Learnability in Optimality Theory (long version)

Bruce Tesar & Paul Smolensky

October 1996

Technical Report
JHU-CogSci-96-3

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

TR-Request@cogsci.jhu.edu

Learnability in Optimality Theory

Bruce Tesar and Paul Smolensky

Contents

How, exactly, does a theory of grammar bear on questions of learnability? Restrictions on what counts as a possible human language restrict the search space of the learner, and would thus appear to aid learnability. But beyond this very coarse observation, what can be said about exactly how learnability is enhanced by the particular kinds of restrictions imposed by a particular theory of grammar?

The central claim of this paper is that the character of the restrictions imposed by Optimality Theory ('OT,' Prince and Smolensky 1991, 1993) have demonstrable and significant consequences for central questions of learnability. Our goal in this work is formal results that are independent of the substantive assumptions of an OT analysis of any particular linguistic domain: results that apply equally to phonology, syntax, or any other domain which might admit an OT grammar. These are results on the learnability consequences of OT's general theory of cross-linguistic variation.

In this paper we offer the following:

(1) Contributions

a. A decomposition of the learning problem into two subproblems: parsing and grammar learning. This decomposition exploits the *optimization* character of the grammatical structure provided by OT (Sections 2 and 8);

b. A detailed analysis of one component of this decomposition: grammar learning from primary data consisting of grammatical structural descriptions. This analysis heavily exploits the special structure of UG under OT. Specifically, we offer a grammar learning principle, *Constraint Demotion* (Section 4), with proofs of correctness and efficiency (Section 5) of the learning procedures it yields.

c. Preliminary exploration of the implications of these results for:
   - ◆ the initial state and OT analysis of child language (Section 7);
   - ◆ learning from overt data only (with a miniature concrete example; Section 8);
   - ◆ acquisition of the phonological lexicon (Section 9).

We also review the principles of OT (Section 3) in sufficient detail to provide a self-contained framework within which to carry out our analysis. Our primary results concern grammar learning (1b): learnability theorems, presented in Sections 5 and 7.1, and proved in the Appendix, Section 11. The rest of the paper begins the exploration of the implications of the formal results, and situates them in a larger perspective on learning.

The table in (2) provides a compact summary of the paper, indicating the unifying structure deriving from OT's characterization of grammar through optimization, structure which underlies the arguments in virtually every problem area we examine here.

(2) Paper summary

Issue	Proposal	Role of Optimization	Sec.
How can grammars be learned from structural descriptions when a learner needs the grammar to compute structural descriptions?	Successive approximation, iterating between ‘robust interpretive parsing’ to assign structure, and grammar learning from assigned structure.	Iterative optimization in model-based learning.	2, 3
What is the relevant grammar-learning subproblem?	Given correct structural descriptions, find the correct grammar: constraint ranking.		
How can constraints be ranked, given grammatical structures?	Constraint Demotion: Demote constraint violations of grammatical structures below those of competing structures.	Grammatical structures are by definition optimal.	4
Is convergence to a correct ranking guaranteed?	Yes.	Optimality is determined by pairwise constraint rankings.	5
How much data is needed?	No more than $N(N–1)$ informative examples.		
How can constraints be evaluated, given only overt learning data?	Robust interpretive parsing.	Given overt form, and lexicon, optimize structure.	8
How can informative competitors be found?	Production-directed parsing.	Given input form, optimize structure.	6
How must the initial state be structured for learnability of ‘subset languages’?	Structural constraints dominate FAITHFULNESS constraints.	Optimal/unmarked inventories produce gaps in data.	7
How can the lexicon of phonological inputs be acquired?	Paradigm-level lexicon optimization.	Given overt form, optimize structure and input.	9
Does interaction of grammatical principles impede learnability?	No, it enhances it.	Optimality is defined via constraint interaction.	10

1. Learnability and the Structure of UG

Any grammatical theory that admits only a finite number of possible grammars reduces the search space of the learner from an infinite number of conceivable languages to a finite space, affording a dramatic asset to learnability. Or so it would seem. In fact, of course, such a grammatical theory really serves only to improve—albeit dramatically—the worst-case performance of the *least informed* search method of all, exhaustive search: at worst, in the finite case, search is guaranteed to terminate after all possible grammars have been examined; in the infinite case, search may continue forever.

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

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

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

Consider principles-and-parameters (P&P) theory in this regard; our discussion will be brief since our objective is not theory comparison but establishing results about Optimality Theoretic learnability. Two types of learnability research are useful as contrasts to what we provide below. The first is *cue learning*, exemplified by work such as Dresher and Kaye 1990. They adopt a particular parametrized space of grammars, and analyze the relationships between the overtly available forms and the parameter settings in great detail. They propose a specific learning algorithm to make use of the structure provided by a specific theory. The overall approach of Dresher and Kaye's work differs from our approach in several respects. Their analysis is entirely specific to their particular parametric system for metrical stress; a cue learning approach to a parametric grammar for some other aspect of linguistic theory, or even an alternative parametric analysis of metrical stress, would essentially require starting over from scratch. Further, their algorithm is evaluated entirely through computer simulations; despite

[1] This assertion does not contradict Gold's 1978 theorem, which entails that, in the absence of negative data, certain infinite language families—such as the regular ('finite state'), context-free, and families higher in the Chomsky hierarchy— are not learnable in the limit. To take an extreme example, consider languages over a single symbol a, writing in the obvious way $aa \equiv a^2$, etc. Then consider the following infinite family of infinite languages: $\{\{a, a^2, a^3, \ldots\}; \{a^2, a^3, a^4, \ldots\}; \{a^3, a^4, a^5, \ldots\}, \cdots\}$. This family of languages is so well-struc- tured that learnability in the limit is almost trivial. Consider the following algorithm: starting with the hypothesis of the empty language $H_0 = \varnothing$, given the ith datum a^k, hypothesize the language $H_i \equiv H_{i-1} \cup \{a^k, a^{k+1}, a^{k+2}, \ldots\}$, that is, the language $\{a^m, a^{m+1}, a^{m+2}, \ldots\}$ where m is the smallest 'power' of a yet observed. Eventually, the smallest power admitted by the language will be observed, and from this point on the algorithm will always hypothesize the correct language. It is the structure, not the size, of the space of possible languages that is really critical.

the great specificity, they have little in the way formal analysis to validate the algorithm.

Our work is aimed at the goal of principles of learning general to the framework of Optimality Theory, principles which apply to all OT systems (not restricted even to, say, phonology). Such a goal is not likely to be reached all in one step. Our work to date addresses only the subproblem of learning the grammar from full structural descriptions. The algorithms we propose apply to all OT systems, and the validity of the algorithms, with regard to both correctness and tractibility, is ensured by theorems. These strong results on learnability from full structural descriptions provide the opportunity to achieve formal results on the more general problem of learning grammars from overtly available data, while applying to OT systems in general. Our proposal for pursuing such further results is presented in Section 2.

Another approach to learnability within P&P, quite different from cue learning, is represented in the work of Gibson and Wexler 1994 and Niyogi and Berwick 1993. The *triggering learning algorithm* (and its variations) is designed to learn grammars from overtly available data. Like our work, it applies to any instance of a general class of systems: in their case, the class of P&P systems. Further, Niyogi and Berwick 1993 provides formal analysis of the algorithms. However, this work differs from ours in a way that represents the opposite extreme from cue learning. The P&P algorithms are minimally informed by the grammatical theory; they treat the grammar only as a black box evaluating learning data as either analyzable or not, and involve either randomly flipping parameters in order to render an input analyzable (Gibson and Wexler's Triggering Learning Algorithm), or randomly flipping parameters without regard to immediate resulting analyzability (which, Niyogi and Berwick argue, can actually outperform the Triggering Learning Algorithm). These P&P learning algorithms are equally appropriate as an algorithm for learning parametrized grammars and as an algorithm for, say, training a neural network[2] (with binary weights) to classify radar images of submarines: if flipping a parameter (connection in the network) gives better classification of a submarine, flip it. These are simply generic search algorithms which employ no properties of the grammatical theory *per se*. Our approach, by contrast, makes explicit use of the special structure of OT. Further, the learnability results relating to triggers presume the existence of data which directly reveal individual parameter values, an assumption of unclear relevance to realistic grammars (see Frank and Kapur 1996); we discuss this further in Section 10.1. Finally, regardless of the availability of such triggering forms, these algorithms offer no guarantee of tractibility. In fact, the only result regarding time complexity is that the probability of learning the correct grammar increases towards 1.0 as the number of learning instances approaches infinity[3], leaving the possibility of doing even worse than brute-force enumeration.

These two approaches to learnability analysis within P&P, then, either: (i) use grammatical structure in the learning algorithm, but the structure of a *particular* parametric system, or (ii) develop general algorithms applicable to *any* P&P system, but algorithms which are so general they apply just as well to any non-grammatical parametrized system. This dichotomy of approaches is likely a

[2] By 'neural network' we will mean a collection of interconnected processing elements, each carrying an 'activation value' which influences that of connected elements via connections with differing numerical strengths. These strengths or weights are the parameters that distinguish different networks and determine which function they will compute, as they convert input activity (perhaps encoding the radar image of a submarine) into output activity (perhaps encoding a classification of the image as a certain type of submarine).

[3] This is the case even when triggering forms exist for all parameter settings, and those forms appear among those presented to the learner with some reasonable frequency.

consequence of the nature of P&P. A particular P&P system, like one for stress, has sufficient structure to inform a learning procedure (option i). But as a general theory of how grammars may differ (as opposed to how stress systems may differ), P&P provides little structure to exploit beyond the existence of a finite space for searching.

But the situation in Optimality Theory is quite different. This theory is reviewed in Section 3, but the immediately relevant claims of OT are these:

(3) OT in a nutshell

a. What is it that all languages have in common? *A set of constraints on well-formedness.*

b. How may languages differ? *Only in which constraints have priority in case of conflict.*

(Note that the constraints of (3a) are the same in all languages: they contain no parameters.) Unlike P&P, this is a theory of cross-linguistic variation with sufficient structure to enable grammatically-informed learning algorithms that are independent of substantive grammatical assumptions—this is the central result of the paper:

(4) Main Claim.

OT is a theory of UG that provides sufficient structure at the level of the grammatical framework itself to allow general but grammatically-informed learning algorithms to be formally defined and proved correct and efficient.

The algorithms we develop are procedures for learning the priority ranking of constraints which, by (3b), is all that distinguishes the grammar of a particular language. These are unquestionably grammar-learning algorithms, not generic search algorithms. Yet the structure that makes these algorithms possible is not the structure of a theory of stress, nor a theory of phonology: it is the structure defining any OT grammar: (3).

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

To begin the study of OT learnability, the problem to be solved must first be formulated rather precisely. This turns out to be a somewhat involved matter in itself: it is the topic of Section 2. This analysis requires one further concept from OT, which will be developed in Section 3.2; for now, the following synopsis will suffice:

(5) Harmony and Optimality

a. The grammar of a particular language is an evaluator of structural descriptions, assigning a degree of *Harmony* which (non-numerically) assesses the degree to which universal well-formedness conditions are met, taking into account language-particular constraint priorities. This provides the *harmonic ordering of forms*, ordering structural descriptions from maximal to minimal Harmony.

b. The grammatical forms of the language are the *optimal* ones: the well-formed structural description of an input is the one with maximal Harmony.

2. Decomposing the Language Learning Problem

Our OT learnability results address a grammar learning problem which must be carefully separated from a closely related but quite distinct problem. Defining and justifying this separation is the goal of this section.

2.1 Grammar Learning and Robust Interpretive Parsing

To begin, we must distinguish three elements:

(6) The players in order of their appearance

a. Overt part of grammatical forms: directly accessible to the learner.
b. Full structural descriptions: combine overt and non-overt, 'hidden,' structure.
c. The grammar: determines which structural descriptions are well-formed.

These three elements are all intimately connected, yet we propose to distinguish two subproblems, as schematically shown in Figure 1. (The term 'robust interpretive parsing' will mean parsing overt structure with a grammar, even when that structure is not grammatical according to the grammar; the importance of this will be discussed shortly.)

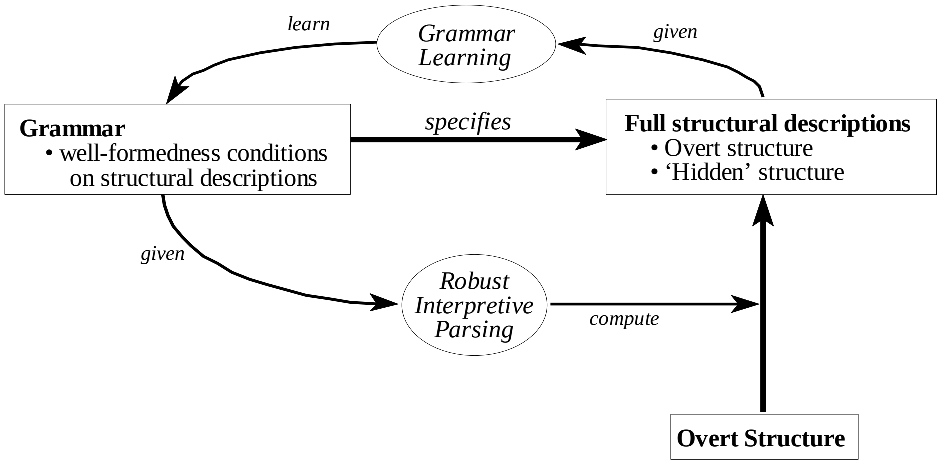
learn
Grammar Learning
given
Grammar
• well-formedness conditions on structural descriptions
specifies
Full structural descriptions
• Overt structure
• 'Hidden' structure
given
Robust Interpretive Parsing
compute
Overt Structure

Figure 1. Problem decomposition.

(7) Decomposition of the Problem

a. *Robust Interpretive Parsing:* mapping the overt part of a form into a full structural description, complete with all hidden structure—given a grammar.
b. *Learning the Grammar*—given a (robust) parser.

Ideally, a grammatical theory should provide sufficient structure so that procedures for both parsing and grammar learning can strongly exploit grammatical principles.

We propose that the problems of parsing and grammar learning be decoupled to some degree. Such separation does at first seem problematic, however. One of the central difficulties of language learning, of course, is that grammars refer crucially to non-overt, hidden structure. Let's take the acquisition of stress as an expository example. The problem, then, is that the grammatical principles concern metrical feet, yet these are hidden in the data presented to the learner: only the location of some stressed syllables is provided overtly. The learner can't learn the metrical grammar until she knows where the feet lie; but she can't know where the feet lie until she knows the grammar. It is the burden of this section to argue that, despite this conundrum, partial decoupling of the parsing and learning problems makes good sense.

2.2 Iterative Model-Based Solutions to the Problem of Learning Hidden Structure

The learner can't deduce the hidden structure in learning data 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 indeed; 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. Given an example of a '5', the computer needs to adapt its model for what makes a good '5'. But in many cases, the system isn't told which digit a given training example exemplifies: it is often impractical for all the digits in a huge training corpus to be hand-labeled as to what category they belong to, so the computer is forced to learn which digits *are* '5's at the same time as learning what makes a *good* '5' . The learner can't improve its model of what makes a well-formed '5' until it knows when it's seeing a '5': but it can't know when it's seen a '5' until it knows what makes a well-formed '5'.

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, a (hidden) phone [f], and (ii) what makes for a good [f].

This problem has been quite successfully addressed, in theory and practice. The necessary formalization is approximately as follows. A parametrized system (e.g., a neural network) 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. (As we see shortly, this model corresponds to the grammar in our problem). Given a hidden [f] in 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]s as being of type [f]. The general problem is usually formalized along the following lines:

(8) 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., 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 successful algorithms for solving this problem, the most important of which is the Expectation Maximization (EM) algorithm (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, which we will call *iterative model-based* learning algorithms, may be characterized in very general terms as follows:[4]

[4] This characterization corresponds most closely to the Viterbi algorithm; see, e.g., Nádas and Mercer 1996, Smolensky 1996a.

(9) Iterative model-based 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 (initially poor) model. This first step of the algorithm is performed on all the available data.
2. Now that we have deduced some hidden structure (initially incorrect), we use it to improve our model, in the second step of the algorithm. Since all the overt (acoustic) data has been connected to corresponding hidden (phonemic) structure, we can now improve the model, changing its parameters so that the imputed hidden structure optimally predicts the actual overt structure observed. (E.g., the model for hidden phone [f] is changed so that it now predicts as closely as possible the actual acoustic values in the data which have been identified as instances of [f].)

1′. Now that the model has been changed, it will assign different (generally more correct) hidden structure to the original overt data. So the algorithm goes back through the data, and executes Step 1 over again, re-assigning hidden structure.

2′. This new assignment of hidden structure permits Step 2 to be repeated, leading to a new (generally improved) model. And so the algorithm executes steps 1 and 2 repeatedly.

