses this structure inherent in the Optimality Theory framework, the algorithm is informed by the linguistic theory, without being parochial to any proposed substantive theory of a particular grammatical module. EDCCD not only tolerates but thrives on constraint interaction, the primary explanatory device of the framework. Thus, an opportunity is now available for greater theoretical synergy in simultaneously meeting the demands of language learnability and those of linguistic explanation.

6. Appendix: Correctness and Data Complexity

The formal analysis of Error-Driven Constraint Demotion learning proceeds as follows. A language L is presumed, which is generated by some total ranking. Section 6.1 sets up the basic machinery of stratified constraint hierarchies. Section 6.2 identifies, for any language L , a distinguished stratified hierarchy which generates it, the *target hierarchy*. Section 6.3 defines Constraint Demotion. The case where all constraints are initially top-ranked is analyzed first, and CD is shown to converge to the target hierarchy. A distance metric between hierarchies is defined, and it is shown that CD monotonically reduces the distance between the working hypothesis hierarchy and the target, decreasing the distance by at least

one unit for each informative example. The maximum number of informative examples needed for learning is thus bounded by the distance between the initial hierarchy and the target. Section 6.4 extends the results to arbitrary initial constraint hierarchies. Section 6.5 demonstrates the adequacy of production-directed parsing for selecting competitors, proving that Error-Driven Constraint Demotion will converge to a hierarchy consistent with all positive data presented.

6.1 Stratified Hierarchies

- (44) **Def.** A *stratum* is a set of constraints. A *stratified hierarchy* is a linearly ordered set of strata which partition the universal constraints. A hierarchy distinguishes one stratum as the top stratum. Each stratum other than the top stratum is immediately dominated by exactly one other stratum. The top stratum immediately dominates the second stratum, which immediately dominates the third stratum, and so forth.
- (45) **Def.** A *total ranking* is a stratified hierarchy where each stratum contains precisely one constraint.
- (46) **Def.** A constraint C_1 is said to dominate constraint C_2 , denoted $C_1 \gg C_2$, in hierarchy \mathcal{H} if the stratum containing C_1 dominates the stratum containing C_2 in hierarchy \mathcal{H} .
- (47) **Def.** The *offset* of a constraint C in a hierarchy \mathcal{H} is the number of strata that dominate the stratum containing C . C is *in a lower stratum* in \mathcal{H}_1 than in \mathcal{H}_2 if the offset of C in \mathcal{H}_1 is greater than in \mathcal{H}_2 . C is *in the same stratum* in \mathcal{H}_1 and \mathcal{H}_2 if it has the same offset in both.
- (48) **Def.** A constraint hierarchy \mathcal{H}_1 *h-dominates* \mathcal{H}_2 if every constraint is in the same or a lower stratum in \mathcal{H}_2 than in \mathcal{H}_1 .
- (49) **Def.** A constraint hierarchy \mathcal{H}_2 is called a *refinement* of \mathcal{H}_1 if every domination relation

$\mathbb{C} \gg \mathbb{C}'$ of \mathcal{H}_1 is preserved in \mathcal{H}_2 .

(50) **Def.** \mathcal{H}_0 denotes the stratified hierarchy with all of the constraints in the top stratum.

(51) **Lemma** \mathcal{H}_0 h-dominates all hierarchies.

Proof \mathcal{H}_0 h-dominates itself, because h-domination is reflexive (h-domination is satisfied by constraints that are in the same stratum in both hierarchies). Consider some constraint \mathbb{C} in some hierarchy \mathcal{H} . \mathbb{C} is either in the top stratum of \mathcal{H} , and thus in the same stratum as in \mathcal{H}_0 , or it is in some lower stratum of \mathcal{H} , and thus in a lower stratum than in \mathcal{H}_0 . Therefore, \mathcal{H}_0 h-dominates all hierarchies. \square

6.2 The Target Stratified Hierarchy

(52) **Def.** The *h-dominant target stratified hierarchy*, or simply the ‘target,’ for a language L generated by a total ranking, is denoted \mathcal{H}_L , and is defined as follows. The top stratum of the target contains precisely those constraints which never assess uncanceled marks to any optimal structural description in L . The second stratum consists of precisely those constraints which assess uncanceled marks only to optimal parses relative to competitors which are assessed at least one uncanceled mark by a constraint in the top stratum. Each stratum consists of precisely those constraints which (a) cannot occur higher in the target, and (b) only assess uncanceled marks to optimal parses relative to competitors assessed an uncanceled mark by at least one constraint ranked higher in the target.

(53) **Lemma** For any L generated by a total ranking, \mathcal{H}_L exists and is unique.

