Using Inconsistency Detection to Overcome Structural Ambiguity in Language Learning

Bruce Tesar
Department of Linguistics
Rutgers Center for Cognitive Science
Rutgers University, New Brunswick
9/12/00

Abstract

This paper proposes the Inconsistency Detection Learner (IDL), an algorithm for language acquisition intended to address the problem of structural ambiguity. An overt, acoustically audible form is structurally ambiguous if different languages admitting the overt form would assign it different linguistic structural analyses. Because the learner has to be capable of learning any possible human language, and because the learner is dependent on overt data to determine what the target language is, the learner must be capable ultimately of inferring which analysis of an ambiguous overt form is correct by reference to other overt data of the language. IDL does this in a particularly direct way, by attempting to construct hypothesis grammars for combinations of interpretations of the overt forms, and discarding those combinations that are shown to be inconsistent. A specific implementation of IDL is given, based on Optimality Theory. Results are presented from a computational experiment in which this implementation of IDL was applied to all possible languages predicted by an Optimality theoretic system of metrical stress grammars. The experimental results show that this learning algorithm learns quite efficiently for languages from this system, completely avoiding the potential combinatoric growth in combinations of interpretations, and suggesting that this approach may play an important role in the acquisition mechanisms of human learners.

Using Inconsistency Detection to Overcome Structural Ambiguity in Language Learning

Bruce Tesar*
Department of Linguistics
Rutgers Center for Cognitive Science
Rutgers University, New Brunswick
9/12/00

1. Structural Ambiguity in Language Learning

1.1. Mutual Entanglement

A central challenge of learning natural languages is that of contending with input data that are structurally ambiguous. The portion of an utterance that is directly perceivable by the learner, labeled here the overt form, is structurally ambiguous if there is more than one complete structural description that may be assigned to it. The situation we are concerned with in this paper is that where the different structural descriptions are grammatical in different languages. Ambiguity within a language, where the same overt form can be assigned more than one analysis by a single language, is not of direct concern here.

Structural ambiguity can be illustrated with metrical stress theory. For present purposes, assume that a structural description of a word consists of the ordered sequence of syllables of the word, a grouping of syllables into feet, and an assignment of a stress level to each syllable. The overt form corresponding to a structural description is the ordered string of syllables, along with the stress levels of the syllables. An overt form is not itself a structural description; it only contains structures for elements that are presumed to be directly observable when a child hears a word uttered[1]. What is missing from the overt form is the foot structure; the child cannot directly 'hear' foot boundaries. An overt form is ambiguous when more than one structural description shares that overt form. We will refer to a full structural description consistent with an overt form as an interpretation of that overt form. An ambiguous overt form has more than one interpretation. A simple example is a three-syllable word with medial main stress: [σ σ σ] (we use σ to denote a syllable). This overt form is ambiguous between at least two interpretations[2], including:

* I would like to thank Jason Eisner, Janet Fodor, Brett Hyde, Jacques Mehler, Joe Pater, Alan Prince, Ken Safir, William Sakas, Vieri Samek-Lodovici, Paul Smolensky, the students of the Spring 2000 Rutgers University Learnability and Linguistic Theory seminar, and the audiences at HOT'97, NELS 28, The CUNY Graduate Center, Carnegie Mellon University, the NELS 30 Workshop on Language Learnability, NYU, MIT, Rochester University, Western Michigan University, and SUNY Stony Brook, for useful comments. Alan Prince also provided many useful comments on an earlier draft of this paper. Part of this research was funded by postdoctoral support from the Department of Linguistics, Rutgers University, and the Rutgers Center for Cognitive Science.

[1] The stress levels in the overt form are a direct translation of the relative prominence of the syllables as expressed in acoustic, observable properties: duration, pitch, and amplitude. The syllable structure itself is, of course, constructed by the learner, based upon the acoustic signal. Syllable structure construction will simply be assumed for the discussion of stress in this paper, but in general elements of syllable structure can also be subject to cross-linguistic structural ambiguity.

[2] Another possible interpretation is one with the stressed syllable as a foot by itself, [σ (σ) σ]. Such a foot is not unexpected in trochaic, quantity-sensitive languages, when the syllable is heavy.