This is summarized in row *a* of table (10); rows *b–d* summarize the rest of this section.

(10) Iterative model-based learning algorithms, from probabilistic data modeling to OT

Iterative model-based solution to the Problem of Learning Hidden Structure …		**Execute Repeatedly:**	
		Step 1	**Step 2**
		Find the hidden structure …	**Find …**
a.	… under general probabilistic data modeling	… that is most probable when paired with the overt data, given the current model of overt/hidden relations.	… the model that makes Step 1's pairing of overt and hidden structure most probable.
b.	… under Harmony Theory/Harmonic Grammar	… that is most Harmonic (probable) when paired with the overt data, given the current grammar (model of overt/hidden relations).	… the grammar that makes Step 1's pairing of overt and hidden structure most Harmonic (probable).
c.	… under OT: **RIP/CD**	… consistent with the overt learning data that has maximal Harmony, given the current grammar. **Robust Interpretive Parsing**	… a grammar that makes Step 1's pairing of overt and hidden structure optimal. **Grammar Learning**
d.	Correctness Criterion	Correctly compute this hidden structure, given the correct model/grammar.	Correctly compute this model/grammar, given the correct hidden structure.

In various formalizations, it has been proven that iterative model-based algorithms converge to a model which is in some sense optimal; in practice, convergence often occurs after only a few iterations, even with a very poor initial model. The key to success is combining correct solutions to the two subproblems addressed in Steps 1 and 2. Crucially, 'correct' here means: finding the correct solution to one subproblem, assuming the other subproblem has been correctly solved. Specifically,

(11) Correctness criteria for solutions of iterative model-based subproblems

Step 1. Given the correct model of overt/hidden relations, correctly compute the hidden structure that is most probable when paired with the overt data.

Step 2. Given the correct hidden structure, correctly compute the model that makes the given pairing of overt and hidden structure most probable.

This is summarized in row *d* of (10).

The iterative model-based approach to learning can be connected directly with OT with the mediation of a piece of neural network theory called Harmony Theory (Smolensky 1983, 1986). In Harmony Theory, the well-formedness of a representation in a neural network is numerically measured by its *Harmony* value *H*, and the probability of a representation is governed by its Harmony: the greater the Harmony, the higher the probability ($prob \propto e^{H}$). A representation has a hidden part and an overt part, and the Harmony function provides the model which relates these two parts: given some overt structure, associating it with different hidden structures leads to different Harmony values (and hence different probabilities). In Step 1 of the iterative learning algorithm (9), given some overt learning data we find that hidden structure which makes the overt data most probable: this means finding the hidden structure which maximizes Harmony, when associated with the given overt structure. In Step 2, we use this hidden structure to change the model: change the Harmony function so that the just-derived hidden/overt associations have the greatest possible Harmony. In Harmonic Grammar (Legendre, Miyata, and Smolensky 1990ab), an application of Harmony Theory to linguistics, the overt and hidden structures are part of linguistic structural descriptions, and the model that governs the relation between overt and hidden structure is a grammar. In this context, the iterative algorithm (10a) becomes (10b), and the correctness criteria (11) for the corresponding subproblems becomes (10d), with 'grammar' in place of 'model.'

In Optimality Theory, the Harmony of structural descriptions is computed from the grammar non-numerically, and there is (as yet) no probabilistic interpretation of Harmony. But the learning procedure of the preceding paragraph still makes perfect sense; it is summarized in (10c), and labelled *RIP/CD*, for 'Robust Interpretive Parsing/Constraint Demotion': Constraint Demotion will be used to perform the Grammar Learning of Step 2.

Given some overt learning data, RIP/CD first computes the hidden structure which has maximal Harmony when combined with the overt structure. Given learning data consisting of a sequence of syllables with stresses, for example, we find the foot structure which, in combination with the given stress pattern, has maximal Harmony. Which foot structure this is depends jointly on the overt stresses and on the currently assumed grammar—the current ranking of metrical constraints. So the algorithm proceeds as follows. Start with an initial grammar (we take up this issue in Section 5). In Step 1 (RIP Step), use this grammar to assign (initially incorrect) hidden structure to the overt learning data by maximizing Harmony. In Step 2, (CD Step) use this hidden structure to learn a new grammar, one in which each combined hidden/overt structure of the currently analyzed data has higher Harmony than all its competitors. With this improved grammar, return to Step 1 and repeat.

Whether this iterative algorithm can be proved to converge, whether it converges in a reasonable time, whether it can be converted into an algorithm that operates effectively on one datum at a time—these and other issues are all open research problems at the moment. But promising preliminary experimental results (Tesar, in preparation b), and previous general experience with iterative model-based algorithms suggests that there are reasonable prospects for good performance, provided we can devise efficient and correct algorithms for solving the two subproblems of our decomposition (10): robust interpretive parsing and grammar learning. The criteria of correctness here (10d) are:

(12) Correctness criteria for solutions to subproblems under OT

Robust interpretive parsing: Given the correct grammar, compute the hidden structure that is most harmonic when paired with the overt data.

Grammar learning: Given the correct hidden structure, find the grammar that makes the pairing of overt and hidden structure optimal.

With respect to the first subproblem, parsing, it is essential that we have a parser that can use a grammar to assign hidden structure to overt forms that are not grammatical according to that very grammar: this is what we mean by 'robustness.' For our problem decomposition, we can now see, imposes a seemingly paradoxical requirement. An overt form will be informative (allow the learner to improve the grammar) if the current grammar (incorrectly) declares it to be ungrammatical. Step 1 of the RIP/CD algorithm requires that we use our current (incorrect) grammar to parse this input (assign it hidden structure), even though the grammar declares it ill-formed. For many formal grammars, such an ungrammatical form is, by definition, unparsable; yet Step 1 requires the grammar to parse it just the same.

OT grammars can easily cope with this demand. An OT grammar provides a Harmony ordering of *all* full structural descriptions (5). This Harmonic Ordering can be used in a variety of ways. The customary use is as follows: given an input *I*, *Gen*(*I*) is the set of all structural descriptions of *I*; we find the maximal-Harmony member of this set, and it is the output assigned to *I*. This use of the grammar corresponds to the 'language generation' problem of computational linguistics, or the 'language production' problem of psycholinguistics. But as proposed in Smolensky (1996b) Harmonic Ordering can be used for the reverse 'language interpretation' or 'language comprehension' problem as well. Now we are given an overt 'phonetic' form φ. The set *Int*(φ) is the class of all structural descriptions with overt part equal to φ. Let us call the maximal-Harmony member of this set the *interpretative parse* assigned to φ by the grammar. Crucially for present purposes, this interpretation process makes sense even when the grammar declares φ ungrammatical (i.e., even when there is no input *I* for which the optimal member of *Gen*(*I*) has overt form φ). An algorithm that can compute this mapping from φ to its interpretative parse is thus a robust interpretive parser capable of performing Step 1 of the RIP/CD algorithm.

We are proposing here to separate the parsing problem and the grammar learning problem; the latter is the topic of this paper, and we restrict our remarks concerning the former to a few points.

Parsing algorithms that exploit the optimization character of OT grammars have been developed in Tesar 1994, 1995ab. Under general formal assumptions on *Gen* and *Con*, these algorithms are proved to be correct and efficient. These algorithms address the parsing problem in the 'language production' direction, rather than the 'language interpretation' direction required here: given an input to the grammar, they compute its optimal structural description. We will call this *production-directed parsing* to contrast it with the interpretive parsing used in RIP/CD.

There are a number of problems which must be solved in constructing parsing algorithms which meet the correctness criterion of robust interpretive parsing in (12), but initial results are positive (Tesar, in preparation a). All evidence to date indicates that the optimization character of OT does not increase the computational cost of parsing. Developers of grammatical frameworks typically take it for granted than the computational challenges of parsing can be dealt with one way or another, and there is no basis at this point for parsing in OT to raise any greater concerns. We thus assume that algorithms for computing optimal interpretative parses from overt inputs exist; obviously, these are needed eventually for a comprehensive OT theory of language processing. The optimization character of OT means that a solution to the intepretive parsing problem is virtually guaranteed to provide a *robust* parser: the optimization required to parse an ungrammatical structure is the same as that required to parse a grammatical structure. The point is that such a robust interpretive parser, already required independently of learning considerations, can be directly recruited via the RIP/OT algorithm into the service of grammar learning.

In the context of theoretical as opposed to computational linguistics, formal, general, efficient algorithms for parsing are neither commonly exhibited nor commonly needed. As we will see in our concrete example of RIP/CD in Section 8, just as OT in theoretical linguistics has to date been entirely workable without explicit production-directed parsing algorithms, RIP/CD is equally workable without explicit interpretive parsing algorithms. The hand-calculational tool of constraint tableaux will allow us to find the interpretive parse of an overt structure just as they have always allowed OT practitioners to find the production-directed parse of an underlying form.

We now put aside parsing until Section 8 and focus on the grammar learning problem. To meet the correctness criterion for this problem (12), we need to solve the following formal problem:

(13) Our learning problem

Given:	The universal components of any OT grammar
	Learning data in the form of [correct] full structural descriptions of grammatical forms
Find:	A language-particular OT grammar consistent with all the given data

In Section 4 we present a family of solutions to this problem. We briefly return in Section 8 to illustrate how our solutions to the grammar learning problem can be combined with robust interpretive parsing to yield the larger RIP/CD learning algorithm which operates only on overt learning data. (Formal study of this overall algorithm is, as we've said, the subject of future research.) We also briefly discuss the problem in phonology acquisition of learning underlying forms (Section 9); this is a prerequisite to interpretive parsing, at least in phonology. In Section 7 we point out some connections between our learnability results and OT research in child language, in the context of some formal questions concerning the initial state.

We begin in Section 3 with a quick review of OT so that we may precisely specify what we mean in the learning problem (13) by 'the universal components of any OT grammar' and 'a language-particular OT grammar.'

But before beginning we should clear up one potential confusion regarding the problem (13) which is the focus of this paper. On first glance, this problem may seem trivial, since knowing the full structural descriptions provides considerable information about the grammar which is not evident in the overt data. What this first glance fails to perceive is that in OT, the grammatical principles (constraints) interact in a rich, complex way. There is nothing like a transparent mapping from the hidden structure to the grammar: the explanatory power of OT lies precisely in the diversity of structural consequences of a constraint embedded within a hierarchy. Knowing the location of the metrical feet in a word, for example, leaves one far short of knowing the metrical grammar. For an OT grammar is a collection of *violable* constraints, and any given foot structure will typically involve the violation of many different constraints: many OT language-particular grammars will be consistent with the given foot structure. A learning algorithm faces a serious challenge in deducing the grammar that explains not just the given form, but also the forms yet to be seen. (Linguists who have actually faced the problem of deducing OT grammars from a complete set of full structural descriptions can attest to the non-triviality of solving this problem, especially in the general case. Indeed, the class of learning algorithms we will present have significant practical value for linguists working in Optimality Theory. Given hypothesized structural descriptions of language data and a hypothesized set of constraints, they can quickly and easily provide a class of constraint rankings which account for the data, or directly determine that no such rankings exist: see Section 11.6.)

The difficulty of the problem of deducing grammars from full structural descriptions is related to the explanatory power of OT as a linguistic theory. We return to this important issue, and some comparisons to Principles-and-Parameters learnability work, in Section 10.1, after we have presented the relevant OT principles and learnability results.

To summarize, in this section we have argued that the following proposal is well-motivated on general learning-theoretic grounds:

(14) Problem Decomposition
Two interrelated subproblems may be distinguished:
Robust interpretive parsing
Grammar learning
Given a solution to each subproblem, correct if the other problem is correctly solved, these can be iteratively applied to yield the RIP/CD algorithm, which learns grammars from overt data.

Having developed a problem decomposition in which the grammar learning subproblem is formulated as in (13), we now proceed to flesh out the problem with the particular grammatical structure provided by OT.

3. The Grammar Learning Problem in Optimality Theory

Optimality Theory defines grammaticality by optimization over violable constraints. The defining reference is Prince and Smolensky 1993 (abbreviated 'P&S' here). Sections 3.1 and 3.2 provide the necessary OT background, while Section 3.3 formulates the precise OT grammar learning problem addressed in the subsequent sections. Readers familiar with OT may wish to move directly to Section 3.3.

We present the basics of OT as a series of general principles, each exemplified within the Basic CV Syllable Theory of P&S (Chapter 6); this syllable theory is also used in Section 4 to illustrate the learning procedure, and in Figure 2 to schematically depict the core structure of OT.

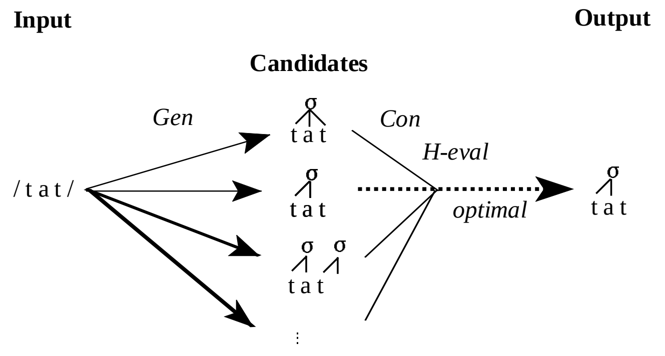

Figure 2: Optimization in OT

3.1 Constraints and Their Violation

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

In 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:

(16)	*a*.	.V.CVC.	$= [_{\sigma}\ \text{V}]\ [_{\sigma}\ \text{CVC}]$
	b.	⟨V⟩.CV.⟨C⟩	$= \text{V}\ [_{\sigma}\ \text{CV}]\ \text{C}$
	c.	⟨V⟩.CV.C□́.	$= \text{V}\ [_{\sigma}\ \text{CV}]\ [_{\sigma}\ \text{C}\acute{\square}]$
	d.	.□V.CV.⟨C⟩	$= [_{\sigma}\ \square\text{V}]\ [_{\sigma}\ \text{CV}]\ \text{C}$

(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 ⟨⟩. 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 □́, 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 □; phonetically, an epenthetic consonant), while the final C is unparsed.

(17) *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 (16) 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.

(18) *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 $\mathbb{C}$ is denoted *$\mathbb{C}$. A parse *a* is more marked than a parse *b* with respect to $\mathbb{C}$ iff $\mathbb{C}$ 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 (19).

(19) Basic CV Syllable Theory Constraints[5]

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 (20).

(20) Constraint Tableau for L_1

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

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 FILL[6]. By contrast, *b* = ⟨V⟩.CV.⟨C⟩ violates PARSE, and more than once.[7] This tableau will be further explained below.

[5] 'ONSET' and 'NOCODA' are the more readable names from McCarthy and Prince 1993 for the constraints P&S call 'ONS' and '–COD'.

[6] These constraints 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.

[7] The only use made in OT of the number of violations of a constraint is comparison of more or less. ⟨V⟩.CV.⟨C⟩ is a worse violation of PARSE than is .V.CV.⟨C⟩: that is all that matters. It is not incorrect to say that the former structure violates PARSE twice and the second structure only once; but it is crucial to keep in mind that the only operation the theory recognizes on the numbers of violations, two *vs.* one, is a comparison of more or less. We will see that this is rooted in the fact that constraints in OT interact via *strict domination*, rather than, e.g., numerically. In another theory, it might be that

3.2 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.

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

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

$$\mathbb{C}_1 \gg \mathbb{C}_2 \gg \cdots \gg \mathbb{C}_n$$

(22) *Harmonic Ordering (H-eval)*: 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 $\mathbb{C}$ with respect to which *a* and *b* are not equally marked: the candidate which is less marked with respect to $\mathbb{C}$ 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-formed[8].

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: it cancels violation marks in common to the two lists. Thus, if a constraint $\mathbb{C}$ assesses one or more marks *$\mathbb{C}$ 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

three violations of FILL equal two violations of PARSE; in such a theory, counting would be essential, in that the numerical values of constraint violations would figure in arithmetic operations (like *number-of-PARSE-violations*/2 + *number-of-FILL-violations*/3) that compute constraint interactions. In OT, however, the violations of each constraint are reckoned separately; all that matters is which of two structures violates a constraint to a greater degree. (These remarks also apply to stratified constraint hierarchies, to be introduced below; all the constraints in a single stratum must simply be aggregated, and treated as though they formed a single constraint.)

[8] 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.

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 $a \prec b$: this is the definition of harmonic ordering $\prec$ in terms of mark cancellation. Mark cancellation is indicated with diagonal shading in the tableau (23): one mark *PARSE cancels between the first two candidates of (20), *d* and *b*, and one uncancelled mark *PARSE remains in *marks'*(*b*).