Proof Existence follows from the definition, and the assumption that L is generated by at least one total ranking of the constraints. The top stratum of \mathcal{H}_L is guaranteed to contain at least the constraint ranked highest in the total ranking. Among the constraints not placed in the top stratum of \mathcal{H}_L , one dominates all the remaining

others in the total ranking, and is thus guaranteed to meet the requirements for placement in the second stratum. The same logic, applied to subsequent strata, shows that all of the constraints will be placed in a stratum in \mathcal{H}_L .

Uniqueness is guaranteed because a constraint cannot meet the requirements for placement in more than one stratum in the hierarchy, because meeting the requirements for one stratum automatically disqualifies it for any lower strata. \square

(54) **Lemma** Each constraint \mathbb{C} with offset $n > 0$ in \mathcal{H}_L for a language generated by a total ranking has the following property. There must exist an optimal description *winner* with a competing suboptimal description *loser* such that \mathbb{C} assesses an uncanceled mark to *winner*, *loser* is assessed an uncanceled mark by a constraint \mathbb{C}_{n-1} with offset precisely $n-1$, and *loser* is not assessed any uncanceled marks by any constraints with offset less than $n-1$.

Proof Consider some constraint \mathbb{C}_n with offset $n > 0$ in target \mathcal{H}_L . Suppose, to the contrary, that no such pair *loser/winner* exists for \mathbb{C}_n . Recall that if \mathbb{C}_n assesses an uncanceled mark to an optimal description relative to some suboptimal competitor, it must be dominated by some other constraint which assesses an uncanceled mark to the suboptimal competitor, for otherwise the optimal description would not be more harmonic, and the correct language would not be generated.

One possibility is that \mathbb{C}_n never assesses an uncanceled mark to any optimal description. But then it would have offset 0 in \mathcal{H}_L , contradicting the assumption that it has offset greater than 0.

The other possibility is that for any *winner* assessed an uncanceled mark by \mathbb{C}_n relative to some *loser*, *loser* is assessed an uncanceled mark by a constraint with offset smaller than $n-1$. But then \mathbb{C}_n could be placed one stratum higher in \mathcal{H}_L , with resulting offset $n-1$, and the resulting hierarchy would generate the same language,

contradicting the fact that by definition every constraint is ranked as high as possible in \mathcal{H}_L .

Hence the supposition must be false, and an appropriate pair must exist. \square

(55) **Theorem** For any language L generated by a total ranking, \mathcal{H}_L generates L . \mathcal{H}_L also has the property that each constraint is ranked as high as possible; that is, \mathcal{H}_L h-dominates every total ranking \mathcal{H}' which generates L .

Proof Consider some description *winner* in L , and any competitor *loser* with non-identical marks. If *winner* has no uncanceled marks, it is more harmonic than *loser*. If *winner* has an uncanceled mark, then its corresponding constraint must be dominated in \mathcal{H}_L by some constraint assessing an uncanceled mark to *loser*, by the definition of \mathcal{H}_L . So \mathcal{H}_L generates L .

For the second part, consider a total ranking \mathcal{H}' which generates L .

Consider a constraint \mathbb{C} with offset 0 in \mathcal{H}' . \mathbb{C} must not assess an uncanceled mark to any optimal description in the language; otherwise, \mathcal{H}' would not generate the language. Therefore, \mathbb{C} must have offset 0 in \mathcal{H}_L . It follows that \mathbb{C} 's offset in \mathcal{H}_L is $\leq \mathbb{C}$'s offset in \mathcal{H}' , as both are 0.

Assume that each constraint with offset $\leq n$ in \mathcal{H}' is in the same or higher stratum in \mathcal{H}_L . Consider the constraint \mathbb{C}_{n+1} with offset $n+1$ in \mathcal{H}' . For any pair of an optimal description *winner* with a suboptimal competitor *loser*, if \mathbb{C}_{n+1} assesses an uncanceled mark to *winner*, *loser* must be assessed an uncanceled mark by a constraint \mathbb{C} with offset $\leq n$ in \mathcal{H}' (that is, $\mathbb{C} \gg \mathbb{C}_{n+1}$ in \mathcal{H}'); otherwise, \mathcal{H}' would not generate the language. By hypothesis, any constraint with offset $\leq n$ in \mathcal{H}' has offset $\leq n$ in \mathcal{H}_L . Therefore, \mathbb{C}_{n+1} has offset $\leq n+1$ in \mathcal{H}_L .

By mathematical induction, every constraint in \mathcal{H}' is in the same or higher stratum in \mathcal{H}_L . It follows directly that every constraint is in the same or lower stratum

in \mathcal{H}' than in \mathcal{H}_L . Therefore, target \mathcal{H}_L h-dominates \mathcal{H}' . \square

(56) **Corollary** Every total ranking which is a refinement of \mathcal{H}_L generates L .