- An iambic analysis, grouping the first two syllables to form an iambic foot: [(σ σ) σ]
- A trochaic analysis, grouping the final two syllables to form a trochaic foot: [σ (σ σ)]

This ambiguity would not necessarily seem consequential if one of the interpretations wasn't really a possible form in a human language; a learner armed with such knowledge could detect the universally ill-formed interpretation and discard it. But in many instances, including the one just given, more than one of the possible interpretations is a possible interpretation of a human language. The iambic interpretation is that employed by Araucanian: *tipa nto* 'year' is analyzed as [(*ti pa n*) *to*] (Echeverria & Contreras, 1965). The trochaic interpretation is that employed by Warao: *kora nu* 'drink it!' is analyzed as [*ko* (*ra nu*)] (Osborn, 1966). The different interpretations assigned by the same language are not motivated by elements intrinsic to that particular trisyllabic form, but by the stress patterns of other forms of the respective languages (some examples are shown in Table 1).

Table 1

Stress patterns for Araucanian and Warao.

Araucanian		**Warao**	
[(*wule*)]	'tomorrow'	[(*ti ra*)]	'woman'
[(*tipa n*)*to*]	'year'	[*ko*(*ra nu*)]	'drink it!'
[(*elu*)(*muyu*)]	'give us'	[(*ru hu*)(*na e*)]	'he sat down'
[(*elu*)(*ae*)*ne w*]	'he will give me'	[*ni*(*ha ra*)(*pa k a*)]	'alligator'

The learner cannot determine the correct interpretation of the overt form based upon that overt form alone. This matters to the learner, because the foot structure necessarily mediates between the overt data and the central principles of stress grammars. Interpretations are full structural descriptions. The principles that can vary among the possible human grammars evaluate full structural descriptions, not overt forms, and they make crucial reference to hidden structure such as feet. Dresher refers to this gap between overt forms and the full structural descriptions evaluated by grammars as the epistemological problem (Dresher, 1999). The learner will draw very different conclusions about the grammar it is attempting to learn if it assumes the iambic interpretation than if it assumes the trochaic interpretation.

Note than competent adult speakers of one of the above languages have no problem disambiguating the stress pattern of the overt form, because they possess the correct grammar for their language. Native speakers of Warao possess a grammar instructing them to group the 2nd and 3rd syllables of the trisyllabic word together into a foot. But this correct grammar is precisely what the language learner lacks, at least at the outset. The learner is provided only with overt forms. Neither the complete grammar nor the full structural analyses of the forms are directly apparent to the learner; they must be inferred. However, they cannot be inferred independently, because each has strong implications for the other: a grammar assigning a trochaic foot at the right edge goes hand in hand with an interpretation like [σ (σ σ)]. We will refer to this as the mutual entanglement of grammar and analysis.

As suggested above, it is safely assumed that, given the correct grammar, the correct interpretation for an overt form can be efficiently determined[3]. Although it may not be immediately apparent, it will be demonstrated in section 2.2 below that, given the correct full interpretations of the overt forms for a language, the correct grammar can be determined fairly easily. Learning appears challenging because the

[3] The assumption is quite safe for metrical stress, as discussed here. In other domains, there do exist grammatical forms for which interpretation is known to be difficult, e.g., garden-path sentences in syntax. For present purposes, the applicability of the discussion to areas like syntax is preserved by assuming that language learners needn't crucially depend upon correctly analyzing garden-path type forms.

learner starts out knowing neither the grammar nor the interpretations, and mutual entanglement suggests that they must somehow be learned together.

At this point, it is worth pointing out that the mutual entanglement problem is not parochial to metrical stress, or even to phonology, but is in fact endemic to language learning. Pinker (Pinker, 1987) referred to this problem as the bootstrapping problem, and specifically discussed it in the context of syntax acquisition. The mutual entanglement there is between the interpretation (syntactic analysis) of sentences and the syntax grammar of a language: the hidden structure of the interpretations includes the syntactic categories of the words in the sentence and their roles in argument structure. In fact, in syntax, problems of this sort remain even if you assume the learner has access to the syntactic categories and argument structure roles of the words, because frequently more than one structural analysis (tree structure) can be assigned. Structural ambiguity can arise any place where raw overt data can receive more than one interpretation in terms of linguistic representation, and mutual entanglement follows when different possible grammars assign the different interpretations.

1.2. Previous Proposals for Overcoming Mutual Entanglements

This subsection reviews a combination of prior work in language learning. Some are prior approaches to learning metrical stress, while others are prior approaches to contending with mutual entanglement (not necessarily in metrical stress, or even phonology). The focus is on how the different proposals deal (or do not deal) with structural ambiguity and mutual entanglement. Section 1.2.6 briefly outlines how the new proposal of this paper, inconsistency detection, relates to the prior work with regard to mutual entanglement.