(23) Mark Cancellation

Candidates		ONSET	NOCODA	FILL$^{\text{Nuc}}$	PARSE	FILL$^{\text{Ons}}$
d.	.□V.CV.⟨C⟩				▨*	*
b.	⟨V⟩.CV.⟨C⟩				▨* *	

Defining grammaticality via harmonic ordering has an important consequence:

(24) *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 (20) under the assumption that the universal constraints are ranked by a particular grammar, L_1, with the ranking given in (25).

(25) Constraint hierachy for L_1: ONSET $\gg$ NOCODA $\gg$ FILL$^{\text{Nuc}}$ $\gg$ PARSE $\gg$ FILL$^{\text{Ons}}$

The constraints (and their columns) are ordered in (20) left-to-right, reflecting the hierarchy in (25). The candidates in this tableau have been listed in harmonic order, from highest to lowest Harmony; the optimal candidate is marked manually[9]. 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 uncancelled 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$^{\text{Nuc}}$. When comparing *b* to *d*, one *PARSE mark cancels, leaving *marks'*(*b*) = {*PARSE} and *marks'*(*d*) = {*FILL$^{\text{Ons}}$}. The worst mark is the uncancelled *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

[9] Determining that this candidate is optimal requires demonstrating that it is more harmonic than *any* of the infinitely many competing candidates. A general technique for such demonstration, the Method of Mark Eliminability (P&S Section 7.3), proceeds by showing that any attempt to avoid the marks incurred by the putatively optimal output leads to alternatives which incur worse marks. Here, avoiding the mark *PARSE of *d* entails violating higher-ranked NOCODA (as in *a*), and avoiding the mark *FILL$^{\text{Ons}}$ of *d* entails violating either ONSET (as in *a*) or PARSE (as in *b*); both these constraints are higher-ranked than FILL$^{\text{Ons}}$. This proves that *d* is more harmonic than all the infinitely many other candidates in *Gen*(/VCVC/).

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 $\Sigma^{\mathrm{CV}}_{\mathrm{ep,del}}$ in P&S:§6.2.2.2).

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

(26) Constraint Hierachy for L_2: ONSET ≫ NOCODA ≫ FILL$^{\mathrm{Ons}}$ ≫ PARSE ≫ FILL$^{\mathrm{Nuc}}$

Now the tableau corresponding to (20) becomes (27); the columns have been re-ordered to reflect the constraint reranking, and the candidates have been re-ordered to reflect the new harmonic ordering.

(27) Constraint Tableau for L_2

Candidates			ONSET	NOCODA	FILL$^{\mathrm{Ons}}$	PARSE	FILL$^{\mathrm{Nuc}}$
/VCVC/ →							
☞	*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 $\Sigma^{\mathrm{CV}}_{\mathrm{del,ep}}$).

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

(28) Typology by Reranking

Systematic cross-linguistic variation is due entirely to variation in language-specific total rankings of the universal constraints in *Con*. Analysis of the optimal forms arising from all possible total rankings of *Con* gives the typology of possible human languages. 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 Chapter 6. In this typology, licit syllables may have required or optional onsets, and, independently, forbidden or optional codas.

Two further principles of OT will figure in our analysis of learnability; we postpone discussion of them until their point of relevance: richness of the base (Section 7.2) and lexicon optimization (Section 9).

As our last issue concerning OT fundamentals, we return to the question of infinity. In the Basic CV Syllable Theory, and quite typically in OT phonology, at least, *Gen*(*I*) contains an infinite number of candidate structural descriptions of each input *I*. In the face of this infinity, is the theory well-defined? Of course, the overwhelming majority of formal systems in mathematics involve an infinity of structures; echoing a theme of Section 1, the mere fact of infinity means only that the most primitive conceivable method, listing all the possibilities and checking each one, is infeasible. But even in finite cases, this method is commonly infeasible anyway. In order for an OT grammar to be well-defined, it must be that for any input, it is formally determinate which structure is optimal. The necessary formal definitions are provided in P&S Chapter 5. In order to show that a given stucture is the optimal parse of *I*, we need to provide a proof that none of the (infinitely many) other parses in *Gen*(*I*) has higher Harmony. One method, developed in P&S:115, is the Method of Mark Eliminability. This proceeds by showing that any attempt to avoid the marks incurred by the putatively optimal output leads to alternatives which incur worse marks; it was illustrated above in footnote 9.

Thus the infinite candidate set has a perfectly well-defined optimum (or optima, if multiple outputs incur exactly the same, optimal, set of marks). Yet it might still be the case that the task of actually *computing* the optimal candidate cannot be performed efficiently. But as Tesar 1995b has shown, computational feasibility is not a problem either, at least in the general cases which have been studied to date. One reason is that the infinity of candidates derives from the unbounded potential for empty structure; but empty structure is always penalized by constraints of the FILL family. Optimal structures may have empty structure, in violation of FILL, only when that is necessary to avoid violation of higher-ranking constraints. This will *not* be the case for unbounded quantities of empty structure.[10] It follows that finite inputs will only have a finite number of structural descriptions which are potentially optimal, under some constraint ranking. Thus a parser constructing an optimal parse of a given input *I* need only have access to a finite part of the infinite space *Gen*(*I*).

The parsing algorithms developed in Tesar 1994, 1995ab construct optimal parses from increasingly large portions of the input, requiring an amount of computational time and storage space that grows with the size of the input only as fast as for parsers of conventional, rewrite-rule grammars of corresponding complexity. The structure in the space of candidates allows for efficient computation of optimal parses, even though the grammar's specification of well-formedness makes reference to an infinite set of parses.

3.3 The Grammar Learning Problem

Having provided the necessary principles, we can now insert the crucial grammatical structure of Optimality Theory into the learning problem (13) that was schematically formulated in Section 2.2:

(29) Our grammar learning problem (including relevant grammatical structure)
Given: The universal components of any OT grammar:
the set of possible inputs
the function *Gen* generating the candidate outputs for any possible input
the constraints *Con* on well-formedness
Learning data in the form of full structural descriptions of grammatical forms

[10] For instance, take the Basic CV Syllable Structure constraints. These may force violation of FILL at a syllable position, but only in a very limited set of environments, deduced in P&S Section 6.2.3 under the name 'epenthesis sites.' In particular, under the syllable structure constraints alone, an entirely empty syllable can never be optimal, since the FILL violations thereby incurred do not avoid any violation of syllable-internal structural constraints. This excludes all but a finite number of parses of any finite-length input *I* from the set of possibly optimal parses—under any ranking—in *Gen*(*I*). More generally, beyond CV syllable theory, empty structure of whatever sort can be optimal, despite violation of the relevant FILL constraints, only when that empty structure is needed for the well-formedness of some larger constituent (segment, syllable, foot, clause, etc.), the larger constituent being needed to parse some part of the input.

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

The initial data for the learning problem are well-formed outputs; each consists of an input together with the structural description which is declared optimal by the target grammar. For example, the learner of the CV language L_1 might have as an initial datum the structure p = .□V.CV.⟨C⟩ (20*d*), the parse assigned to the input I = /VCVC/.

4. Constraint Demotion

Having isolated the grammar learning problem from the parsing problem in Section 2, and provided the grammar learning problem with the UG structure characteristic of OT in Section 3, we are now in a position to present our proposal, Constraint Demotion, for addressing the OT grammar learning problem (29). In Sections 5 and 6 we show that Constraint Demotion does indeed solve the problem.

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

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

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

In our CV example, a learner given the positive datum p = .□V.CV.⟨C⟩ knows that, with respect to the unknown constraint hierarchy of the language being learned, the alternative parse of the same input, p' = .V.CVC., is less harmonic:

.V.CVC. $\prec$.□V.CV.⟨C⟩

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

Thus each single piece of positive initial data conveys a large amount of inferred comparative data of the form:

[sub-optimal parse of input *I*, ‘*loser* ’] ≺ [optimal parse of input *I*, ‘*winner* ’]

Such pairs are what feed our learning algorithm. Each pair carries the information that the constraints violated by the sub-optimal parse *loser* must out-rank those violated by the optimal parse *winner*—that, in some sense, we must have *marks*(*loser*) ≫ *marks*(*winner*).

(30) The key
loser-marks ≫ *winner-marks*

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

4.1 The Basic Idea

In our CV language L_1, the winner for input /VCVC/ is .□V.CV.⟨C⟩. Table (20) 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 (31). A *mark-data pair* is the paired lists of constraint violation marks for a *loser/winner* pair.

(31) Mark-data pairs (L_1)

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

To make contact with more familiar OT constraint tableaux, the information in (31) will also be displayed in the format of (32).

(32) Initial data

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

At this point, the constraints are unranked; the dotted vertical lines separating constraints in (32) conveys that no relative ranking of adjacent constraints is intended. The winner is indicated with a ✓; ☞ will denote the structure that is optimal according to the current grammar, which may not be the same as the winner (the structure that is grammatical in the target language). The constraint violations of the winner, *marks*(*winner*), are distinguished by the symbol ⊛. Diagonal shading denotes mark cancellation, as in tableau (23).

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 (22), what this means more precisely is that *loser* must incur the worst uncancelled mark, compared to *winner*. This requires that uncancelled marks be identified, so the first step is to cancel the common marks in (31).

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

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

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 $b \prec d$ and $c \prec d$, but not in the pair $a \prec d$. Note also that cancellation of marks is done token-by-token: in the row $b \prec d$, one but not the other mark *PARSE in *marks*(b) is cancelled.

The table (33) of mark-data after cancellation is the data on which Constraint Demotion operates. Another representation in tableau form is given in (32), 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 uncancelled winner marks ⊛ are dominated by at least one loser mark *. Using the standard tableau convention of positioning the highest-ranked constraints to the left, the columns containing uncancelled ⊛ 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 uncancelled * (loser mark) which is further to the left (dominant in the hierarchy) than all of the columns containing uncancelled ⊛ (winner marks).

The algorithm to accomplish this is based upon the principle in (34).

(34) The Principle of Constraint Demotion: for any constraint $\mathbb{C}$ assessing an uncancelled winner mark, if $\mathbb{C}$ is not dominated by a constraint assessing an uncancelled 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 $a \prec d$ above is such a case). This is the challenge successfully overcome by Constraint Demotion.

4.2 Stratified Domination Hierarchies

Optimality-Theoretic 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:

(35) Stratified Domination Hierarchy

$$\{\mathbb{C}_1, \mathbb{C}_2, ..., \mathbb{C}_3\} \gg \{\mathbb{C}_4, \mathbb{C}_5, ..., \mathbb{C}_6\} \gg ... \gg \{\mathbb{C}_7, \mathbb{C}_8, ..., \mathbb{C}_9\}$$

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

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

The definition of harmonic ordering (22) needs to be elaborated slightly for stratified hierarchies. When $\mathbb{C}_1$ and $\mathbb{C}_2$ are in the same stratum, two marks $*\mathbb{C}_1$ and $*\mathbb{C}_2$ are equally weighted in the computation of Harmony. In effect, all constraints in a single stratum are collapsed together, and treated as though they were a single constraint, for the purposes of determining the relative Harmony of candidates. Minimal violation with respect to a stratum is determined by the candidate incurring the smallest sum of violations assessed by all constraints in the stratum. The tableau in (36) gives a simple illustration.

(36) Harmonic ordering with a stratified hierarchy: $\mathbb{C}_1 \gg \{\mathbb{C}_2, \mathbb{C}_3\} \gg \mathbb{C}_4$

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

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; this is valid independent of the target constraint hierarchy, which, during learning, is unknown.

4.3 An Example: Basic CV Syllable Theory

Constraint Demotion (abbreviated CD) will now be illustrated using CVT; specifically, with the target language L_1 of (20, 25). The initial stratified hierarchy is set to

(37) $\mathcal{H} = \mathcal{H}_0 = \{$Fill$^{\text{Nuc}}$, Fill$^{\text{Ons}}$, Parse, Onset, NoCoda$\}$

Suppose that the first loser/winner pair is *b* ≺ *d* of (31). Mark Cancellation is applied to the corresponding pair of mark lists, resulting in the mark-data pair shown in (38).

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

	loser ≺ *winner*	*marks′*(*loser*)	*marks′*(*winner*)
b ≺ *d*	⟨V⟩.CV.⟨C⟩ ≺ .□V.CV.⟨C⟩	~~*Parse~~ *Parse	~~*Parse~~ *Fill$^{\text{Ons}}$

Now CD can be applied. The highest-ranked (in $\mathcal{H}$) uncancelled 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$^{\text{Ons}}$, which is *not* so dominated. CD therefore calls for demoting Fill$^{\text{Ons}}$ to the stratum immediately below Parse. Since no such stratum currently exists, it is created. The resulting hierarchy is (39).

(39) $\mathcal{H} = \{$Fill$^{\text{Nuc}}$, Parse, Onset, NoCoda$\} \gg \{$Fill$^{\text{Ons}}\}$

This demotion is shown in tableau form in (40); 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 uncancelled winner mark ⊛ is demoted to a (new) stratum immediately below the stratum containing the highest uncancelled winner mark *, which now becomes a fatal violation *! rendering irrelevant the dominated violation ⊛ (which is therefore greyed out).

(40) First Demotion

loser/winner pair			Fill$^{\text{Nuc}}$	Fill$^{\text{Ons}}$	Parse	Onset	NoCoda	Fill$^{\text{Ons}}$
d	☞✓	.□V.CV.⟨C⟩		⊛	⊛			⊛
b		⟨V⟩.CV.⟨C⟩			* *!			

Now another loser/winner pair is selected. Suppose this is *a* ≺ *d* of (31):

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

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

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.

(42) $\mathcal{H}$ = {FILL^{Nuc}, ONSET, NOCODA} ≫ { FILL^{Ons}, PARSE}

In tableau form, this demotion is shown in (43). (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.)

(43) Second Demotion

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

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

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

	loser ≺ *winner*	*marks*′(*loser*)	*marks*′(*winner*)
c ≺ *d*	⟨V⟩.CV.C□́. ≺ .□V.CV.⟨C⟩	~~*PARSE~~ *FILL^{Nuc}	~~*PARSE~~ *FILL^{Ons}

Since the uncancelled loser mark, *FILL^{Nuc} already dominates the uncancelled 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, .□V.⟨C⟩.

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

loser ≺ *winner*	*marks*′(*winner*)	*marks*′(*winner*)
⟨VC⟩ ≺ .□V.⟨C⟩	~~*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 (46).

(46) $\mathcal{H}$ = {FILL^{Nuc}, ONSET, NOCODA} ≫ {PARSE} ≫ {FILL^{Ons}}

This demotion is shown in tableau (47).

(47) Third Demotion

loser/winner pair	FILL$^{\text{Nuc}}$	ONSET	NOCODA	FILL$^{\text{Ons}}$	PARSE	FILL$^{\text{Ons}}$
☞✓ .□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 uncancelled mark assessed by a constraint dominating all the constraints assessing uncancelled marks to *winner*. Thus, no further data will be informative: L_1 has been learned.

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

(48) The Disjunction Problem

loser/winner pair		FILL$^{\text{Ons}}$	ONSET	FILL$^{\text{Nuc}}$	NOCODA	PARSE
d ✓	.□V.CV.⟨C⟩	⊛				⊛
a	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 (49).

(49) (ONSET **or** NOCODA) ≫ (FILL$^{\text{Ons}}$ **and** PARSE)

Demotion moves the constraints corresponding to the winner's marks. They are contained in a conjunction (**and**); thus, once the highest-ranked loser mark is identified, *all* 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 (48), that is ONSET). Thus, if other data have already determined that ONSET $\gg$ NOCODA, that relationship is preserved. The constraints violated by the winner are only demoted as far as necessary.

5. Analysis of Constraint Demotion

Having defined Constraint Demotion, we now turn to its analysis. Technical definitions and proofs have been relegated to the Appendix, Section 11.

5.1 Learnability Results: Convergence and Efficiency

The illustration of Constraint Demotion given in Section 4.3 started with initial hierarchy $\mathcal{H}_0$, given in (37), having all the constraints in one stratum. Using this initial hierarchy is convenient for demonstrating some formal properties. By starting with all constraints at the top, CD can be understood to demote constraints down toward their correct position. Because CD only demotes constraints as far as necessary, a constraint never gets demoted below its target position, and will not be demoted further once reaching its target position. The formal analysis in Sections 11.1 to 11.3 assumes $\mathcal{H}_0$ as the initial hierarchy, and proves the following results, as (83, 92):

(50) 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.

(50) Theorem: Data complexity of Constraint Demotion

The number of informative winner/loser pairs required for learning is at most $N(N–1)/2$, where N = number of constraints in *Con*.

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

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

The significance of this result is perhaps best illustrated by comparing it to the number of possible grammars. Given that any target grammar is consistent with at least one total ranking of the constraints, the number of possible grammars is the number of possible total rankings, $N!$. This number grows very quickly as a function of the number of constraints N, and if the amount of data required for learning scaled with the number of possible total rankings, it would be cause for concern indeed.