Proof By the definition of \mathcal{H}_L (52), for every loser/winner pair of L , each uncanceled winner mark is dominated, with respect to \mathcal{H}_L , by an uncanceled loser mark. By the definition of refinement (49), any refinement of \mathcal{H}_L preserves all such domination relations of \mathcal{H}_L . Therefore, any refinement which is a total ranking generates L . \square

6.3 Constraint Demotion

(57) **Def.** The mark cancellation procedure, $\text{MarkCancel}(\text{marks}(\text{loser}), \text{marks}(\text{winner}))$, is:

For each occurrence of $*\mathbb{C}$ in both $\text{marks}(\text{loser})$ and $\text{marks}(\text{winner})$

Remove that occurrence of $*\mathbb{C}$ from both lists

Return the lists as $(\text{marks}'(\text{loser}), \text{marks}'(\text{winner}))$

(58) **Def.** The constraint demotion procedure, $\text{CD}((\text{marks}'(\text{loser}), \text{marks}'(\text{winner})), \mathcal{H})$, is:

Set \mathcal{H}' to \mathcal{H}

Find the constraint \mathbb{C}_l with a mark in $\text{marks}'(\text{loser})$ ranked highest in \mathcal{H}'

For each \mathbb{C}_w with a mark in $\text{marks}'(\text{winner})$

If \mathbb{C}_l does not dominate \mathbb{C}_w in \mathcal{H}' ,

demote \mathbb{C}_w to the stratum immediately below \mathbb{C}_l

Return \mathcal{H}'

(59) **Lemma** The hierarchy output by CD is h-dominated by the input hierarchy.

Proof Because constraint demotion only demotes constraints, each constraint is in either the same or lower stratum in the output hierarchy than it was in the input hierarchy. \square

(60) **Lemma** If the input hierarchy h-dominates \mathcal{H}_L , so does the output hierarchy.

Proof This holds because CD will never demote a constraint lower than necessary. Let \mathbb{C}_w

be some constraint demoted by CD. Then there is a mark-data pair ($marks'(loser)$, $marks'(winner)$) requiring that C_w be dominated by one of the constraints assessing uncanceled marks to *loser*. Let C_l be the one with the smallest offset (highest ranked) in \mathcal{H} , the input hierarchy, and let n denote its offset. By assumption, \mathcal{H} h-dominates \mathcal{H}_L , so C_l in \mathcal{H}_L has offset $\geq n$. Thus, every constraint assessing an uncanceled mark to *loser* must have offset $\geq n$. Therefore, C_w must have offset at least $n+1$ in \mathcal{H}_L . CD demotes C_w to the stratum immediately below the one containing C_l , so C_w has offset $n+1$ in the resulting hierarchy. Thus, C_w has offset in the output hierarchy less than or equal to its offset in \mathcal{H}_L , guaranteeing that the output hierarchy h-dominates \mathcal{H}_L . \square

(61) **Def.** An *informative pair* for a hierarchy \mathcal{H}' is a mark-data pair that, when given as input to CD along with \mathcal{H}' , causes at least one demotion to occur. The property of being informative is jointly determined by the mark-data pair and the hierarchy being evaluated.

(62) **Def.** The *h-distance* between a hierarchy \mathcal{H}_1 and a hierarchy \mathcal{H}_2 h-dominated by \mathcal{H}_1 is the sum, over all constraints C , of the difference between the offset of C in \mathcal{H}_1 and in \mathcal{H}_2 .

(63) **Lemma** Suppose the input hierarchy h-dominates \mathcal{H}_L . The *h-distance* between the output hierarchy and \mathcal{H}_L is decreased by at least one (from the h-distance between the input hierarchy and \mathcal{H}_L) for each demotion.

Proof By lemma (59), the input hierarchy h-dominates the output hierarchy. Let C be a constraint that is demoted, with offset n in the input hierarchy, offset m in the output hierarchy, and offset t in \mathcal{H}_L . C is demoted, so $m > n$. By lemma (60), the output hierarchy h-dominates \mathcal{H}_L , so $t \geq m > n$. Therefore, $(t - m) < (t - n)$, so the contribution

of \mathbb{C} to h-distance is smaller for the output hierarchy. Thus, the output hierarchy h-distance is at least one less for each constraint demoted. \square

(64) **Lemma** Let N be the number of constraints. The h-distance from \mathcal{H}_0 to \mathcal{H}_L cannot exceed $\frac{1}{2}(N-1)N$.