1.2.1. Avoiding (explicit) linguistic abstractions

One approach to learning language phenomena starts out by denying, in a sense, the linguistic nature of the phenomena. This approach commonly attempts to utilize quite general pattern recognition techniques for mapping and reproducing the overt forms of the language, without particular attention to what kinds of abstractions the model develops as it is trained. Thus, the question of structural ambiguity isn't so much answered as ignored; the question of what structural analysis is assigned to an overt form is intentionally left unasked. Examples of techniques applied in this way to metrical stress are the nearest neighbor method combined with a statistically trainable distance metric (Daelemans, Gillis, & Durieux, 1994), perceptron networks (Gupta & Touretzky, 1994), and recurrent backpropagation networks (Joanisse & Curtin, 1999).

The studies of Joanisse & Curtin, and of Daelemans, Gillis & Durieux evaluated their models on only a single language (Dutch); the latter study restricted the input to only the final three syllables of words, preempting any thought of trying it on, e.g., languages with initial main stress. Those studies focused on the problem of reproducing both regular and irregular forms within a single language, rather than on a learner equally capable of learning the predominant pattern for any of a number of languages. Perhaps it is not too surprising that possible differences in analysis across languages do not appear prominently in studies focusing on only a single language.

Gupta & Touretzky trained their perceptron on 19 attested stress patterns, and examined differences in the time required to learn each (including the two languages the model failed to learn). They did analyze the trained weights of the perceptron for the different languages, but they focused on weight patterns that correlated with descriptive properties of the stress patterns (such as "inconsistent primary stress", "stress clash avoidance"), and were primarily interested in explaining differences in learning times, including failure to learn at all, exhibited by the perceptron, and not in explaining how humans succeed in learning stress patterns. In particular, they did not discuss how two different languages that overlap on a particular stress form might differ in analysis.

It should be emphasized that just because the notion of 'structural description' is not commonly discussed with respect to a certain type of model does not automatically mean that the concept does not

apply. For example, networks such as the backpropagation network used by Joanisse & Curtin have hidden units, and the activation values of those units for a particular input form constitute an analysis, or 'description', of that input. Any model which generalizes beyond a memorized list of input/output pairs has an abstraction mechanism of some kind, and therein lie its ways of interpreting particular input properties for the purposes of assigning an output. Further, one could try to analyze and understand the abstractions used by such models, even in linguistic terms. While the analysis of hidden unit activations in connectionist networks is notoriously difficult, it is not in principle impossible, as a number of specifically constructed example networks demonstrate.

What is missing from all of the studies just cited is any exploration of the range of possible stress systems predicted by the models; Gupta & Touretzky are quite explicit in their lack of regard for such typological issues. These models lack the capacity to explain the central properties of human stress systems, for example their fundamentally rhythmic quality (Liberman & Prince, 1977). The kinds of structural descriptions (and related theoretical principles) of concern in this paper are usually motivated by an interest in capturing what different stress systems have in common, as well as capturing the specific ways in which they differ.

1.2.2. Avoiding assignment and use of structural descriptions during learning

The cue learning approach has been applied to principles and parameters theories for metrical stress (Dresher, 1999; Dresher & Kaye, 1990). In this approach, grammars assign structural descriptions to overt forms, and different grammars can assign different interpretations to the same overt form. However, the problem of structural ambiguity in learning is dealt with by completely avoiding the assignment and use of structural descriptions by the learner while learning. Learners instead invoke mechanisms for evaluating overt forms that are specific to learning and separate from the standard parsing of overt forms involved in language use. These learning-specific mechanisms search for cues, which are specified patterns in overt forms. Each parameter has at least one cue associated with it, and if the cue is observed in some sufficient quantity, the corresponding parameter is set to a specific value. Typically, the nature of the cues is such that the parameters must be set in a specific order. The learner cannot set a particular parameter until it has set the other parameters ordered before it (the precise form of a cue may depend upon the values of earlier parameters). This learning approach is decidedly not error-driven[4]; throughout learning, the learner remains oblivious to whether then can actually parse the overt forms they are seeing. Indeed, depending upon how one interprets the not-yet-set parameters, the learner may not have a properly determined, usable grammar until the very end of learning.