Fortunately, the data complexity of CD is quite reasonable in its scaling. In fact, it does not take many universal constraints to give a drastic difference between the data complexity of CD and the number of total rankings: when N=10, the CD data complexity is 45, while the number of total rankings is over 3.6 million. With 20 constraints, the CD data complexity is 190, while the number of total rankings is over 2 billion billion (2.43×10^{18}). This reveals the restrictiveness of the structure imposed by 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.

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

This dramatic difference between the size of the OT grammar space and the number of informative examples needed to learn a grammar is due to the well-structured character of the space of fully-ranked constraint hierarchies. It is useful to consider a set of parameters in the grammar space that suffice to specify the $N!$ grammars: these parameters state, for each pair of different constraints $\mathbb{C}_i$ and $\mathbb{C}_j$, which is dominant, i.e., whether $\mathbb{C}_i \gg \mathbb{C}_j$ or $\mathbb{C}_j \gg \mathbb{C}_i$. There are in fact $N(N\text{–}1)/2$ such dominance parameters[11], and this is the maximum number of informative examples needed to learn a correct hierarchy.[12] Efficient learning via Constraint Demotion is possible because the enlarged hypothesis space allows these dominance parameters to be unspecified (in the initial state, they are *all* unspecified), and because evidence for adjusting these dominance parameters can be assessed independently (via the

[11] There are N choices for i, and for each one, $N-1$ choices for a different j. This gives $N \times (N-1)$ *ordered* pairs. This counts each pair of constraints twice, as (j,i) and (i,j), so dividing by 2 we get the number of *unordered* pairs, $N(N-1)/2$.

[12] There are subtleties here which require considerable caution. There is not a one-to-one correspondence between the demotions arising from informative examples and the setting of the ranking parameters. For example, a constraint $\mathbb{C}_i$ may be demoted below another $\mathbb{C}_j$ at one point during learning, and then $\mathbb{C}_j$ may be demoted below $\mathbb{C}_i$ later. Thus the *ij* ranking parameter would be set one way first, then the other way later. Given this, it is unclear why the number of informative examples needed cannot be greater than the number of parameters, or, indeed, why constraint demotion is ever guaranteed to converge to a solution. The ranking parameters may not to be an effective formal tool in analyzing the algorithm, although they are conceptually helpful in understanding how the ranking structure on the space of grammars can enable efficient search through an enormous space. The technically more powerful analytic tool is the concept of h-domination, developed in the Appendix.

key idea (30): *loser-marks* ≫ *winner-marks*). A single adjustment may not irrevocably set a correct value for any dominance parameter, each adjustment brings the hierarchy closer to the target, and eventually the adjustments are guaranteed to produce a correct set of parameter values. Note that what is independently adjustable here is not the substantive *content* of individual grammatical principles: it is the *interaction* of the principles, as determined by their relative rankings. This is an important point to which we will return in Section 10.1.

5.2 Learnability and Total Ranking

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

The discussion in this paper assumes that the learning data are generated by a UG-allowed grammar, which, by (28), 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 (28) is the broad finding that correct typologies of adult languages do not seem to result when constraints are permitted to form stratified hierarchies. Generally speaking, allowing constraints to have equal ranking produces empirically problematic constraint interactions.

From the learnability perspective, the formal results given for Constraint Demotion depend critically on the assumption that the target language is given by a totally-ranked hierarchy. This is a consequence of a principle implicit in CD. 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). CD makes vigorous use of this learning principle.

When presented data from a non-totally-ranked stratified hierarchy, it is in fact possible for CD to run endlessly. 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, CD will assume the competitor p' to be suboptimal, since its marks are not identical to those of p. CD 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, CD 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 CD 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. Following the tenets of OT, we take the target grammars to be totally-ranked hierarchies. Constraint Demotion, however, does not operate within the confines of the space of totally-ranked hierarchies, however. All evidence to date indicates that, consistent with a general theme of recent work in Computational Learning Theory (e.g., Pitt and Valiant 1988, Kearns and Vazirani 1994; for a tutorial, see Haussler 1996), feasible learning of the target space requires a learning algorithm to search within a larger space: the space of stratified hierarchies.

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 assumption has the potential disadvantage that the hypotheses all involve full commitment with respect to dimensions of grammatical variation, even though no evidence may yet have been obtained to justify such commitments. The stratified hierarchies constituting the hypothesis space exploited by Constraint Demotion, by contrast, can be widely uncommitted on the relative ranking of constraints, useful when no relevant evidence has yet been observed. This is crucial to the successful operation of CD. On the other hand, the more traditional assumption that the learner's hypotheses are all UG-allowed grammars 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. In our case, since learning takes place in the larger space of stratified hierarchies, we must explicitly address two questions: How does the learner get to a UG-allowed grammar (a totally-ranked hierarchy)? And why must adult grammars always be totally-ranked 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 *any* further refinement into a fully-ranked hierarchy will correctly account for all the learning data. Lacking any evidence on which to do so, the learning algorithm does not commit to any of these. In human terms, however, 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 on which fully-ranked hierarchy is chosen.

Why must adult grammars be totally-ranked hierarchies? The situation we have described is a rather curious one. When learning data from a fully ranked hierarchy is presented to our learning algorithms, they generally terminate in a stratified hierarchy with the property that all of its refinements into totally-ranked hierarchies correctly generate the learning data. But when the learning data derive from a non-totally-ranked stratified hierarchy, the algorithms can fail to terminate at all. Thus the space of fully-ranked hierarchies is learnable by Constraint Demotion followed by refinement to some (any) fully-ranked hierarchy; the larger space of stratified hierarchies is not learnable via CD, as far as we can determine at this point.

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), CD has no problems whatever coping with ties for optimality between outputs with the *same* marks (possible given a totally-ranked hierarchy): given two such outputs as a data pair, all marks cancel, and CD correctly leaves the hierarchy unchanged.

6. Selecting Competing Structural 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.

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 7.2 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: given an input *I* and a stratified hierarchy $\mathcal{H}$, compute the optimal parse(s) of *I*. This is the computational problem solved in a number of general cases by Tesar (1995b), as discussed in Section 2.2.

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).

(52) 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.[13]

7. The Initial State

The results of Sections 5 and 6 demonstrate that our grammar learning problem is efficiently solvable by Error-Driven Constraint Demotion, under the convenient simplifying assumption of an initial stratified hierarchy in which all constraints are top-ranked. In this section, we focus attention on the initial state, and argue that further learnability considerations lead to an initial hierarchy with somewhat more articulated structure.

7.1 Extension of the Learnability Results to Arbitrary Initial Hierarchies

Our first question is whether the learnability results established in previous sections require the initial stratified hierarchy to have all constraints top-ranked.

The role of the initial hierarchy in the learnability proofs is as follows. Constraint demotion will always ensure that the current winner/loser pair is treated correctly by the grammar after demotion. The danger is that demotion will go on forever, failing to converge to a stable hierarchy. In the Appendix, essentially the following method of proof is used to show that indefinite demotion cannot occur. In order for a constraint $\mathbb{C}_n$ to be demoted to the *n*th stratum, there must be another constraint $\mathbb{C}_{n-1}$ in the *n*–1st stratum such that, in the target hierarchy, $\mathbb{C}_{n-1} \gg \mathbb{C}_n$. And similarly, in order for $\mathbb{C}_{n-1}$ to ever have been demoted from the top of the hierarchy down to the *n*–1st stratum, there must have been another constraint $\mathbb{C}_{n-2}$ in the *n*–2nd stratum such that $\mathbb{C}_{n-2} \gg \mathbb{C}_{n-1}$ in the target hierarchy. The target hierarchy only has a finite number of constraints, *N*, so this chain of dominations can never be longer than *N*. Thus, the stratum *n* that the constraint $\mathbb{C}_n$ was demoted to at the beginning of this argument cannot

[13] There is a choice to be made in exactly how to apply EDCD 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 same 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.

possibly be greater than *N*. No constraint can ever undergo more than *N* demotions, so constraint demotion is guaranteed to stop; at this point, the hierarchy correctly treats all the training data (otherwise, more demotions would occur).

Now the role of the initial hierarchy here is very limited. Suppose constraints start off ranked in an arbitrary initial hierarchy, rather than all top-ranked. In the argument of the previous paragraph, we can no longer conclude that if a constraint is in some stratum below the top, it must have gotten there by demotion: for it might well have just *started* there. So the proof needs to be modified somewhat. Suppose the lowest initial statum is numbered *K*. We can still conclude that any constraint in a statum *below K* must have gotten there by demotion, which means it must be dominated in the target hierarchy by another constraint. This leads to the same conclusion as before, that the number of demotions must be finite; indeed, no constraint can be demoted below stratum *N*+*K*, which allows us to compute the maximal number of possible demotions, i.e., the maximal number of informative examples necessary to complete learning. This reasoning yields the following result (the proof is given in the Appendix, Section 11.4).

(53) Theorem: Constraint Demotion with arbitrary initial hierarchy

Starting with an arbitrary initial constraint hierarchy, Constraint Demotion converges to a correct hierarchy after no more than $N(N–1)$ informative examples (where N = number of constraints).

The conclusion, then, is that the learnability results of Section 5 are robust with respect to the initial hierarchy: efficient convergence on a correct hierarchy is guaranteed regardless of the initial hierarchy.[14]

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

(54) Coronal unmarkedness universal subhierarchy

*PL/Lab ≫ *PL/Cor

In syntax, Legendre et al. 1995, Legendre, Smolensky and Wilson 1996 have proposed a universal hierarchy MINLINK which realizes the ‘Shortest Link’ principle. We now see that having such UG rankings in the initial state does not jeopardizing learnability. The Constraint Demotion algorithm is easily adapted so that whenever a constraint that is part of a universal markedness subhierarchy is demoted, the constraints below it in the hierarchy are also demoted if necessary to preserve the universal

[14] With arbitrary initial hierarchies, CD can lead to empty strata; this can be seen as follows. Because the data observed must all be consistent with some total ranking, there is at least one constraint never assessing an uncancelled 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 uncancelled 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.

subhierarchy.

7.2 The Subset Principle, Richness of the Base, and Acquisition Theory in OT

The learnability considerations thus far do not yet tell us anything about the initial state, then. As pointed out to us by Alan Prince (1993), however, some progress can be made, if we turn to a relevant central principle of Optimality Theory not yet considered:

(55) Richness of the base: The set of possible inputs to the grammars of all languages is the same. The grammatical inventories of languages are defined as the forms appearing in the outputs 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 systematic 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*.

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 (20). This can only be achieved by rankings in which faithfulness constraints are dominated by the structural constraints. (25) 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.

But what about cases in which evidence from alternation is absent? Prince suggests that perhaps the *initial hierarchy* 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 aquisition (Pater and Paradis 1996, Bernhardt and Stemberger 1995, 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.

(56) Assumptions of OT acquisition work
(a) the child's input is close to the adult form;
(b) the initial ranking is one in which the faithfulness constraints are dominated by the structural constraints.

We have just sketched an argument that learnability of languages with unmarked inventories requires that (b) hold (for a more complete version of this argument, see Smolensky 1996c). Assumption (a) too can be explained using ideas developed here: it can also be shown (Smolensky 1996b) that robust interpretive parsing explains how the child's lexical entries can be quite 'faithful' to adult forms even when their grammars, characterized by property (b), produce outputs that are massively 'unfaithful' to the adult forms (and to the child inputs). Thus the learnability considerations of this paper have significant implications for the foundations of current OT work in child language.

8. Recomposing the Language Learning Problem

Having provided a solution to our grammar learning subproblem, we now return to the larger problem decomposition in which this subproblem was embedded in Section 2. There we proposed a learning algorithm, RIP/CD (10), which combines a solution to our grammar learning problem with robust interpretive parsing to learn from overt learning data:

(57) RIP/CD: Iterative model-based approach to the Problem of Learning Hidden Structure under OT
Step 1. Find the hidden structure consistent with the overt learning data that has maximal Harmony, given the current grammar. [Robust Interpretive Parsing]
Step 2. Find a grammar that makes this pairing of overt and hidden structure optimal. [Grammar Learning: Constraint Demotion]
Starting with some initial grammar, execute Steps 1 and 2 repeatedly.

In this section we use Constraint Demotion to sketch an illustration of how RIP/CD might look when applied to the domain of learning stress systems. This illustration is tentative in several respects; we offer it to render concrete the general discussion of Section 2.

Our objective in this example is to illustrate as expeditiously as possible how RIP/CD proceeds to learn a grammar from overt data only. To facilitate some more general comparisons, we consider the same grammatical domain as the learnability study of Dresher and Kaye 1990, stress systems. A serious study would start with a characterization of the metrical module of UG already under vigorous development in the OT literature: a set of structures and constraints such that all possible rankings of the constraints yields all and only the observed stress systems. To allow us to proceed directly to the relevant learnability issues, however, we will simply consider a few plausible constraints which allow some major dimensions of metrical variation to arise through reranking.

Two of the major dimensions of variation that a stress learner must cope with, in the fairly standard pre-OT terminology used by Dresher and Kaye 1990, are quantity sensitivity/insensitivity and ±extrametricality. The former is particularly interesting because it can be seen as a rather major distinction in how the input is to be analyzed.

It is convenient for our illustration to adopt the following four constraints:

(58) Illustrative metrical constraints[15]

a.	BISYLL:	A foot is bisyllabic.
b.	WSP:	A heavy syllable is a head of a foot. (Weight-to-Stress Principle)
c.	PARSE-σ:	A syllable is parsed into a foot.
d.	NONFIN:	A foot is not final in the prosodic word.

To focus the analysis, we will assume in this example that the learner has already correctly ranked the constraints governing the trochaic/iambic distinction, the 'directionality' of footing, and whether the word's head foot is on the right or left; we also assume these are all higher-ranked than the constraints in (58) which we examine. Simultaneously ranking all these constraints would constitute not an example but a sizable paper.

Our first dimension of interest, the quantity sensitive/insensitive contrast, is characterized in part by the relative ranking of the two constraints (58a,b). When BISYLL dominates WSP, feet will always be bisyllabic even when that entails heavy syllables falling in the weak position in a foot; footing will be driven by the higher-ranked constraints on foot form and 'directionality' and by BISYLL; WSP will not be active and so the weight of syllables will not affect footing. On the other hand, when WSP ≫ BISYLL, footing is weight-sensitive, with heavy syllables banned from foot-weak position.

Our second dimension, ±extrametricality, is governed by the two constraints (58c,d). In the +extrametrical case, the final syllable is left unfooted; this ensures that no foot is final in the word, in satisfaction of NONFIN; but this entails a violation of PARSE-σ since the final syllable is not parsed into a foot. Thus the +extrametrical case arises when NONFIN dominates PARSE-σ; the reverse ranking yields –extrametricalty.

In our example, we assume the learner is faced with a stress system like that of Latin: stress is penultimate when the penult is heavy, otherwise it is antepenultimate. This is a quantity-sensitive stress system with +extrametricality. We assume that the learner's current ranking is wrong on both dimensions. We assume the feet to be troachic, right-to-left, with main stress on the right; as already remarked, we presume the learner to have correctly learned these dimensions. When we pick up the learner, the ranking of the four constraints in (58) is taken to be:

(59) PARSE-σ ≫ NONFIN ≫ BISYLL ≫ WSP

We suppose the next datum is LĹLL. Note that here we are taking as given to the learner only the *overt* structure: syllable weights (L=light, H=heavy) and stresses, assumed by Dresher and Kaye to be directly available to the learner. Note that this datum is not grammatical according to the current grammar, which declares that LLLL should be parsed as (LL)(ĹL) [quantity-insensitive, –extrametrical], surfacing as LLĹL; this is shown in tableau (60). The optimal candidate according to the current

[15] These constraints are closely related to those of P&S. BISYLL is half of the constraint FTBIN (P&S:47, (61)), "Foot Binarity: Feet are binary at some level of analysis (μ, σ)." The WSP is P&S:53 (67); PARSE-σ is the relevant member of the PARSE family, introduced explicitly in P&S:58. NONFIN is formulated in P&S:52 (66) as "NONFINALITY: No head of PrWd is final in PrWd"; our slightly simpler formulation is possible here because in the relevant candidates the final foot is the head of PrWd, the prosodic word.

grammar, *a*, is, as usual, marked with ☞.[16] The correct parse L(ĹL)L [quantity-sensitive, +extrametrical] is shown with ✓.