Proof By lemma (51), \mathcal{H}_0 h-dominates every hierarchy, and therefore must h-dominate \mathcal{H}_L . The greatest h-distance will be when \mathcal{H}_L is a totally ranked hierarchy. The furthest constraint from the top stratum will be the one in the bottom stratum, which has offset $(N-1)$. The next lowest constraint has offset $(N-2)$, and so forth. Thus, the h-distance will be:

$$(N-1) + (N-2) + \dots + 1 + 0$$

which is precisely $\frac{1}{2}(N-1)N$. \square

(65) **Theorem** Starting with \mathcal{H}_0 and repeatedly presenting CD with mark-data pairs, the target hierarchy \mathcal{H}_L is converged upon after at most $\frac{1}{2}(N-1)N$ informative pairs.

Proof By lemma (63), each informative pair reduces the h-distance by at least one. Therefore the target hierarchy is converged upon after a number of informative pairs that is at most the h-distance between \mathcal{H}_0 and the target. Lemma (64) guarantees that this distance is at most $\frac{1}{2}(N-1)N$. \square

6.4 Extension to Arbitrary Initial Hierarchies

\mathcal{H}_0 here denotes some arbitrary initial hierarchy, and K denotes the maximal offset in \mathcal{H}_0 (one less than the number of strata, since the offset of the top stratum is zero). A slight elaboration of CD is made: if the last constraint in a stratum gets demoted, for bookkeeping purposes the empty stratum is retained in the hierarchy. That way, all the constraints which are not demoted will have their offset (stratum number) unaffected (empty strata can occur when starting with an arbitrary initial hierarchy, but not when starting with all constraints top-

ranked).

With an arbitrary initial hierarchy, the target hierarchy \mathcal{H}_L is not the exact goal of learning. Instead, it is used to define a new hierarchy $\mathcal{H}^\#$ which CD approaches and can never go beyond. As before, this bound on demotion makes it possible to compute a limit to the number of demotions possible before a correct solution is reached.

(66) **Def.** The offset of constraint \mathbb{C} in hierarchy \mathcal{H} is denoted $v(\mathbb{C}, \mathcal{H})$.

(67) **Def.** The *lower bounding hierarchy* for L , $\mathcal{H}^\#$, is the stratified hierarchy in which the first K strata are empty, and then a copy of \mathcal{H}_L runs from the stratum with offset K down. That is, the offset of any constraint \mathbb{C} in $\mathcal{H}^\#$ is K more than its offset in \mathcal{H}_L :

$$v(\mathbb{C}, \mathcal{H}^\#) = K + v(\mathbb{C}, \mathcal{H}_L)$$

(68) **Def.** During the operation of the CD algorithm, let D denote the h-distance (62) between the algorithm's current stratified hierarchy \mathcal{H} and the lower bounding hierarchy $\mathcal{H}^\#$:

$$D \equiv \sum_{\mathbb{C}} [v(\mathbb{C}, \mathcal{H}^\#) - v(\mathbb{C}, \mathcal{H})]$$

(69) **Lemma** For each loser/winner pair of L in which $*\mathbb{C}$ is an uncanceled winner mark, \mathbb{C} is dominated in \mathcal{H}_L by some \mathbb{C}' such that $*\mathbb{C}'$ is an uncanceled loser mark.

Proof \mathcal{H}_L correctly accounts for all the data in L (55) so in any loser/winner pair each uncanceled winner mark must be dominated by an uncanceled loser mark. \square

(70) **Lemma** The CD algorithm never demotes \mathbb{C} below the stratum with offset $v(\mathbb{C}, \mathcal{H}^\#)$.

Proof By induction on $v(\mathbb{C}, \mathcal{H}_L)$, the offset of \mathbb{C} in \mathcal{H}_L .

Let \mathbb{C} be a constraint in the top stratum of \mathcal{H}_L , with $v(\mathbb{C}, \mathcal{H}_L) = 0$. Then \mathbb{C} will never be demoted by CD. For, by the definition of CD, such demotion would require that $*\mathbb{C}$ be an uncanceled mark of a winner, which is impossible for a

constraint in the top stratum of \mathcal{H}_L . Thus for each constraint C with $v(C, \mathcal{H}_L) = 0$, C is never demoted by CD, and remains in whatever stratum it happens to occupy in \mathcal{H}_0 . The lowest stratum in \mathcal{H}_0 has offset K , so a constraint C with $v(C, \mathcal{H}_L) = 0$ ends up where it starts, in a stratum with offset at most $K + 0 = K + v(C, \mathcal{H}_L) \equiv v(C, \mathcal{H}^\#)$. This establishes the base case for the induction.