The definitions of some cues involve the construction of what might be called "partial descriptions", structures which organize certain aspects of the overt form being checked. These constructions can sometimes look suspiciously similar to components of the structural descriptions assigned by grammars; for example, the cue for the main stress parameter is "Scan a constituent-sized window at the edge of a word. Main stress should consistently appear in either the left or right window." (emphasis added - BT) (Dresher, 1999). However, they are necessarily different from full structural descriptions, as the order and design of these analytic structures are designed to avoid the structural ambiguity of the full descriptions actually licensed by the grammars (and the linguistic theory). In other words, although the regular structural descriptions are posited by linguistic theory precisely because they are so effective at mediating between overt forms and central linguistic principles, the learner uses an entirely separate class of analytic devices for mediating between overt forms and linguistic principles for the purposes of learning. This can be a cause of dissatisfaction, especially when the complexity of the learning-specific system can rival (or even exceed) that of the linguistic theory itself. See (Bertolo, Broihier, Gibson, & Wexler, 1997a) for further discussion and results on cue learners.

[4] For a description of error-driven learning, see section 1.2.3.

1.2.3. Enumerative search and random search

One of the oldest proposals for learning grammars is simple enumerative learning (Gold, 1967). An ordered list of the possible grammars is given, and the learner performs what has come to be known as error-driven learning (Wexler & Culicover, 1980). An error occurs when the learner encounters an overt form that cannot be successfully parsed (that is, assigned a grammatical structural description) by the learner's current grammar. The learner starts with the first grammar on the list; anytime an error occurs falsifying the learner's current grammar, the learner proceeds to the next grammar in the list. Not surprisingly, this approach incurs extremely long convergence times for large grammar spaces; if your target language matches the first grammar on the list, there you are, but if it matches grammar number ten million, learning will take a while. This kind of exhaustive search is generally regarded as hopelessly inefficient for realistic grammar spaces.

An alternative to strictly ordered deterministic search is random search, the hope being to be able to learn any of the possible languages without having to exhaustively check every single grammar. One such algorithm is the Triggering Learning Algorithm (TLA) (Gibson & Wexler, 1994). Grounded in the principles and parameters framework, the TLA is error-driven, responding to an error on an overt form by changing the value of a single parameter at random. If the parameter change results in a grammar that can parse the overt form without error, then the change is kept. Unfortunately, in general the randomness of the search and the constraining topology of the parametric theory do little to prevent learning times from rivaling those of exhaustive search. In fact, the use of parametric theory to constrain the direction of search can be a significant hindrance, making learning impossible in some cases (Niyogi & Berwick, 1996). For further discussion of the limitations of the TLA, see (Fodor, 1998).

A more complex form of random search is genetic algorithms. Such algorithms have been applied to many types of problems, including grammar learning (Clark, 1992; Clark & Roberts, 1993; Pulleyblank & Turkel, 1998). This type of learner maintains a population of several grammars, evaluating each on a collection of data. The grammars are then changed randomly, both via individual mutation and by combining elements of different grammars together, with preference given to grammars which have higher evaluations on the data. These algorithms also have high costs in terms of learning time and required computational effort.

What all of these approaches have in common is that they use structural analyses of overt forms, but only for the purpose of determining if an overt form is possible for a particular grammar. When a grammar fails to successfully parse an overt form (when an error occurs), nothing about the overt form is used to determine how the grammar might be changed to achieve success; other extrinsic devices (ordered grammar lists, random grammar changes) are used instead. The overt forms only figure in the evaluation of the grammars. The analyses of overt forms are relevant only in verifying that a grammar is capable of parsing those overt forms. Ambiguity of interpretation for overt forms is certainly possible (and indeed common) in the linguistic systems employed by these learning algorithms, and it can cause difficulty for these learners (an incorrect grammar may appear to be correct because it successfully parses an overt form, but by assigning it the wrong interpretation). But, the learners themselves are relatively oblivious to structural ambiguity; either a grammar can parse an overt form (by some or other structural interpretation) or it cannot. The price is paid in terms of learning times.