(60) Ranking when we pick up the learner

LĹLL: before	PARSE-σ	NONFIN	BISYLL	WSP
a. ☞ (LL)(ĹL)		*		
b. ✓ L(ĹL)L	* *			
c. ➽ (L)(ĹL)L	*		*	

We want to learn from this informative datum LĹLL by Constraint Demotion, but this requires a full structural description for the datum. According to our overall learning algorithm RIP/CD (10), we get a structural description by applying robust interpretive parsing, the first step of the algorithm. Recall from Section 2 that the correct interpretive parse is defined to be the most harmonic candidate among those that have the correct overt part. Candidate *a* has incorrect overt part (penultimate stress), while candidates *b*–*c* have correct overt part. Comparing them, we see that *c* has maximal Harmony: it is the robust interpretive parse, as indicated by ➽, although it is not the correct parse. (Note the need for *robust* parsing here: we need to use the current grammar to parse LĹLL into foot structure, even though this stress pattern is not grammatical according to the current grammar, which declares that input /LLLL/ must be stressed LLĹL.) The RIP/CD algorithm assigns the hidden foot structure of the robust interpretive parse *c* to this datum, and proceeds to the second step, grammar learning from the full structural description given by interpretive parsing.

For this we will use Error-Driven Constraint Demotion. The relevant winner-loser pair is determined as follows. The loser, according to the Error-Driven algorithm, is the optimal candidate from the current grammar: *a*. The winner, from the interpretive parsing step, is *c*. Both the winner marks *PARSE-σ and *BISYLL must be dominated by the loser mark, *NONFIN. This is already true for *BISYLL, but PARSE-σ must be demoted to just below NONFIN, into the same stratum as BISYLL, yielding the ranking shown in (61). (Recall that the dotted line between PARSE-σ and BISYLL indicates they are equally ranked: both part of the same stratum)

[16] We have of course omitted other candidates which must be shown sub-optimal, such as those violating the high-ranked constraints requiring trochaic feet, candidates like (L)(ĹL)(L) which are universally sub-optimal (universally less harmonic than *a*), etc.

(61) After robust interpretive parsing + Error-Driven Constraint Demotion on datum LĹLL

LĹLL: after		PARSE-σ	NONFIN	PARSE-σ	BISYLL	WSP
a.	(LL)(ĹL)		*!			
b. ☞✓	L(ĹL)L	⊛ ⊛		* *		
c. ☞	(L)(ĹL)L	⊛		⊛	⊛	

Now *c* is optimal, as desired. But *b* is optimal as well: their constraint violations differ only in that *c* has an additional violation of PARSE-σ while *d* has an additional violation of BISYLL; since these constraints occupy the same stratum in the hierarchy, these candidates are equally harmonic. After this demotion, the ranking of NONFIN and PARSE-σ has reversed, and the stress system has switched from –extrametrical to +extrametrical. (Lower-ranked PARSE-σ is still active in the grammar, however; for example, it causes *c* to tie for optimality with *b*—incorrectly, as it happens.)

The quantity-insenstitivity of the grammar has not changed; that requires an input with quantity contrasts, such as LH́L. As shown in tableau (62), the current grammar parses the input LHL as (ĹH)L, with incorrect stress placement due to quantity insensitivity:

(62) Initial analysis of second datum

LH́L: before		NONFIN	PARSE-σ	BISYLL	WSP
a. ☞	(ĹH)L				*
b.	L(H́L)	*	*		
c. ➳✓	L(H́)L		* *	*	
d. ➳	(L)(H́)L		*	* *	

To learn from this error, we apply the first step of RIP/CD: robust interpretive parsing. The maximal-Harmony parses with correct overt part (stress) are *c* and *d* (tied); we suppose the incorrect one is chosen, *d*. So the hidden (foot) structure imposed on the overt data is (L)(H́)L. With this full structural description, we can now perform the second step of RIP/CD: Constraint Demotion. The winner/loser pair has *d* as winner and *a* as loser; the two constraints violated by the winner, PARSE-σ and BISYLL must be demoted beneath the one constraint violated by the loser, WSP:

(63) After robust interpretive parsing + Error-Driven Constraint Demotion on datum LH́L

LH́L: after		NONFIN	PARSE-σ	BISYLL	WSP	PARSE-σ	BISYLL
a.	(ĹH)L				*!		
b.	L(H́L)	*!	✻			*	
c. ☞✓	L(H́)L		✻ ✻			* *	*
d. ☞	(L)(H́)L		⊛	⊛ ⊛		⊛	⊛ ⊛

After demotion, two candidates *c* and *d* tie for optimality. We have assumed *c* to be the correct parse; *d* will have to be rendered less harmonic by a later demotion of PARSE-σ, or by constraints on foot form which we have not taken into account, such as the part of Foot Binarity, FTBIN, which requires feet to be binary at the level of the mora, banning (L). But we will not follow this learning process any further. We simply observe that after the second iteration of RIP/CD, the first datum we considered is still assigned correct stress (although the structural ambiguity involving monomoraic feet still obtains):

(64) Resulting analysis of first datum, LĹLL

LĹLL: final		NONFIN	WSP	PARSE-σ	BISYLL
a.	(LL)(ĹL)	*!			
b. ☞✓	L(ĹL)L			* *	
c. ☞	(L)(ĹL)L			*	*

To see how one novel datum is treated by this grammar, we consider ĹLH. As it happens, the grammar correctly assigns antepenultimate stress, propertly handling the interaction between the quantity-sensitivity and extrametricality dimensions in the case of a final heavy syllable:

(65) Resulting analysis of novel datum ĹLH

ĹLH: final		NONFIN	WSP	PARSE-σ	BISYLL
a. ☞✓	(ĹL)H		*	*	
b.	(LL)(H́)	*!			*
c.	L(ĹH)	*!	*	*	
d.	(L)(ĹH)	*!	*		*
e.	L(Ĺ)H		*	* *!	*
f.	(L)(Ĺ)H		*	*	*! *

(As the fatal violations *! in this tableau and tableau (63) show, every constraint in this little grammar is active, i.e., used to eliminate suboptimal candidates.)

In concluding this little illustration, let us be clear about what it is and is not intended to show. As we have said, the formal properties of RIP/CD are the subject of future research; we are not presently in a position to claim for RIP/CD what we can claim for Constraint Demotion: that it efficiently computes a correct solution to its learning problem, namely, learning from overt data alone.[17]

Our example is intended primarily to illustrate concretely how RIP/CD addresses the problem of learning when hidden grammatical structure is not provided in the primary learning data, and how Constraint Demotion can provide the grammar-learning engine at the heart of a larger learning algorithm operating on overt data: the only additional element needed is robust interpretive parsing. Rather than heuristic procedures for finessing absent hidden structure, based on particular properties of a given theory of, say, stress (e.g., the theory-specific *cues* of Dresher and Kaye 1990) RIP/CD's solution to the problem of assigning hidden structure is grammatical-framework general, based on the fundamental principles of optimization that are the defining foundation of the framework.

The example also demonstrates the point that, in linguistically interesting cases, determining the correct robust parse can be no more problematic than determining ordinary optimality: comparison of candidates using tableaux works in much the same way in the two cases. From the perspective of theoretical (as opposed to computational) linguistics, the availability of correct, efficient and highly

[17] In evaluating the overall state of learnability results in OT five years after the framework's creation, a relevant context is the state of learnability results in Principles-and-Parameters theory, more than fifteen years after its creation. Consider the examples discussed in Section 1. The primary claims of Gibson and Wexler 1994 are: (a) the *failure* of learnability in a particular, simple UG with three parameters, and (b) learnability given the existence of triggers, a conclusion challenged by Frank and Kapur 1996. Concerning the success of their learning algorithm specially designed for a particular metrical theory, Dresher and Kaye's assessment is that it "does quite well" (1990:175); "while determining that a system is QS is usually unproblematic, the Learner does less well than a linguist in making the further discrimination between QS [Rime] and QS [Nucleus]" (1990:175–176; "QS" = quantity sensitive); "aside from some problems with extrametricality … the Learner performs quite well within the bounds of its theory" (1990:177). Clearly, deriving linguistically relevant formal learnability results is a rather difficult problem.

general algorithms for computing robust interpretive parses is simply not an issue.

Finally, the example illustrates how the interaction of robust interpretive parsing and Constraint Demotion in RIP/CD leads to the minimal demotion that will fit the overt form. For interpretive parsing chooses the structural description for the datum which is closest to grammatical (among those with correct overt form); choosing this 'winner' for Constraint Demotion entails that the smallest demotion is required to render this winner optimal. Thus RIP/CD preserves a crucial property of Constraint Demotion: the reranking performed is always the minimal change that allows correct analysis of the given datum.

9. Acquisition of Inputs

The final topic we take up is acquisition of grammatical inputs. According to the principle of richness of the base (55), 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, 1995), and null subjects (Grimshaw and Samek-Lodovici 1995, in press, 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 usually 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.[18]

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

(66) 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.[19]

For the moment, we retain the original context of this principle, and assume the correct grammar has already been learned; lexicon optimization is then used to acquire new lexical entries.

We take this principle as our starting point, and extend it to cope with phonological alternations. Consider the alternations due to syllable-final devoicing in German, for example:

[18] 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.

[19] 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: even when there is no grammatical option, the maximal-Harmony (but ungrammatical) structure is used nonetheless.

(67)	a. [tak]	‘day’	NOM SING
	b. [tag+ə]	‘days’	NOM PL

Presented with (67a) only, lexicon optimization chooses /tak/ as the underlying form of ‘day’: assuming the correct (devoicing) grammar, there are two underlying forms to choose from: /tak/ and /tag/. Both surface as [tak], but /tag/ does so with marks: Faithfulness violations (the voicing feature in the final underlying segment of /tag/ is not realized in the surface form [tak]). The less marked structural description thus arises from the underlying form /tak/, which is therefore selected by lexicon optimization. In general, lexicon optimization (in its simplest form) minimizes deep/surface disparities in its selection of underlying forms.

But if presented with (67b) only, minimizing deep/surface disparities will lead lexicon optimization to choose /tag/ for ‘day.’ Thus in its bare formulation, this principle is indeterminate in the face of alternations. As this example clearly shows, we really need to apply lexicon optimization not to individual forms, but to entire *paradigms*.

This conclusion converges with several lines of research in OT phonology which, independently of any learning considerations, point to the conclusion that, in general, grammatical optimization needs to be performed at the level of the paradigm. In a variety of contexts (e.g., ‘cyclic effects’), phonological explanation seems to require identity of the expression of a morpheme across its paradigm: what Burzio (1993, 1994, 1995ab, 1996) has termed *anti-allomorphy*, or in other terms, Faithfulness constraints holding between pairs of outputs in the same paradigm (Benua 1995, Buckley 1995, Flemming and Kenstowicz 1995, Kenstowicz 1994, 1995, McCarthy 1995, Gafos 1996). We’ll call such constraints OO-Faith for ‘output/output faithfulness’.

Interestingly, paradigm-level optimization in phonology proper is typically needed in cases where expected alternation does *not* occur (*anti*allomorphy); paradigm-level *lexicon* optimization is needed in cases when alternation *does* occur. In (67), for example, OO-Faith constraints demanding identity of expression of German ‘day’ in different environments must be out-ranked by the constraints that enforce syllable-final devoicing. Adopting the analysis of syllable-final devoicing of Lombardi 1995, we have the miniature *paradigm tableau* in (68). OnsFaith requires faithfulness in onset segments; this positionally-sensitive faithfulness constraint is the key to Lombardi’s analysis of coda devoicing: *Voi prohibits voicing generally, but this is overridden in the onset by faithfulness to underlying voicing.

(68) Paradigm tableau with phonological alternation

	/tag/ + $\left\{\begin{matrix}\text{Ø}\\ \text{ə}\end{matrix}\right\}$	*overt part*	OnsFaith	*Voi	Faith	OO-Faith
a. ☞	$\left\{\begin{matrix}.\text{tag}_{\langle\text{voi}\rangle}.\\ .\text{ta.g+ə}.\end{matrix}\right\}$	$\left\{\begin{matrix}[\text{tak}]\\ [\text{tagə}]\end{matrix}\right\}$		 *	* 	*
b.	$\left\{\begin{matrix}.\text{tag}.\\ .\text{ta.g+ə}.\end{matrix}\right\}$	$\left\{\begin{matrix}[\text{tag}]\\ [\text{tagə}]\end{matrix}\right\}$		* *		
c.	$\left\{\begin{matrix}.\text{tag}_{\langle\text{voi}\rangle}.\\ .\text{ta.g}_{\langle\text{voi}\rangle}\text{+ə}.\end{matrix}\right\}$	$\left\{\begin{matrix}[\text{tak}]\\ [\text{takə}]\end{matrix}\right\}$	*		* *	

This tableau shows how the choice of underlying form /tag/ is parsed by the correct (devoicing) grammar, in the part of its paradigm corresponding to nominative singular and plural. Three candidate paradigm fragments are shown. In the first, *a*, in the singular, the voicing feature of the final /g/ is unparsed (thus unrealized): the surface form is [tak]; in the plural, this feature is parsed. This candidate paradigm violates three constraints: *Voi, which prohibits voicing; general faithfulness constraints denoted 'Faith', in virtue of failing to parse the underlying feature; and the OO-Faith constraint, since the expression of 'day' differs in singular and plural. Despite these three violations, *a* is the optimal candidate paradigm. Candidate paradigms *b* and *c* both obey OO-Faith: in *b*, 'day' surfaces as [tag] in both singular and plural, while in *c*, it surfaces uniformly as [tak]. However *b* loses to *a* because *a* better satisfies *Voi; *c* loses to *a* because *c* violates top-ranked OnsFaith.[20]

Our interest in paradigm optimization here concerns the acquisition of underlying forms. We propose to apply lexicon optimization at the paradigm level: the underlying form of a morpheme is the one, among all those that give the correct surface forms, which yields the maximum-Harmony paradigm.

(69) Lexicon optimization tableau for [tak] ~ [tagə]

				overt part	OnsFaith	*Voi	Faith	OO-Faith
a. ☞	/tag/ + $\left\{\begin{matrix}Ø \\ ə\end{matrix}\right\}$	→	$\left\{\begin{matrix}.tag_{\langle voi\rangle}. \\ .ta.g+ə.\end{matrix}\right\}$	$\left\{\begin{matrix}[tak] \\ [tagə]\end{matrix}\right\}$		*	*	*
b.	/tak/ + $\left\{\begin{matrix}Ø \\ ə\end{matrix}\right\}$	→	$\left\{\begin{matrix}.tak. \\ .ta.k_{\boxed{voi}}+ə.\end{matrix}\right\}$	$\left\{\begin{matrix}[tak] \\ [tagə]\end{matrix}\right\}$	*	*	*	*

In this lexicon optimization tableau[21], we compare two underlying forms for 'day,' /tag/ and /tak/, both parsed so that they surface with the correct surface form paradigm, {[tak], [tagə]}. In the first candidate paradigm, the alternation is achieved by underparsing an underlying voice feature of the final /g/, while in the second candidate paradigm, alternation occurs through *over*parsing: *Gen* has added a voice feature to the underlying /k/. In the second candidate, the unfaithful parsing occurs in onset position, while in the first candidate paradigm it appears in coda position. Thus in addition to the constraint violations of the first candidate paradigm, the second candidate paradigm incurs a violation of OnsFaith: *b* is thus less harmonic than *a*. Lexicon optimization picks /tag/ as the underlying form because it gives rise to the most harmonic structural description of the paradigm.

So far, we have assumed that the correct grammar has already been learned. In reality, of course, underlying forms and grammars must be learned simultaneously, to a considerable extent. In the case of phonology, the interpretive parser required in the RIP/CD algorithm for acquiring the grammar must have access to the lexicon of underlying forms: the parser must know, for example, that the correct

[20] We have simplified slightly here; Lombardi's actual constraints are as follows. IDOnsLar requires that an output consonant preceding a tautosyllabic sonorant/vowel agree in Laryngeal features with its corresponding input segment ('OnsFaith'); *Lar penalizes Laryngeal features ('*Voi'); and MaxLar requires any Laryngeal features in the input to appear in its corresponding output segment ('Faith'). Output-output faithfulness constraints ('OO-Faith') do not figure in her analysis.

[21] See also the lexicon optimization 'tableaux des tableaux' of Itô, Mester and Padgett 1995.

interpretive parse of [tak] is .tag$_{\langle\text{voi}\rangle}$. and not .tak., because the underlying form is /tag/ and not /tak/.