Now for the inductive step we assume that for all constraints C with $v(C, \mathcal{H}_L) < k$, the CD algorithm never demotes C below offset $v(C, \mathcal{H}^\#)$. Let C be a constraint with $v(C, \mathcal{H}_L) = k$. By Lemma (69), for each loser/winner pair in which $*C$ is an uncanceled winner mark, C is dominated in \mathcal{H}_L by some C' such that $*C'$ is an uncanceled loser mark. This implies that C' has a lower offset than C in \mathcal{H}_L , and it follows that $v(C', \mathcal{H}_L) < k$. Thus, by the inductive hypothesis, C' is demoted to a stratum no lower than

$$v(C', \mathcal{H}^\#) \equiv K + v(C', \mathcal{H}_L) \leq K + k - 1 = v(C, \mathcal{H}^\#) - 1.$$

Each time C is demoted (due to an error on a loser/winner pair in which $*C$ is an uncanceled winner mark), it is demoted to just below the highest stratum containing a C' such that $*C'$ is an uncanceled loser mark. We are guaranteed by induction that such a C' is to be found among the top $v(C, \mathcal{H}^\#) - 1$ strata, so C cannot be demoted below stratum $v(C, \mathcal{H}^\#)$. And C can't *start* below this stratum either, since $v(C, \mathcal{H}^\#) \geq K$, and no constraint starts lower than K . This completes the inductive step. \square

(71) **Lemma** D can never go negative. D monotonically decreases during the execution of CD.

Proof That D is never negative now follows immediately since all the terms in the sum defining it are non-negative, by Lemma (70). That D monotonically decreases during the execution of CD is obvious, since CD only demotes constraints, which can only decrease D . \square

(72) **Theorem** CD converges to a hierarchy generating L after no more than $N(N-1)$ informative examples.

Proof By Lemma (71), the number of demotions can't exceed the initial value of D : each demotion decreases D by at least 1, and D can never go negative. How large can the initial value of D be? In the worst case, the final hierarchy is totally-ranked and the initial hierarchy is the exact inverse of the final hierarchy. In this case, $K = N-1$, and the initially-top-ranked constraint must be demoted $2K$ strata, the constraint below it must be demoted $2(K-1)$ strata, and so on, with the initially lowest-ranked constraint not being demoted at all, and ending up top-ranked. The total number of strata passed through, D , in this worst case is thus twice the corresponding sum in the case where all constraints are initially top-ranked (64):

$$\begin{aligned} 2K + 2(K-1) + \dots + 0 &= 2(N-1) + 2(N-2) + \dots + 0 \\ &= 2[(N-1) + (N-2) + \dots + 0] = 2[N(N-1)/2] = N(N-1) \end{aligned}$$

After the last demotion, at latest after $N(N-1)$ informative examples, the fact that there are no more demotions means that there are no more remaining informative examples. If the hierarchy did not generate L , then there would exist further informative examples, and by assumption the learner would receive them and make further demotions. \square

6.5 Error-Driven Constraint Demotion

(73) **Def.** The Error-Driven Constraint Demotion algorithm $\text{EDCD}(PositiveData, \mathcal{H})$, is:

For each description *winner* in *PositiveData*:

Set *loser* to be the optimal description assigned by \mathcal{H} to I , the underlying form of *winner*.

If *loser* is identical to *winner*, keep \mathcal{H} ;

Else:

- apply Mark Cancellation, getting ($marks'(loser)$, $marks'(winner)$)
- apply Constraint Demotion to ($marks'(loser)$, $marks'(winner)$) and \mathcal{H}
- adopt the new hierarchy resulting from demotion as the current hierarchy

(74) **Theorem** Error-Driven Constraint Demotion converges to a hierarchy consistent with all positive evidence from L , and converges after at most $N(N-1)$ informative examples.

Proof The theorem follows directly from theorem (72), and the fact that, for any observed winner, if the learner's hypothesized hierarchy does not find the winner optimal, production-directed parsing will produce a competitor guaranteed to result in at least one demotion when CD is applied. \square

Theorem (74) states that EDCD converges to “a hierarchy consistent with all positive evidence from L ,” rather than “a hierarchy generating L ,” for the following reason: if different grammars have subset relations, where the language generated by one grammar is a strict subset of the language generated by another, then EDCD, when given positive evidence from a subset language, may converge on a superset language, consistent with all the positive evidence but not generating the same language. The outcome may depend on the starting hierarchy, among other factors; see section 4.3. This subset sensitivity is a consequence of the error-driven nature of EDCD combined with only positive evidence; if the appropriate *loser/winner* pairs were obtained, the Constraint Demotion principle itself, properly applied, would guarantee convergence to the (correct) subset language. For further discussion, see Smolensky 1996c.