1.2.4. Insisting on unambiguous forms

A more explicit confrontation of ambiguity appears in the Structural Triggers Learner (STL) (Fodor, 1998; Sakas & Fodor, in press). A trigger is an overt form that indicates to the learner the value of a particular parameter. An overt form that is structurally ambiguous is also likely to be ambiguous with respect to the settings of some parameters: one interpretation might require one setting of a parameter, while another interpretation might require a different setting of the same parameter. The proposal is for the learner to only learn from unambiguous forms, that is, to require triggers to be unambiguous. If the

learner receives an unambiguous trigger, the learner can set those parameters that the trigger requires to have specific values.

The term "unambiguous" has a different meaning for the STL than it does for much of the rest of this paper. The STL is more immediately concerned with whether an overt form is consistent with more than one setting of a parameter (parametric ambiguity) than it is with whether an overt form has more than one structural interpretation (structural ambiguity). In practice, the two types of ambiguity frequently go together. The STL interprets an overt form based upon whatever parameters have already been set (initially, none). For the learner to get started, it must encounter an overt form which acts as a trigger for at least one parameter. The learner can then make use of those parameter settings, which may serve to disambiguate another overt form: the overt form may have several interpretations, but if all but one of them contradict at least one of the learner's set parameters, the learner can presume the consistent interpretation. Thus, for the STL, a trigger can be "unambiguous" relative to what other prior parameters have been set. The learner can succeed if it can find a critical sequence of triggers, each trigger rendering the next unambiguous, that ultimately sets all of the parameters.

For the STL to work, it must be capable of parsing overt forms while having only some (or none) of its parameters set. This turns out to be a quite nontrivial matter. The cue learning approach dealt with this by not even attempting to parse, instead having specific cue-searching mechanisms for each parameter defined so as to not depend upon unset parameters. For the STL, the proposal is a mechanism known as a superparser. It is not at all clear what a superparser would be like in a more traditional principles and parameters system, such as that used by the Triggering Learning Algorithm, and by the genetic algorithm studies of Clark & Roberts. Fodor & Sakas instead cast the TLA within a framework based upon the minimalist program (Chomsky, 1995); in their view, a parameter is a small piece of syntactic structure, a tree fragment, and learning a grammar means identifying which of the possible parameters are in use by the target language. Constructing such a superparser is a challenge, and is the topic of current research (Fodor, to appear).

Apart from the challenges of superparsing, another potential problem is the availability of unambiguous triggers; the learner could end up waiting a long time, allowing lots of potentially informative input data to go by, waiting for an unambiguous trigger to appear (Bertolo, Broihier, Gibson, & Wexler, 1997b). A related problem is the requirements made of the linguistic system. The STL learner might get stuck forever if it reaches a point where each overt form relevant to an unset parameter is ambiguous for that parameter, even if a small set of overt forms would be collectively unambiguous. In the metrical stress system used in the experiment described in section 4, every overt form has at least two candidate interpretations; any learner that insisted upon starting with a form admitting only one possible interpretation would never even get started.

The STL responds to an overt form by attempting to assign it structural interpretations, using whatever parameter settings it already has. This is done to determine if the learner yet knows enough to adequately limit the number of interpretations it would need to consider. If only a single interpretation remains, or perhaps if the remaining interpretations all agree in their required setting of a not-yet-set parameter, then that interpretation is used to learn more about the target grammar. If not, the STL responds to the ambiguity by refusing to learn from the form. Thus, the STL makes more use of the structural interpretations in directing learning than the previous methods discussed, but places strong restrictions upon the structural interpretations it can actually use.

1.2.5. Iterative refinement of analysis and grammar

The Robust Interpretive Parsing / Constraint Demotion algorithm (RIP/CD) (Tesar, 1998b; Tesar & Smolensky, 2000) is cast within the framework of Optimality Theory (OT), and takes advantage of the optimizing nature of OT. The 'RIP' part uses the learner's working constraint hierarchy to interpret an overt form. If the form has more than one interpretation, the learner selects the interpretation that fares best on the learner's current constraint hierarchy (see section 3.2.1 for a more detailed illustration of RIP).

In this way, the learner can use their current grammar to assign an interpretation to an overt form even when that overt form does not correspond to any description which is grammatical according that same grammar. This is the 'robust' part of RIP; the learner always makes their best effort to make sense out of what they hear. This still permits error-driven learning, however, because the learner can then take the linguistic input portion of their interpretation, and see how their current grammar would have analyzed it. If their current grammar's analysis matches their interpretation of the overt form, that is, if the learner would say it the way they heard it, then no error has occurred for learning purposes. If the two do not match, that is, if the learner would say it differently from how they heard it, then an error has occurred, and learning should take place. For OT, the "mismatch" takes the place of "failing to parse" in defining an error.