This adds one additional dimension of complexity to the full iterative algorithm ultimately needed to acquire phonology. *Three* components must now function in concert: a robust parser which adds hidden structures to overt learning data, assuming the current grammar and lexicon of underlying forms (the RIP component); a grammar learner, which reranks constraints to make optimal the structural descriptions generated by the parser (the CD component); and a lexicon learner, which takes the current grammar and overt learning data and derives underlying forms (see Figure 3). For the last component, we propose lexicon optimization, operating at the level of the morphological paradigm (the 'paradigmatic lexicon optimization' or 'PLO' component). As described in Section 2, the overall strategy deriving from iterative model-based learning algorithms is to design components which correctly compute their answers, assuming the other components have correctly computed their answers—and then to show that, starting from an initial guess, the components iteratively converge to a correct state. This is one of the next steps in the research program developed here: study of the three-component RIP/CD/PLO algorithm.[22]

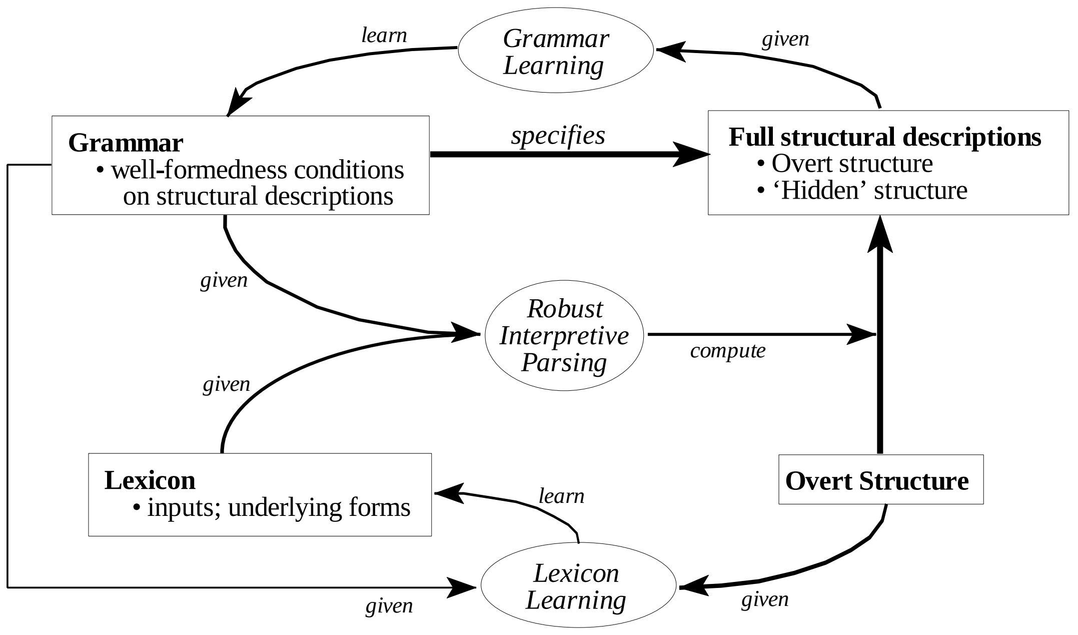

Figure 3. Problem decomposition, including lexicon.

[22] In this context it is clear why the version of lexicon optimization needed must allow for situations in which no outputs optimal for the currently hypothesized, incorrect, grammar have the correct overt form (see note 19). If the current grammar were not a coda-devoicing grammar (or an intervocalic voicing grammar), then no underlying form for 'day' would produce the correct surface paradigm. As formulated here, lexicon optimization considers all outputs with the correct overt form, even if they are not optimal; thus outputs like (69ab) would still be considered, and some underlying form for 'day' would be hypothesized: the one yielding the maximum-Harmony output paradigm given the currently hypothesized grammar. If the choice is /tag/, subsequent learning steps will rerank constraints to produce a coda-devoicing grammar.

The components of the learning system we have proposed are all strongly shaped by the optimization character of the grammar being acquired. Lexicon optimization, robust interpretive parsing, and the optimization usually taken to define the operation of an OT grammar, production-directed parsing, are all merely different ways of accessing the evaluative structure which is an OT grammar. An OT grammar assigns a Harmony value to a structural description or parse *p* which includes an input *I* and an overt or 'phonetic' part φ. As shown in (70), in this paper we have exploited three different ways of maximizing Harmony.

(70) Three directions of Harmony optimization
/*I*/ = input
.*p*. = structural description
[φ] = overt, 'phonetic' form (with morphological structure)
{/*I*/} = lexicon of underlying forms

	Type of Optimization	Given …		Compute …	Examples		
a.	Production-Directed Parsing	/*I*/	→	.*p*., [φ]	/tag/ /LLH/	→ →	.tag$_{\langle\text{voi}\rangle}$. , [tak] (ĹL)H , [ĹLH]
b.	Paradigmatic Lexicon Optimization	[φ]	→	.*p*., /*I*/	{ [tak] [tag+ə] }	→	{ .tag$_{\langle\text{voi}\rangle}$. .ta.g+ə. } , /tag/ + { Ø ə }
c.	Robust Interpretive Parsing	[φ], {/*I*/}	→	.*p*.	[tak], {/tag/,…} [ĹLH]	→ →	.tag$_{\langle\text{voi}\rangle}$. (ĹL)H

In its usual operation, the OT grammar takes an input *I* and produces as output the structure *p* (including overt part φ) which has maximal Harmony, among all structures that parse the given input *I* (70a). For example, given an input /tag/, in the grammar of (68), .tag$_{\langle\text{voi}\rangle}$. (overt: [tak]) is the structural description with greater Harmony than any other parse of /tag/; given the input LLH, and the grammar of (65), the structure (ĹL)H (overt: [ĹLH]) has greater Harmony than any other parse of this input.

In lexicon optimization, we start with a surface form φ, and find the input *I* and complete structural description *p* that maximize Harmony, among all structures with overt part φ: this was just illustrated for the example shown in (70b).

Robust interpretive parsing is yet a third direction in which to perform optimization. Given an overt form φ, we find the structural description with overt part φ which has maximal Harmony, considering only inputs found in the given lexicon {/*I*/}. The first example shown in (70c) was mentioned a few paragraphs earlier: assuming a grammar for German, the interpretive parse that has maximal Harmony among all those with the given phonetic form [tak] is of course .tak.; but restricting to underlying forms in the German lexicon, which lacks /tak/, the maximal Harmony form is .tag$_{\langle\text{voi}\rangle}$., with underlying form /tag/. Robust interpretive parsing was also illustrated in Section 8: given a phonetic form [ĹLH] and grammar (65), the structure (ĹL)H has greater Harmony than any other foot structure with that stress pattern.

Thus we see that the particular structure of grammar under Optimality Theory—optimization relative to a hierarchy of constraints—enables us to intimately tie learning the lexicon of underlying forms to the basic operation of the grammar—pairing output structures to inputs—as well as to the assignment of hidden structure to overt learning data. All three are simply different ways of deploying the core function of the grammar: assessing the Harmony of structural descriptions.

10. Concluding Discussion

10.1 Parametric Independence and Linguistic Explanation

Before summing up, let us consider an important learnability consequence of the different conceptions of cross-linguistic variation found in Optimality Theory and in the Principles-and-Parameters framework. In P&P theory, 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. Work on learnability focuses on the relationship between data and the parameter values, usually discussed in terms of *triggers*. A trigger is a datum (e.g., for syntax, a type of sentence) which indicates the appropriate value for a specific parameter (see, for example, the definitions of trigger in Gibson and Wexler 1994, Frank and Kapur 1996). It is significant that a trigger provides information about the value of a single parameter, rather than relationships between the values of several parameters.[23] 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

[23] 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.

progress for both theoretical explanation and learnability. Clearly, this is a desirable property for a theoretical framework.

10.2 Summary

An Optimality Theoretic grammar is a ranked set of violable constraints which defines a notion of relative Harmony of structural descriptions, the maximally harmonic or optimal structures being the grammatical ones. The consequences of constraint hierarchies for surface patterns can be quite subtle and often surprising. Remarkably different surface patterns can emerge from the reranking of the same set of universal constraints.

All this is integral to the explanatory power of OT as a linguistic theory. But it also raises concerns about learnability. If the relations between grammatical forms and grammars is so complex and opaque, how can a child cope?

Linguists working in OT are frequently faced with a hypothesized set of universal constraints and a collection of surface forms which they have given hypothetical structural descriptions; the question is, is there a ranking of the constraints that yields the correct structures? Typically, this turns out to be a challenging question to answer. Of course, with even a modest number of constraints, the number of possible rankings is much too large to explore exhaustively.

So the starting point of the present research is the question, are there reliable, efficient means for finding a ranking of a given set of constraints which correctly yields a given set of grammatical structural descriptions? As we have seen in Section 5, the answer is yes, if the learner is given informative pairs of optimal structures with suboptimal competitors. For any set of such data pairs consistent with some unknown total ranking of the given constraints, Constraint Demotion finds a stratified hierarchy consistent with all the data pairs.

A key to these results is the implicit negative evidence that comes with each positive example: all the universally-given competitors to each optimal structure (excluding any which may have identical constraint violations); these are guaranteed to be suboptimal and therefore ill-formed. The pairs of optimal forms and suboptimal competitors are the basis of Constraint Demotion: the constraints violated by the optimal form are minimally demoted to lie below some constraint violated by the suboptimal form (excluding cancelled marks).

Is it necessary that informative suboptimal forms be provided to the learner? As we saw in Section 6, the answer is no. Given a grammatical structural description as a learning datum, the learner can identify the input in the structural description, and compute the optimal parse of that input using the currently hypothesized hierarchy: that parse can be used as the suboptimal competitor, unless it is equal to the given parse, in which case the example is not informative—no learning can occur. This is Error-Driven Constraint Demotion.

In the cases mentioned so far, the learning procedure can be proved to converge to a correct hierarchy, as shown in the Appendix. These results all assume that learning begins with no relative ranking of constraints: all constraints begin at the top of the hierarchy, and some are then demoted. Does learnability depend on this assumption about the initial state? As explained in Section 7.1, the answer is no; the same results can be shown to follow when the initial state is any arbitrary hierarchy.

As discussed in Section 5.1, learning in all these cases is efficient, in the sense that the number of informative examples, or number of learning operations (demotions), is guaranteed to be no more than $N(N–1)$, where N is the number of constraints. This grows quite modestly with N, and is vastly less than the number of grammars, $N!$.

This brings us to the current frontier of the formal results. In addition to demonstrating these results, we have also discussed some implications of these results, and made several proposals for how the results might be further extended.

The most important extension addresses the question, must full structural descriptions of positive examples be provided to the learner? The Constraint Demotion algorithms operate on constraint violations or marks, and these can be determined only from full structural descriptions. In Section 2, we proposed that the learner, given only the overt part of grammatical structures, can compute the full structural description needed for Constraint Demotion by using robust interpretive parsing: using the currently hypothesized grammar, the learner finds the maximal-Harmony structural description consistent with the overt form (and the currently hypothesized lexicon). Such parsing is a necessary part of the overall theory anyway, independent of learning, since grammar *users* must perform it when interpreting overt forms. Coupling interpretive parsing to the Constraint Demotion solution to the problem of learning a grammar from full structural descriptions yields an algorithm, RIP/CD, a new member of the family of iterative model-based solutions to the general problem of learning hidden structure. In other learning domains, these solutions have been highly successful in both theory and practice. In Section 8, we gave a small illustration of how this might look in the domain of acquiring stress.

In the case of phonology acquisition, must the learner be provided with the lexicon of underlying forms (necessary for interpretive parsing, as well as the inputs to production-directed grammatical parsing)? In Section 8, we proposed that, as part of the same iterative process that is adapting the grammar to accomodate the structural descriptions produced by interpretive parsing (RIP/CD), the learner can incrementally learn the lexicon via lexicon optimization at the level of the morphological paradigm. At each stage of learning, the current grammar is used to find the underlying form for morphemes which yields the maximum-Harmony structural descriptions for paradigms. We provided a miniature example of how this can work in the face of phonological alternation.

Taken as a whole, in this paper we have developed and illustrated a proposal for how a learner, provided with the universal elements of any OT UG system, and the overt parts of forms grammatical with respect to some grammar admitted by that UG, could learn the grammar, the structural descriptions, and the lexicon. This proposal decomposes the problem into three subproblems: robust interpretive parsing, lexicon learning, and grammar learning. Currently, we have a set of formal results on the grammar learning subproblem.

How do these learnability considerations relate to OT work on actual acquisition? We have considered the question of the initial state, and reviewed a 'subset'-type argument which uses the OT principle of richness of the base to show that in general FAITHFULNESS constraints must be low-ranked in the initial state if unmarked inventories are to be learnable. The concept of robust interpretive parsing developed here makes sense of the proposal that children's inputs are essentially the correct adult forms. This connects a fundamental principle of OT and learnability considerations to two important assumptions of much OT research on phonological acquisition: initial low-ranking of FAITHFULNESS and the hypothesis that children's inputs closely approximate the adult forms.

And finally, how does the emerging OT learning theory relate to linguistic explanation? In Section 10.1 we observed that in OT, constraint interaction is simultaneously the key to both linguistic explanation and learnability: constraint conflict, resolved by language-specific ranking, provides both the explanatory power of OT as a linguistic theory, and the evidence learners need to home in on their target grammar.

How, exactly, does a theory of grammar bear on questions of learnability? This paper provides evidence that Optimality Theory's claims about the structure of Universal Grammar have manifold implications for learning. The claim that constraints are universal entails that the learner can use a given set of constraints to evaluate structural descriptions. The claim that grammatical structures are optimal, and grammars are total rankings of violable constraints, entails that with every piece of explicit positive evidence comes a mass of implicit negative evidence, and that constraints can be ranked so that those violated by positive data are dominated by those violated by implicit negative data. The claim that grammars are evaluators of structural descriptions provides a uniform basis for the problems of parsing overt forms to determine their hidden structure, parsing inputs to determine their grammatical output, and deducing new inputs for insertion into the lexicon: these are merely three different directions of accessing the evaluative structure that is the grammar. The claim of richness of the base connects the OT basis for adult typologies with fundamental hypotheses underyling acquisition research.

All these implications follow not from a particular OT theory of stress, nor an OT theory of phonology, but from the fundamental structure which OT claims to be inherent in all of grammar. Our learning algorithms derive from this general grammatical structure alone, and so apply to the learning of any OT grammar. At the same time, our algorithms are not generic search procedures, uninformed by a theory of grammar, equally applicable to the problem of learning to classify submarines. The special, characteristically linguistic, structure imposed by OT on UG is sufficiently strong to allow the proof of learnability theorems which state that large spaces of possible grammars can be efficiently navigated to home in on a correct grammar.

11. Appendix: Correctness and Data Complexity of Constraint Demotion

The formal analysis of Constraint Demotion learning proceeds as follows. A language L is presumed, which is generated by some total ranking. Section 11.1 sets up the basic machinery of stratified constraint hierarchies. Section 11.2 identifies, for any language L, a distinguished stratified hierarchy which generates it, the *target hierarchy* $\mathcal{H}_L$. Section 11.3 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 11.4 extends the results to arbitrary initial constraint hierarchies. Section 11.5 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. Section 11.6 discusses a family of Constraint Demotion algorithms, including Recursive Constraint Demotion, which is of considerable practical value to the working linguist for automatically ranking constraints, or determining that no ranking exists which is consistent with a body of data.

11.1 Stratified Hierarchies

(71) **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.

(72) **Def.** A *total ranking* is a stratified hierarchy where each stratum contains precisely one constraint.

(73) **Def.** A constraint $\mathbb{C}_1$ is said to dominate constraint $\mathbb{C}_2$, denoted $\mathbb{C}_1 \gg \mathbb{C}_2$, in hierarchy $\mathcal{H}$ if the stratum containing $\mathbb{C}_1$ dominates the stratum containing $\mathbb{C}_2$ in hierarchy $\mathcal{H}$.

(74) **Def.** The *offset* of a constraint $\mathbb{C}$ in a hierarchy $\mathcal{H}$ is the number of strata that dominate the stratum containing $\mathbb{C}$. $\mathbb{C}$ is *in a lower stratum* in $\mathcal{H}_1$ than in $\mathcal{H}_2$ if the offset of $\mathbb{C}$ in $\mathcal{H}_1$ is greater than in $\mathcal{H}_2$. $\mathbb{C}$ is *in the same stratum* in $\mathcal{H}_1$ and $\mathcal{H}_2$ if it has the same offset in both.

(75) **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 $\mathcal{H}_1$.

(76) **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 $\mathcal{H}_2$.

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

(78) **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. ❐

11.2 The Target Stratified Hierarchy

(79) **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 uncancelled marks to any optimal structural description in *L*. The second stratum consists of precisely those constraints which assess uncancelled marks only to optimal parses relative to competitors which are assessed at least one uncancelled 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 uncancelled marks to optimal parses relative to competitors assessed an uncancelled mark by at least one constraint ranked higher in the target.