Notes

We are greatly indebted to Alan Prince, whose challenges, insights, and suggestions have improved nearly every page of this paper. We are also grateful for important comments and encouragement from John McCarthy, Géraldine Legendre, Jane Grimshaw, Bruce Hayes, Linda Lombardi, Luigi Burzio, Clara Levelt, Vieri Samek-Lodovici, and two anonymous *LI* reviewers. For helpful questions and suggestions we thank the participants and organizers of the First Rutgers Optimality Theory Workshop (ROW-1), the Las Cruces meeting of the Association for Computational Linguistics, the University of Maryland Learnability Workshop, the MIT Workshop on Optimality in Syntax, and the Royaumont Workshop on Approaches to Phonology. We are grateful to the Institute of Cognitive Science at the University of Colorado at Boulder for partial support of ROW-1 where some of this work was first presented (and reported in Tesar and Smolensky 1993; see also Tesar and Smolensky 1995). B. Tesar acknowledges the support of an NSF Graduate Fellowship, P. Smolensky the support of a Guggenheim Fellowship, and both authors the support of NSF grant IRI-9213894.

1. Even a UG with only 20 binary parameters admits over a million grammars.
2. The PARSE and FILL constraints defined here constitute the parse/fill model of faithfulness in OT. The parse/fill model is used in P&S, and in much of the OT literature. More recently, an alternative conception of faithfulness has been proposed, the correspondence model (McCarthy and Prince 1995). The distinctions between the two are irrelevant to the learnability work described in this paper; the same results hold for OT grammars adopting the correspondence model of faithfulness.
3. If two candidates are assessed exactly the same set of marks by the constraints, then they are equally harmonic (regardless of the constraint ranking). If they tie as the most harmonic candidates for an input, then the two outputs are both optimal, with the interpretation of free

alternation. $a \sim b$ denotes that a and b have equal Harmony. In practice, it is rather rare to have more than one optimal output for an input.

4. Determining that this candidate is optimal requires demonstrating that it is more harmonic than *any* of the infinitely many competing candidates: see P&S:§7.3 for one technique.

5. There is a choice to be made in exactly how to apply this approach to a set of observed, optimal structural descriptions, resulting in two variations. Because applying CD to a single mark-data pair does not ensure that the observed parse (the winner) is yet optimal with respect to all candidates (not just the loser), the learner could re-parse the input according to the new constraint ranking. If the resulting parse is different from the winner, the new parse may be used to create a new mark-data pair, to which CD is applied. This process could be repeated until the learner's hierarchy selects the winner as the optimal description. This allows the learner to extract more information out of a single winner, at the cost of greater processing dedicated to each winner. The decision here is whether or not to repeatedly apply parsing and CD to a single winner.

6. And future OT work on syntax is likely to take on syntactic properties of lexical items, such as argument structure, where related acquisition issues may be expected to arise.

7. The formulation of P&S is slightly different: only underlying forms which *optimally* surface as φ are considered. In our version, *all* structures which surface as φ are considered, because we want to use lexicon optimization as part of a learning process; when the current grammar is incorrect, there may well be no underlying forms which optimally surface as φ . Thus our formulation of lexicon optimization is 'robust' in the same sense as our formulation of interpretive parsing: when there is no grammatical option, the maximal-Harmony (but ungrammatical) structure is used nonetheless.

8. This conclusion assumes (for both OT and P&P) that enough information is available to the learner to permit overt structures to be evaluated against a particular hypothesized

grammar. Since the lexicon is unbounded, this enumeration approach is not even in principle applicable to the acquisition of lexical information such as phonological underlying forms or syntactic argument structure.

9. Under the normal definitions of trigger, a single datum can be a trigger for more than one parameter, but is such independently. In such a case, the datum would not be interpreted as expressing any relationship between the values of the two parameters.

References

- Ackema, Peter, and Ad Neeleman. In press. Optimal questions. In *Proceedings of the Workshop on Optimality in Syntax—“Is the best good enough?”* Cambridge, Mass: MIT Press and MIT Working Papers in Linguistics.
- Angluin, Dana. 1978. Inductive inference of formal languages from positive data. *Information and Control* 45:117–135.
- Baum, L. E., and T. Petrie. 1966. Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics* 37, 1559–1563.
- Bahl, L. R., F. Jelinek and R. L. Mercer 1983, A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-5, 179–190.
- Brown, P. F., J. Cocke, S. A. Della Pietra, V. J. Della Pietra, F. Jelinek, J. D. Lafferty, R. L. Mercer, and P. S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics* 16.
- Berwick, Robert. 1986. *The acquisition of syntactic knowledge*. MIT Press, Cambridge, MA.
- Billings, Loren, and Catherine Rudin. 1994. Optimality and superiority: A new approach to overt multiple *wh* ordering. In *Proceedings of the Third Symposium on Formal Approaches to Slavic Linguistics*.
- Clark, Eve. 1987. The principle of contrast: A constraint on language acquisition. In Brian MacWhinney, ed., *Mechanisms of language acquisition*, Hillsdale, NJ: Erlbaum.
- Clark, Robin. 1990. *Papers on learnability and natural selection*. Technical reports in formal and computational linguistics, No. 1, Université de Genève.
- Clements, G. N and S.J. Keyser. *CV phonology*. Cambridge, MA: MIT Press.

- Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38.
- Demuth, Katherine. 1995. Markedness and the development of prosodic structure. In Jill Beckman, ed., *NELS 25*, 13–25. Amherst, MA: GLSA, University of Massachusetts.
- Dresher, B. Elan, and Jonathan Kaye. 1990. A computational learning model for metrical phonology. *Cognition* 34, 137-195.
- Gibson, Edward, and Ken Wexler. 1994. Triggers. *Linguistic Inquiry*, 25(4).
- Gnanadesikan, Amalia. 1995. Markedness and faithfulness constraints in child phonology. Ms., October 1995.
- Grimshaw, Jane. 1993. Minimal projection, heads, and inversion. Ms. Rutgers University, New Brunswick, NJ.
- Grimshaw, Jane. To appear. Projection, heads, and optimality. *Linguistic Inquiry*.
- Grimshaw, Jane, and Vieri Samek-Lodovici. 1995. Optimal subjects. In *Papers in Optimality Theory*, eds. J. Beckman, L. W. Dickey, and S. Urbanczyk. University of Massachusetts occasional papers 18. Amherst, MA: GLSA, University of Massachusetts.
- Grimshaw, Jane, and Vieri Samek-Lodovici. In press. Optimal subjects and subject universals. In *Proceedings of the Workshop on Optimality in Syntax—“Is the best good enough?”* Cambridge, Mass: MIT Press and MIT Working Papers in Linguistics.
- Hamburger, Henry, and Ken Wexler. 1973. Identifiability of a class of transformational grammars. In K. J. J. Hintikka, J. M. E. Moravcsik, and P. Suppes, eds., *Approaches to natural language*. Dordrecht: Reidel.

- Haussler, David. 1996. Probably Approximately Correct learning and decision-theoretic generalizations. In Paul Smolensky, Michael C. Mozer, and David E. Rumelhart, eds., *Mathematical perspectives on neural networks*. Hillsdale, NJ: Lawrence Erlbaum Publishers.
- Hinton, Geoffrey. 1989. Connectionist learning procedures. *Artificial Intelligence* 40:185–234.
- Jakobson, Roman. 1962. *Selected writings 1: Phonological studies*. The Hague: Mouton.
- Kearns, Michael, and Umesh Vazirani. 1994. *An introduction to computational learning theory*. Cambridge, MA: MIT Press.
- Legendre, Géraldine, William Raymond, and Paul Smolensky. 1993. Analytic typology of case marking and grammatical voice. *Proceedings of the Berkeley Linguistics Society*, 19.
- Legendre, Géraldine, Paul Smolensky, and Colin Wilson. In press. When is less more? Faithfulness and minimal links in *wh*-chains. In *Proceedings of the Workshop on Optimality in Syntax—“Is the best good enough?”* Cambridge, Mass: MIT Press and MIT Working Papers in Linguistics.
- Legendre, Géraldine, Colin Wilson, Paul Smolensky, Kristin Homer, and William Raymond. 1995. Optimality and *wh*-extraction. In *Papers in Optimality Theory*, eds. J. Beckman, L. W. Dickey, and S. Urbanczyk. University of Massachusetts Occasional Papers 18. Amherst, MA: GLSA, University of Massachusetts.
- Levelt, Clara. 1995. Unfaithful kids: Place of Articulation patterns in early child language. Paper presented at the Department of Cognitive Science, Johns Hopkins University, Baltimore, Md., September, 1995.