It is after an error has been detected, and the learner needs to change their grammar, that the interpretation produced by RIP becomes most useful. The learner detected the error by comparing their interpretation with their grammar's current grammatical description. By comparing these two, the learner can derive some quite specific information about how to change their grammar to one closer to making the interpretation of the overt form grammatical. The algorithm for performing this grammar change is Constraint Demotion[5], the 'CD' part of RIP/CD. Having now modified their grammar, the learner can repeat the process, using the new grammar to robustly parse the overt form and check for an error. The idea is for the learner to iterate back and forth between the two stages, parsing (RIP) and grammar learning (CD), until a convergent grammar is arrived at in which no error is detected.

If there were no structural ambiguity in overt forms, this algorithm would be guaranteed to succeed, and to do so quickly. The challenge to this learner posed by structural ambiguity is the possibility of having RIP assign the wrong interpretation to an overt form. This will lead CD to attempt a different grammar than the target grammar. The hope is that, because the interpretation arrived at by RIP is constrained to match the overt form, that even if RIP assigns the wrong description, the learner is still gaining some useful information, and that over time the processing of many forms will cause it to end up with the correct grammar. This hope is inspired by convergence results for a class of statistical learning algorithms that have a similar iterative, back-and-forth quality, the Expectation Maximization algorithms, or EM (Dempster, Laird, & Rubin, 1977).

RIP/CD has been tested experimentally on some OT systems for metrical stress (Tesar, 1998b; Tesar & Smolensky, 2000). When it succeeds, it is very fast, arriving at a correct grammar in very few steps. However, success is not guaranteed, and some conditions have been determined that can cause the algorithm to get stuck, doomed to never converge on a correct grammar. These conditions pose a threat to the viability of RIP/CD as a model of human language acquisition, absent some method of avoidance of or recovery from these conditions.

RIP/CD contrasts with the STL in its response to a structurally ambiguous overt form. While the STL refuses to learn until it can confidently take a guaranteed step, RIP/CD always presses ahead using its best guess at the correct interpretation. RIP/CD differs from algorithms like the TLA in that its best guess is not random, but guided and constrained by the overt form, because RIP/CD has a structural interpretation of the overt form to work with.

[5] This Constraint Demotion (CD) algorithm is distinct from, although fundamentally related to, the Recursive Constraint Demotion (RCD) algorithm presented in section 2.2.2. CD takes an existing constraint hierarchy and modifies it in response to a single piece of information; it can be invoked repeatedly on different pieces of information to perform a sequence of changes to the hierarchy. In contrast, RCD takes a complete set of information at once, and constructs an entire hierarchy from scratch based upon all of the information in the set. See (Tesar & Smolensky, 2000) for further discussion of the relationship between the two algorithms.

1.2.6. A New Approach: Inconsistency Detection

What is desirable is a learning algorithm that can use the observed overt forms to actively guide the learner toward the correct hypothesis, does not need to wait indefinitely for strictly unambiguous forms, but can proceed in a way that avoids getting stuck. RIP/CD gets stuck in part because it deals with structurally ambiguous overt forms by always limiting itself to one of the possible interpretations. The alternative proposed in this paper is that the learner can benefit greatly from actively considering more than one possible interpretation, using other overt forms of the language to determine for sure which interpretation is the correct one. The claim is that the learner can do this in a way which is guaranteed to succeed, but is nevertheless computationally efficient. The key to the approach is inconsistency detection.

1.3. Detecting Inconsistencies Between Analyses

1.3.1. The Concept of (In)consistency

If a single overt form is structurally ambiguous, that ambiguity can only be resolved by reference to other forms in the language. This is what linguists do when analyzing a language; they draw conclusions about things like foot structure based upon a variety of forms in the language. This paper proposes a learning algorithm which brings other overt forms to bear on an ambiguous form in a rather direct way, by looking for interpretations of overt forms that are mutually consistent.