(80) **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. ❒

(81) **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 uncancelled mark to *winner*, *loser* is assessed an uncancelled mark by a constraint $\mathbb{C}_{n-1}$ with offset precisely $n-1$, and *loser* is not assessed any uncancelled 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 uncancelled mark to an optimal description relative to some suboptimal competitor, it must be dominated by some other constraint which assesses an uncancelled 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 uncancelled 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 uncancelled mark by $\mathbb{C}_n$relative to some *loser*, *loser* is assessed an uncancelled 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. ❒

(82) **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 uncancelled marks, it is more harmonic than *loser*. If *winner* has an uncancelled mark, then its corresponding constraint must be dominated in $\mathcal{H}_L$ by some constraint assessing an uncancelled 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 uncancelled 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 uncancelled mark to *winner*, *loser* must be assessed an uncancelled 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 $\mathcal{H}'$. ❐

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

Proof By the definition of $\mathcal{H}_L$ (79), for every loser/winner pair of L, each uncancelled winner mark is dominated, with respect to $\mathcal{H}_L$, by an uncancelled loser mark. By the definition of refinement (76), 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. ❐

11.3 Constraint Demotion

(84) **Def.** The mark cancellation procedure, MarkCancel(*marks*(*loser*), *marks*(*winner*)), is:

```
For each occurrence of *$\mathbb{C}$ in both marks(loser) and marks(winner)
    Remove that occurrence of *$\mathbb{C}$ from both lists
Return the lists as (marks′(loser), marks′(winner))
```

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

```
Set $\mathcal{H}'$ to $\mathcal{H}$
Find the constraint $\mathbb{C}_l$ with a mark in marks′(loser) ranked highest in $\mathcal{H}'$
For each $\mathbb{C}_w$ with a mark in marks′(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 $\mathcal{H}'$
```

(86) **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. ❐

(87) **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 $\mathbb{C}_w$ be dominated by one of the constraints assessing uncancelled marks to *loser*. Let $\mathbb{C}_l$ 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 $\mathbb{C}_l$ in $\mathcal{H}_L$ has offset $\geq n$. Thus, every constraint assessing an uncancelled mark to *loser* must have offset $\geq$ n. Therefore, $\mathbb{C}_w$ must have offset at least $n+1$ in $\mathcal{H}_L$. CD demotes $\mathbb{C}_w$ to the stratum immediately below the one containing $\mathbb{C}_l$, so $\mathbb{C}_w$ has offset $n+1$ in the resulting hierarchy. Thus, $\mathbb{C}_w$ has offset in the output hierarchy less than or equal to its offset in $\mathcal{H}_L$, guaranteeing that the output hierarchy h-dominates $\mathcal{H}_L$. ❐

(88) **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.

(89) **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 $\mathbb{C}$, of the difference between the offset of $\mathbb{C}$ in $\mathcal{H}_1$ and in $\mathcal{H}_2$.

(90) **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 (86), the input hierarchy h-dominates the output hierarchy. Let $\mathbb{C}$ be a constraint that is demoted, with offset n in the input hierarchy, offset m in the output hierarchy, and offset t in $\mathcal{H}_L$. $\mathbb{C}$ is demoted, so $m>n$. By lemma (87), 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. ❐

(91) **Lemma** Let N be the number of constraints. The h-distance from $\mathcal{H}_0$ to $\mathcal{H}_L$ cannot exceed ½$(N-1)N$.

Proof By lemma (78), $\mathcal{H}_0$ 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:

$$(N-1) + (N-2) + ... + 1 + 0$$

which is precisely ½$(N-1)N$. ❐

(92) **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 ½$(N-1)N$ informative pairs.

Proof By lemma (90), 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 (91) guarantees that this distance is at most ½$(N-1)N$. ❐

11.4 Extension to Arbitrary Initial Hierarchies

We now consider CD starting from some arbitrary initial hierarchy, denoted $\mathcal{H}_0$. 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.

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

(94) **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 $\mathcal{H}_L$:

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

(95) **Def.** During the operation of the CD algorithm, let *D* denote the h-distance (89) between the algorithm's current stratified hierarchy $\mathcal{H}$ and the lower bounding hierarchy $\mathcal{H}^{\#}$:

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

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

Proof $\mathcal{H}_L$ correctly accounts for all the data in *L* (82) so in any loser/winner pair each uncancelled winner mark must be dominated by an uncancelled loser mark. ❐

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

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

Let $\mathbb{C}$ be a constraint in the top stratum of $\mathcal{H}_L$, with $\nu(\mathbb{C},\mathcal{H}_L) = 0$. Then $\mathbb{C}$ will never be demoted by CD. For, by the definition of CD, such demotion would require that $*\mathbb{C}$ be an uncancelled mark of a winner, which is impossible for a constraint in the top stratum of $\mathcal{H}_L$. Thus for each constraint $\mathbb{C}$ with $\nu(\mathbb{C},\mathcal{H}_L) = 0$, $\mathbb{C}$ 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 $\mathbb{C}$ with $\nu(\mathbb{C},\mathcal{H}_L) = 0$ ends up where it starts, in a stratum with offset at most $K + 0 = K + \nu(\mathbb{C},\mathcal{H}_L) \equiv \nu(\mathbb{C},\mathcal{H}^{\#})$. This establishes the base case for the induction.

Now for the inductive step we assume that for all constraints $\mathbb{C}$ with $\nu(\mathbb{C},\mathcal{H}_L) < k$, the CD algorithm never demotes $\mathbb{C}$ below offset $\nu(\mathbb{C},\mathcal{H}^{\#})$. Let $\mathbb{C}$ be a constraint with $\nu(\mathbb{C},\mathcal{H}_L) = k$. By Lemma (96), for each loser/winner pair in which $*\mathbb{C}$ is an uncancelled winner mark, $\mathbb{C}$ is dominated in $\mathcal{H}_L$ by some $\mathbb{C}'$ such that $*\mathbb{C}'$ is an uncancelled loser mark. This implies that $\mathbb{C}'$ has a lower offset than $\mathbb{C}$ in $\mathcal{H}_L$, and it follows that $\nu(\mathbb{C}',\mathcal{H}_L) < k$. Thus, by the inductive hypothesis, $\mathbb{C}'$ is demoted to a stratum no lower than

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

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

(98) **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 (97). That D monotonically decreases during the execution of CD is obvious, since CD only demotes constraints, which can only decrease D. ❐

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

Proof By Lemma (98), 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 (91):

$$2K + 2(K-1) + \cdots + 0 = 2(N-1) + 2(N-2) + \cdots + 0$$
$$= 2[(N-1) + (N-2) + \cdots + 0] = 2[N(N-1)/2] = N(N-1)$$

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. ❐

11.5 Error-Driven Constraint Demotion

(100) **Def.** The Error-Driven Constraint Demotion algorithm 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
```

(101) **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 (99), 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. ❐

Theorem (101) 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.

11.6 Recursive Constraint Demotion and Other Constraint Demotion Algorithms

The Recursive Constraint Demotion algorithm is first exemplified using CVT; a general definition and analysis of the algorithm is then provided. A family of Constraint Demotion algorithms is next defined, and related to the Recursive and Error-Driven variants.

The Recursive Constraint Demotion (RCD) Algorithm: An Example

As with all Constraint Demotion algorithms, the starting point is the initial data, a set of loser/winner mark data pairs, with common marks cancelled. For our CV language L_1, these data are given in (33), and in tableau form in (32); they are repeated here:

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

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

(103) Initial data in tableau form (L_1)

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

After mark cancellation, the remainder of the algorithm proceeds recursively, finding first the constraints that may be ranked highest while being consistent with the mark-data pairs, then eliminating those constraints from the problem and starting over again to rank the remaining, lower, constraints. Conceived as a sequence of passes, the first pass through the data determines the highest-ranking constraints, the next pass the next-highest ranking constraints, and so forth down the hierarchy. If the data provide enough information to completely determine the total ranking, then only one constraint will be returned by each pass. In general, however, the result of the algorithm will be a stratified hierarchy.

When the algorithm begins, the not-yet-ranked constraints comprise the entire universal set (19):

not-yet-ranked-constraints = {ONSET, NOCODA, PARSE, FILL^{Nuc}, FILL^{Ons}}

Examining the rightmost column of the mark-data table (102) we see that two marks, *Parse and *Fill^{Ons}, appear in the list of uncancelled winner marks: these are the constraints whose columns in (103) contain uncancelled '⊛' marks. So the two constraints Parse and Fill^{Ons} must be dominated by other constraints (those violated by the corresponding losers); they cannot be the highest-ranked of the *not-yet-ranked-constraints*. The remaing constraints, on the other hand, *can* be top-ranked, because they have assess no uncancelled marks to any winners. Thus:

(104) *highest-ranked-constraints* = {Onset, NoCoda, Fill^{Nuc}}

This constitutes the output of the first pass: these three constraints form the highest stratum in the hierarchy. The data do not support any distinctions in ranking among the three, so none are made. Now,

(105) *not-yet-ranked-constraints* = {Parse, Fill^{Ons}}

The result of the first pass is shown in tableau form in (105). The columns with uncancelled winner marks ⊛ have been demoted below those without; this is all there is to it.

(106) Hierarchy after pass one

			already-ranked			*not-yet-ranked*	
winner/loser pairs			Fill^{Nuc}	Onset	NoCoda	Fill^{Ons}	Parse
d	☞✓	.□V.CV.⟨C⟩				⊛	⊛
a		.V.CVC.		*!	*!		
d	✓	.□V.CV.⟨C⟩				⊛	⊛
b		⟨V⟩.CV.⟨C⟩					* *
d	☞✓	.□V.CV.⟨C⟩				⊛	⊛
c		⟨V⟩.CV.C□́.	*!				*

Now that the highest ranked constraints have been determined, the list of mark-data pairs can be trimmed down by removing any mark-data pairs that are completely accounted for by the constraints selected as highest. This is the case if at least one of the marks incurred by the loser of a pair is among the highest ranked constraints. Such a mark is guaranteed to dominate all of the corresponding winner's marks, because all of the winner's marks were disqualified from being ranked highest. In the tableau (105), we see that *d* is now more harmonic than *a*, which earns two fatal uncancelled marks in the top stratum; these marks render irrelevant the winner marks in the lower stratum, as shown by the grey shading. Similarly, *d* is more harmonic than *c*. But the hierarchy at this point does not yet make *d* more harmonic than *b*: their two Parse marks cancel, and in addition, the second Parse mark of *b* cancels the Fill^{Ons} mark of *d*, since these two constraints are currently equally-ranked.

So to proceed to further ranking, we eliminate from the mark-data table those winner/loser pairs which are now accounted for; this amounts to eliminating every row in which any of the highest-ranked constraints appear. So we eliminate the pair $a \prec d$ because *Onset appears (or, alternatively, because *NoCoda appears), and also the pair $c \prec d$, because *Fill^{Nuc} appears. The new mark-data table is thus:

(107) Mark-data pairs (L_1, after first pass)

sub-opt ≺ *opt*				*loser-marks*	*winner-marks*
b ≺ *d*	〈V〉.CV.〈C〉	≺	.□V.CV.〈C〉	{~~*PARSE~~ *PARSE}	{~~*PARSE~~ *FILL^{Ons}}

In tableau form, the remaining data pair is:

(108) Data after first pass

			already-ranked			*not-yet-ranked*	
winner/loser pairs			FILL^{Nuc}	ONSET	NOCODA	FILL^{Ons}	PARSE
d	✓	.□V.CV.〈C〉				⊛	▨⊛
b		〈V〉.CV.〈C〉					▨* *

At the end of the first pass, we now have the first (highest) stratum (29a), and a reduced list of not-yet-ranked constraints (29b), and a reduced mark-data table (29c). Crucially, the reduced mark-data table involves only the *not-yet-ranked-constraints*, so we can now recursively invoke the same algorithm, using the remaining data to rank the remaining constraints. This initiates the next pass.

Repeating this process with the reduced mark-data table (29c), we examine the rightmost column of the table, and observe that of the two *not-yet-ranked-constraints* (29b), only one, FILL^{Ons}, appears. The remaining constraint, then, is output as the next stratum of highest-ranked constraints:

(109) *highest-ranked-constraints* = {PARSE}

This leaves

(110) *not-yet-ranked-constraints* = {FILL^{Ons}}

In tableau form, the column with the winner mark ⊛ is demoted, ensuring that *d* is more harmonic than *b*:

(111) After second pass

			already-ranked				*not-yet-ranked*
winner/loser pairs			FILL^{Nuc}	ONSET	NOCODA	PARSE	FILL^{Ons}
d	☞✓	.□V.CV.〈C〉				▨⊛	⊛
b		〈V〉.CV.〈C〉				▨* *!	

The final step of the second pass is to trim the mark-data table, eliminating rows in which the *highest-ranked-constraints* appear. This eliminates the only row in the table, so that the new mark-data table is empty.

(112) Mark-data pairs (L_1, after second pass): none

The result of the first two passes is the highest segment of the stratified hierarchy:

{Onset, NoCoda, Fill^{Nuc}} ≫ {Parse}

The third pass operates on an empty mark-data table. Since there are no marks in the rightmost column of such a table, no remaining constraints must be dominated: all the not-yet-ranked constraints are output as the highest-ranked. In this case, that is the one remaining constraint Fill^{Ons}.

(113) *highest-ranked-constraints* = {Fill^{Ons}}

This leaves

(114) *not-yet-ranked-constraints* = {}

so the algorithm terminates, with the final result:

(115) Learned Stratified Hierarchy for L_1:
{Onset, NoCoda, Fill^{Nuc}} ≫ {Parse} ≫ {Fill^{Ons}}

This result represents a class of totally-ranked constraint hierarchies all of which give rise to the target language L_1: the same optimal outputs arise regardless of the ranking of the three highest constraints. One of these refinements of the learned stratified hierarchy (115) is the particular total ranking given in (25): this is but one of the correct hierachies for the target language.

The final result is shown in tableau form in (116), where it may be verified that every winner/loser pair is now correctly accounted for by the hierarchy:

(116) Learned hierarchy for L_1 and learning data

			already-ranked				
	winner/loser pairs		Fill^{Nuc}	Onset	NoCoda	Parse	Fill^{Ons}
d	☞✓	.□V.CV.⟨C⟩				⊛	⊛
a		.V.CVC.		*!	*!		
d	☞✓	.□V.CV.⟨C⟩				⊛	⊛
b		⟨V⟩.CV.⟨C⟩				* *!	
d	☞✓	.□V.CV.⟨C⟩				⊛	⊛
c		⟨V⟩.CV.C□́.	*!			*	

It is easy to see how the course of learning L_2 differs from that of L_1, assuming the learner's initial datum is the parse of the same input, /VCVC/, which is now ⟨V⟩.CV.C□́. (candidate *c* in (16); see the tableau (27)). The mark-data table used by the algorithm, containing the marks of *sub-opt* ≺ *opt* pairs after cancellation, is now:

(117) Mark-data Pairs after Cancellation (L_2)

	sub-opt	≺	*opt*	*loser-marks*	*winner-marks*
$a \prec c$	.V.CVC.	≺	⟨V⟩.CV.C□́.	*Onset, *NoCoda	*Parse, *Fill^{Nuc}
$b \prec c$	⟨V⟩.CV.⟨C⟩	≺	⟨V⟩.CV.C□́.	~~*Parse~~ *Parse	~~*Parse~~ *Fill^{Nuc}
$d \prec c$	.□V.CV.⟨C⟩	≺	⟨V⟩.CV.C□́.	~~*Parse~~ *Fill^{Ons}	~~*Parse~~ *Fill^{Nuc}

This table is identical to its L_1 counterpart (102) except that the marks *Fill^{Nuc} and *Fill^{Ons} are interchanged. The result of the algorithm is therefore the same as before, (115), with this exchange made.

(118) Learned Stratified Hierarchy for L_2:

$$\{\text{Onset, NoCoda, Fill}^{\text{Ons}}\} \gg \{\text{Parse}\} \gg \{\text{Fill}^{\text{Nuc}}\}$$

Again, this stratified hierarchy is correct: its further refinements into totally-ranked hierarchies, including the one we singled out in (26), all give rise to L_2.

That these CV languages can each be learned completely from a single positive example attests to the power of the implicit negative data which comes with each positive example in Optimality Theory.

General Statement of the RCD Algorithm

Having illustrated the Recursive Constraint Demotion algorithm in the context of a particular linguistic theory, the Basic CV Syllable Structure Theory, we now formulate the algorithm in full generality.

(119) The Grammar Learning Problem

Given:

universal-constraints = a set of universal constraints

initial-data = a set of well-formed outputs of the target language *L*

where *L* arises from some (not necessarily unique) total ranking of the universal constraints.

To Find:

A stratified hierarchy in which these are the optimal parses of their corresponding inputs.

RCD solves this problem via an intermediate step in which *initial-data* is prepared for the algorithm.

(119) Data Preparation

Given:

universal-constraints = a set of universal constraints

initial-data = a set of well-formed outputs of the target language *L*

Generate:

a. For each output in *initial-data* (*opt*), choose a set of sub-optimal competitors (*sub-opt*)

b. Make a table of these pairs, *sub-opt* ≺ *opt*, listing the marks incurred by the *sub-opt*s in one column, *loser-marks*, and the marks incurred by *opt*s in another, *winner-marks*.

c. The result is *mark-data* =

sub-opt ≺ *opt*	*loser-marks* = *marks*(*sub-opt*)	*winner-marks* = *marks*(*opt*)
...	...	...

(119) Recursive Constraint Demotion (RCD)

```
    Given:
        universal-constraints = a set of universal constraints
        mark-data = a table of pairs of mark lists (loser-marks, winner-marks)
    To Find:
        A stratified hierarchy with respect to which each of the loser-marks is less harmonic
                than its corresponding winner-marks.