- McCarthy, John, and Alan Prince. 1995. Faithfulness and reduplicative identity. In *Papers in Optimality Theory*, eds. J. Beckman, L. W. Dickey, and S. Urbanczyk. University of Massachusetts Occasional Papers 18. Amherst, MA: GLSA, University of Massachusetts.
- Nádas, Arthur, and Robert. L. Mercer. 1996. Hidden Markov Models and some connections with artificial neural nets. In Paul Smolensky, Michael C. Mozer, and David E. Rumelhart, eds., *Mathematical perspectives on neural networks*. Hillsdale, NJ: Lawrence Erlbaum Publishers.
- Niyogi, Partha, and Robert C. Berwick. 1993. Formalizing triggers: A learning model for finite spaces. A.I. Memo No. 1449. Artificial Intelligence Laboratory, MIT.
- Pinker, Steven. 1986. Productivity and conservatism in language acquisition. In *Language learning and concept acquisition*, ed. W. Demopoulos and A. Marras, Ablex, Norwood, NJ.
- Pitt, Leonard, and Leslie Valiant 1988. Computational limitations on learning from examples. *Journal of the ACM* 35, 965-984.
- Prince, Alan. 1993. Internet communication, September 26.
- Prince, Alan and Paul Smolensky. 1991. Notes on connectionism and Harmony Theory in linguistics. Technical Report CU-CS-533-91. Department of Computer Science, University of Colorado, Boulder.
- Prince, Alan, and Paul Smolensky. 1993. *Optimality Theory: Constraint interaction in generative grammar*. Ms. Rutgers University, New Brunswick, NJ, and University of Colorado, Boulder. To appear as Linguistic Inquiry Monograph, MIT Press, Cambridge, MA.

- Pulleyblank, Douglas, and William J. Turkel. In press. The logical problem of language acquisition in Optimality Theory. In *Proceedings of the Workshop on Optimality in Syntax—“Is the best good enough?”* Cambridge, Mass: MIT Press and MIT Working Papers in Linguistics.
- Pulleyblank, Douglas, and William J. Turkel. 1995. Traps in constraint ranking space. Paper presented at Maryland Mayfest 95: Formal Approaches to Learnability.
- Safir, Ken. 1987. Comments on Wexler and Manzini. In Thomas Roeper and Edwin Williams, eds., *Parameter setting*, 77–89. Dordrecht: Reidel.
- Samek-Lodovici, Vieri. 1995. Topic and focus effects on clause structure: An Optimality Theoretic analysis. Doctoral Dissertation, Department of Linguistics, Rutgers University.
- Smolensky, Paul. 1996a. Statistical perspectives on neural networks. In Paul Smolensky, Michael C. Mozer, and David E. Rumelhart, eds., *Mathematical perspectives on neural networks*. Hillsdale, NJ: Lawrence Erlbaum Publishers.
- Smolensky, Paul. 1996b. On the comprehension/production dilemma in child language. *Linguistic Inquiry* 27.
- Smolensky, Paul. 1996c. The initial state and ‘richness of the base’ in Optimality Theory. Technical Report, Department of Cognitive Science, Johns Hopkins University.
- Tesar, Bruce. 1994. Parsing in Optimality Theory: A dynamic programming approach. Technical Report CU-CS-714-94, April 1994. Department of Computer Science, University of Colorado, Boulder.
- Tesar, Bruce. 1995a. Computing optimal forms in Optimality Theory: Basic syllabification. Technical Report CU-CS-763-95, February 1995. Department of Computer Science, University of Colorado, Boulder.

- Tesar, Bruce. 1995b. Computational Optimality Theory. Doctoral Dissertation, Department of Computer Science, University of Colorado at Boulder.
- Tesar, Bruce. In press. Error-driven learning in Optimality Theory via the efficient computation of optimal forms. In *Proceedings of the Workshop on Optimality in Syntax—“Is the best good enough?”* Cambridge, Mass: MIT Press and MIT Working Papers in Linguistics.
- Tesar, Bruce. In preparation a. Robust interpretive parsing with alignment constraints.
- Tesar, Bruce. In preparation b. An iterative strategy for language learning.
- Tesar, Bruce, and Paul Smolensky. 1993. The learnability of Optimality Theory: An algorithm and some basic complexity results. Technical Report CU-CS-678-93, Oct. 1993. Department of Computer Science, University of Colorado at Boulder.
- Tesar, Bruce, and Paul Smolensky. 1995. The learnability of Optimality Theory. *Proceedings of the Thirteenth West Coast Conference on Formal Linguistics*, 122–137.
- Wexler, Kenneth. 1981. Some issues in the theory of learnability. In C. Baker and J. McCarthy, eds., *The logical problem of language acquisition*, 30–52. Cambridge, MA: MIT Press.
- Wexler, Kenneth, and Peter Culicover. 1980. *Formal principles of language acquisition*. Cambridge, MA: MIT Press.
- Wexler, Kenneth, and M. Rita Manzini. 1987. Parameters and learnability in binding theory. In Thomas Roeper and Edwin Williams, eds., *Parameter setting*. Dordrecht: Reidel.