RCD Algorithm:
    Set not-yet-ranked-constraints = universal-constraints
    I. Mark Cancellation
        For each pair (loser-marks, winner-marks) in mark-data:
            a. For each occurrence of a mark *ℂ in both loser-marks and winner-marks
                    in the same pair, remove that occurrence of *ℂ from both.
            b. If, as a result, no winner-marks remain, remove the pair from mark-data.
    II. Recursive Ranking
        a. Output highest-ranked-constraints = all the constraints in not-yet-ranked-
                constraints which do not appear in the column winner-marks of mark-data;
                these form the highest-ranked stratum of the not-yet-ranked constraints.
        b. Remove the highest-ranked-constraints from the not-yet-ranked-constraints.
        c. Remove all rows from mark-data which contain any marks assessed by the highest-
                ranked-constraints.
        d. Call Recursive Ranking again, with the reduced mark-data and the reduced not-yet-
                ranked-constraints.
```

Note that in step c of Recursive Ranking, the relevant marks (those assessed by the *highest-ranked-constraints*) can only appear in the column *loser-marks*; for any constraint contributing a mark to the column *winner-marks* is not, by step a, among the relevant constraints (those in *highest-ranked-constraints*).

Analysis of the RCD Algorithm

Observe first that multiple uncancelled tokens of the same type of mark in the *mark-data* table, either for winner or loser, have the same effect as a single token. For in Recursive Ranking step a, we simply determine which constraints assess no marks at all in the *winner-marks* column: whether a single or multiple tokens of a mark appear makes no difference. Then in Recursive Ranking step b, a row is removed from *mark-data* if it contains any marks at all assessed by the *highest-ranked-constraints*; multiple tokens of a mark type have the same effect as a single token. Thus, for efficiency considerations below, we can assume that in Mark Cancellation Initialization step a, if a row of the *mark-data* table contains multiple tokens of the same type of mark after cancellation, duplicates are eliminated, leaving at most one token of each type. In other words, in the initial mark-data table prior to cancellation, what really matters is, for each constraint, which of *sub-opt* or *opt* incurs more violations of the constraint ℂ; if it is *sub-opt*, then a token of the mark *ℂ appears in the column *loser-marks*; if it is *opt*, then the token of *ℂ appears instead in the column *winner-marks*. What matters in the assessment of optimality is only which of two candidates more seriously violates each constraint: not any absolute magnitude of violation (see footnote 7).

The correctness of the algorithm should be clear. The *highest-ranked-constraints* output at each pass of the algorithm are exactly the constraints which need not be dominated in order to explain the available *data*; the remaining *not-yet-ranked-constraints*, by contrast, must be dominated and so cannot be highest-ranked. We now show that the algorithm must terminate.

On each pass of the algorithm, at least one constraint must be output. For suppose not. That would mean that every one of the *not-yet-ranked-constraints* appears in the column *winner-marks*, i.e., as an uncancelled mark of an optimal form. But that would mean that every one of the *not-yet-ranked-constraints* must be dominated by one of the other *not-yet-ranked-constraints*: which means there is no ranking of these constraints which is consistent with the *mark-data*, in contradiction to the basic assumption that the *mark-data* derive from some ranking of the given constraints.

It is worth noting at this point that if the initial data presented to the RCD algorithm are not consistent with any total ranking of the given constraints, the algorithm discovers this fact: at some stage of its execution, it fails at Step II.a to find a constraint which does not appear among the *winner-marks*.

So, under the assumption that the data are consistent with some total ranking of the given constraints, on each pass, at least one constraint is eliminated from the *not-yet-ranked-constraints*. Thus the number of passes required cannot be more than the number of universal constraints: call this N_{constr}. The number of steps required for each pass cannot exceed the number of uncancelled marks in the *mark-data* table: each mark in the column *winner-marks* is examined in Recursive Ranking step a to ensure that its corresponding constraint will not be output as a *highest-ranked-constraint*, and, in the worst case, each mark in the column *loser-marks* must be examined in Recursive Ranking step c to determine which rows must be eliminated from *data*. The number of uncancelled marks per row of the table can't exceed the number of constraints N_{constr}, so the total number of steps per pass can't exceed $N_{constr}N_{pairs}$, where N_{pairs} is the number of rows in the initial *mark-data* table, i.e., the number of pairs *sub-opt* $\prec$ *opt* used.

So the number of steps required for all the passes can't exceed

$$(N_{constr})^2 N_{pairs}$$

Thus the algorithm is quite efficient: in the worst case, the RCD algorithm is quadratic in the number of constraints, and linear in the number of mark-data pairs.

In summary, then, we have established the following result.

(120) Theorem: Correctness and data complexity of Recursive Constraint Demotion

Suppose given a total ranking of N_{constr} constraints in a set *Con*, and N_{pairs} data pairs from this ranking, each an optimal structural description of some input paired with a competing parse of the same input. Given this data, the Recursive Constraint Demotion algorithm converges to a stratified constraint hierarchy consistent with the data, the number of operations required being at most

$$(N_{constr})^2 N_{pairs}$$

General Constraint Demotion

We now isolate the core part of the RCD algorithm, which we call the Core CD Algorithm, and show how it can be used in a number of ways to yield a family of CD algorithms which differ primarily in how much data is processed at one time.

The crux of the RCD algorithm is the actual reranking step: the winner's (uncancelled) marks must be demoted below the loser's (uncancelled) marks to make the winner more harmonic. What this step accomplishes is not the entire task of finding a correct ranking, but the incremental task of making the hierarchy consistent with some given subset of the data.

Prior to the use of the Core CD algorithm we now define, as for RCD (119), *mark-data* must be prepared as appropriate for the version of CD which will be used: the appropriate Data Preparation is indicated in the particular algorithms defined below. The core reranking is achieved as follows:

```
(121)  Core Constraint Demotion (Core CD)
        Given:
            $\mathcal{H}$ = a stratified hierarchy of constraints
            mark-data = a table of pairs of mark lists loser-marks $\prec$ winner-marks
        To Find:
            A stratified hierarchy with respect to which each of the loser-marks is less harmonic
                than its corresponding winner-marks.
    Core CD Algorithm
        I. Mark Cancellation: As for RCD (0dI)
        II. Constraint Demotion (repeat this step until no demotions occur)
            for each pair loser-marks $\prec$ winner-marks in mark-data
                i. find the mark $*\mathbb{C}_l$ in loser-marks which is highest-ranked in $\mathcal{H}$
                ii. for each $*\mathbb{C}_w$ in winner-marks
                    if $\mathbb{C}_l$ does not dominate $\mathbb{C}_w$ in $\mathcal{H}$, then demote constraint $\mathbb{C}_w$ to the
                        stratum in $\mathcal{H}$ immediately below that of $\mathbb{C}_l$ (creating such a
                        stratum if it does not already exist)
        Output $\mathcal{H}$
```

Versions of the CD Algorithm

We now present several ways of using the Core CD algorithm to learn a language. All start with an initial stratified hierarchy $\mathcal{H}_0$, e.g., the one in which all the universal constraints are in the top stratum. The constraints in this stratified hierarchy are then demoted as demanded by the data.

We start with a version of CD, Batch CD, which, we will see, is closely related to RCD: all the data is processed simultaneously by Core CD:

```
(122)  Batch CD
        a. Compile all available output forms into initial-data.
        b. Perform Data Preparation, generating mark-data.
        c. Set $\mathcal{H} = \mathcal{H}_0$
        d. Execute Core CD on mark-data and $\mathcal{H}$ (once).
```

At the other extreme, On-line CD operates by applying Core CD separately to each *sub-opt* $\prec$ *opt* pair:

(123) On-line CD

```
a. Set $\mathcal{H} = \mathcal{H}_0$
  Execute repeatedly:
      b. Select one optimal output *opt* and one suboptimal competitor *sub-opt*.
      c. Set *mark-data* = {*marks*(*sub-opt*) $\prec$ *marks*(*opt*)}.
      d. Execute Core CD on *mark-data* and current $\mathcal{H}$.
```

One way of using Core CD in between the extremes of Batch CD (all data processed at once) and On-Line CD (one mark-data pair processed at a time) is illustrated by I/O CD. One form *opt* from the language is processed at a time, but a number of competitors *sub-opt* to *opt* may be processed at once.

(124) I/O CD

```
a. Set $\mathcal{H} = \mathcal{H}_0$
  Execute repeatedly:
      b. Select one optimal output *opt*.
      c. Perform Data Preparation, generating *mark-data*.
      d. Execute Core CD on *mark-data* and current $\mathcal{H}$.
```

The Relation of RCD to Batch CD

The first iteration of Batch CD runs once through all the data, demoting a certain number of constraints. The constraints which remain in the top stratum at the end of this first iteration can never be demoted in subsequent iterations through the data; and other constraints can never be promoted into this stratum. So after the first iteration, the top stratum has its final form. All mark-data pairs containing these constraints can be removed and ignored in subsequent iterations; they will not cause further demotions. This same property then holds recursively for subsequent iterations. In general, the nth iteration of Batch CD definitively determines the nth stratum from the top. This is the same as the corresponding stratum output by RCD on the nth call to RCD (a recursive call for $n > 1$).

In other words, the real difference between Batch CD and RCD is slight: Batch CD demotes constraints to a number of levels simultaneously, never demoting any constraints in the top n strata after iteration n, while the nth call to RCD in effect just demotes constraints from the nth to the $n-1$st stratum. This is why we have used the term Recursive Constraint Domination even though, unlike the other algorithms we have discussed, RCD does not explicitly employ Core CD.

Acknowledgements

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 and two anonymous *LI* reviewers of Tesar and Smolensky 1993. 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 this earliest work described here was first presented (and reported in Tesar and Smolensky 1993; also in 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. This paper is an expanded version of Tesar and Smolensky 1996, to appear.

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. ROA-69.

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.

Benua, Laura. 1995. Identity effects in morphological truncation. In Jill Beckman, Laura Walsh Dickey, and Suzanne Urbanczyk, eds., *University of Massachusetts Occasional Papers in Linguistics* 18: Papers in Optimality Theory. Amherst, MA: Graduate Linguistic Student Association.77–136. ROA-74.

Bernhardt, Barbara H., and Joseph P. Stemberger. 1995. *Nonlinear phonology and phonological development: A constraint-based approach.* Ms., University of British Columbia, Vancouver and University of Minnesota, Minneapolis, Minn.

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.*

Buckley, Eugene. 1995. Cyclicity as correspondence. Paper presented at the University of Maryland, Nov. 10. ROA-93c.

Burzio, Luigi. 1993. English stress, vowel length, and modularity. *J. Linguistics* 29:359–418.

Burzio, Luigi. 1994. *Principles of English Stress.* Cambridge: Cambridge University Press.

Burzio, Luigi. 1995a. The rise of Optimality Theory. *Glot International* 1(6), pp. 3–7.

Burzio, Luigi. 1995b. Surface constraints vs. underlying representations. To appear in Jacques Durand and Bernard Laks, eds., *Current Trends in Phonology: Models and Methods*. CNRS, Paris X, and University of Salford Publications.

Burzio, Luigi. 1996. Multiple correspondence. Ms., Johns Hopkins. To appear in *Proceedings of the BCN Workshop on Conflicting Constraints*, Groningen.

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, George N. and Samuel Jay 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. ROA-50.

Dresher, B. Elan, and Jonathan Kaye. 1990. A computational learning model for metrical phonology. *Cognition* 34, 137-195.

Flemming, Edward and Michael Kenstowicz. 1995. Base-identity and uniform exponence: Alternatives to cyclicity. Ms., MIT.

Frank, Robert, and Shyam Kapur. 1996. On the use of triggers in parameter setting. *Linguistic Inquiry,* 27.

Gafos, Adamantios. 1996. On the notion 'paradigm structure' in Temiar. Paper presented at the GLOW '96 workshop on the morphology-phonology interface.

Gibson, Edward, and Ken Wexler. 1994. Triggers. *Linguistic Inquiry*, 25(4).

Gnanadesikan, Amalia. 1995. Markedness and faithfulness constraints in child phonology. Ms., University of Massachusetts. ROA-67.

Gold, E. M. 1978. Complexity of automatic identification from given data. *Information and Control* 37, 302–320.

Grimshaw, Jane. 1993. Minimal projection, heads, and inversion. Ms. Rutgers University, New Brunswick, NJ. (Revision: ROA-68.)

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.* Mahwah, NJ: Lawrence Erlbaum Publishers.

Itô, Junko, R. Armin Mester, and Jaye Padgett. 1995. Licensing and underspecification in Optimality Theory. *Linguistic Inquiry* 26, 571–613.

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.

Kenstowicz, Michael. 1994. Cyclic vs. non-cyclic constraint evaluation. *MIT Working Papers in Linguistics* 21. To appear in *Phonology*. ROA-31.

Kenstowicz, Michael. 1995. Base-identity and uniform exponence: Alternatives to cyclicity.Ms., MIT. ROA-105.To appear in Jacques Durand and Bernard Laks, eds., *Current Trends in Phonology: Models and Methods*. CNRS, Paris X, and University of Salford Publications. ROA-103.

Legendre, Géraldine, Yoshiro Miyata, and Paul Smolensky. 1990a. Can connectionism contribute to syntax? Harmonic Grammar, with an application. *Proceedings of the 26th Meeting of the Chicago Linguistic Society*. Chicago, IL.

Legendre, Géraldine, Yoshiro Miyata, and Paul Smolensky. 1990b. Harmonic Grammar — A formal multi-level connectionist theory of linguistic well-formedness: Theoretical foundations. *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, Cambridge, MA. July. 388–395.

Legendre, Géraldine, William Raymond, and Paul Smolensky. 1993. Analytic typology of case marking and grammatical voice. Proceedings of the *Berkeley Linguistics Society,* 19. ROA-3.

Legendre,Géraldine, Paul Smolensky, and Colin Wilson. 1995. When is less more? Faithfulness and minimal links in *wh*-chains. Ms., Johns Hopkins. ROA-117. To appear 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. 607–636. ROA-85.

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.

Lombardi, L. 1995. Positional faithfulness and the phonology of voicing in Optimality Theory. Ms., University of Maryland.

McCarthy, John and Alan Prince. 1993. *Prosodic Morphology I: constraint interaction and satisfaction*. Ms. University of Massachusetts, Amherst, and Rutgers University, New Brunswick, NJ. To appear as Linguistic Inquiry Monograph, MIT Press, Cambridge, MA.

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. ROA-60.

McCarthy, John. 1995. Extensions of faithfulness: Rotuman revisited. Ms., University of Massachusetts,. ROA-64.

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.* Mahwah, 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.

Pater, Joe, and Johanne Paradis. 1996. Truncation without templates in child phonology. In *Proceedings of the Boston University Conference on Language Development* 20, 540–552. Somerville, MA: Cascadilla Press.

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. 1996. *Constraints on Subjects: An Optimality Theoretic Analysis.* Doctoral Dissertation, Department of Linguistics, Rutgers University.

Smolensky, Paul. 1983. Schema selection and stochastic inference in modular environments. *Proceedings of the National Conference on Artificial Intelligence.* 378–82.

Smolensky, Paul. 1986. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations.* Cambridge, MA: MIT Press/Bradford Books. 194–281.

Smolensky, Paul. 1996a. Statistical perspectives on neural networks. In Paul Smolensky, Michael C. Mozer, and David E. Rumelhart, eds., *Mathematical perspectives on neural networks.* Mahwah, NJ: Lawrence Erlbaum Publishers.

Smolensky, Paul. 1996b. On the comprehension/production dilemma in child language. *Linguistic Inquiry* 27. ROA-118.

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, Department of Computer Science, University of Colorado at Boulder.

Tesar, Bruce. 1995a. Computing optimal forms in Optimality Theory: Basic syllabification. Technical Report , Department of Computer Science, University of Colorado at Boulder. ROA-52.

Tesar, Bruce. 1995b. Computational Optimality Theory. Doctoral Dissertation, Department of Computer Science, University of Colorado at Boulder. ROA-90.

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, Department of Computer Science, University of Colorado at Boulder. ROA-2.

Tesar, Bruce, and Paul Smolensky. 1995. The learnability of Optimality Theory. *Proceedings of the Thirteenth West Coast Conference on Formal Linguistics*, 122–137.

Tesar, Bruce, and Paul Smolensky. 1996. Learnability in Optimality Theory (short version). Technical Report, Department of Cognitive Science, Johns Hopkins University.

Tesar, Bruce, and Paul Smolensky. To appear. Learnability in Optimality Theory. *Linguistic Inquiry.*

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.