Here is an illustration of the concept of consistency. As discussed above, the overt form [σ σ σ] is ambiguous. One interpretation of this form is [σ (σ σ)]. This interpretation is consistent with some interpretation of another overt form if there exists a possible grammar in which both interpretations are grammatical. The interpretations [σ (σ σ)] and [(σ σ) (σ σ)] are consistent with each other, because there exists a grammar (several, in fact) that admits both as grammatical. An example would be a grammar that iterates trochaic feet from right to left and places main stress on the leftmost stressed syllable. The interpretations [σ (σ σ)] and [σ σ (σ σ)] are also consistent with each other, because there exists a grammar that places a single trochaic foot at the right edge of the word (with no secondary stresses). However, [σ (σ σ)] is not consistent with [(σ σ) σ σ], if we accept standard assumptions; no grammars exist which assign right-aligned trochaic feet to three-syllable words and left-aligned iambic feet to four-syllable words. Given two overt forms from the target language, and given an interpretation of each of those overt forms, if the two interpretations are inconsistent, then the learner may conclude that at least one of the interpretations is not the correct analysis of its corresponding overt form. The overt form of [(σ σ) σ σ], which is [σ σ σ σ], can completely rule out the interpretation [σ (σ σ)] if the learner goes on to notice that every other interpretation of [σ σ σ σ], namely [σ (σ σ) σ] and [σ (σ) σ σ], is also inconsistent with [σ (σ σ)]. By eliminating [σ (σ σ)] and [σ (σ) σ] in this way, the learner could use the observation of overt form [σ σ σ σ] to effectively disambiguate the overt form [σ σ σ] in favor of the interpretation [(σ σ) σ].

A few points are worth emphasizing right away. First, the nonexistence of a grammar can only directly determine the incompatibility of two interpretations, that is, two full structural descriptions. Determining that one interpretation of an overt form, like [σ (σ σ)] above, is inconsistent with a different overt form, like [σ σ σ σ] above, requires showing that the former interpretation is inconsistent with every possible interpretation of the latter overt form. Second, (in)consistency is a property not just of pairs of structural descriptions, but of sets of arbitrary size. Even a single structural description is effectively inconsistent if it is not admitted as grammatical by any possible grammar. It is also possible to have two structural descriptions which are consistent as a pair, but when combined with another structural description form an inconsistent triple. Third, this conception of consistency requires a strong theory of what the possible human linguistic grammars are.

1.3.2. Using Inconsistency Detection in Learning

Given a set of overt forms from the target language, the learner could use inconsistency detection to rule out combinations of interpretations of the overt forms. Given enough data, the learner should in principle be able to eliminate all but one interpretation of each overt form. The grammar admitting each of the (non-eliminated) interpretations as grammatical should be the correct grammar. That is the inspiration behind the learning proposal of this paper, the Inconsistency Detection Learner (IDL).

Forging that inspiration into a viable learning theory requires several things. It requires a strong linguistic theory, making specific claims about the possible grammars, and what structural descriptions are grammatical for each grammar. It requires some kind of efficient procedure for determining when a set of structural descriptions is mutually inconsistent, as well as an efficient procedure for determining a grammar corresponding to a consistent set of structural descriptions. If determining whether a set of descriptions is consistent required separately examining the extensional consequences of every possible grammar, then this approach would be woefully inadequate, given the size of realistic grammar spaces, in the same way as the enumerative and random search approaches discussed earlier. Fortunately, it is in fact possible to determine the (in)consistency of a set of descriptions far more efficiently, given the proper linguistic theory; this will be proven below.

The details of IDL as presented here are based upon a particular approach to linguistic theory, Optimality Theory (Prince & Smolensky, 1993). An overview of Optimality Theory and related learning principles is presented in section 2. With that background established, the learning algorithm is presented in detail in section 3. The results of a computational experiment, designed to evaluate the computational efficiency of the algorithm, are presented in section 4.

2. Learning in Optimality Theory

2.1. Optimality Theory

In Optimality Theory (Prince & Smolensky, 1993), universal grammar consists of a universal candidate mapping, GEN, and a set of universal constraints, CON. GEN maps each linguistic input to an assigned set of candidate structural descriptions. The candidates for an input are understood to compete with each other. The competition is based on the constraints of CON, which evaluate structural descriptions. The constraints are universal: the same constraints are present in all languages. However, each language ranks the constraints in a particular order of importance. Languages can differ in the order in which they rank the constraints, and this is the mechanism by which Optimality Theory (henceforth OT) accounts for cross-linguistic variation. The interpretation of the ranking of the constraints is that of strict domination: if constraint C1 dominates constraint C2, a relationship denoted C1 $\gg$ C2, then C1 gets strict priority, and the grammar would prefer to tolerate an arbitrary number of violations of C2 if doing so permits even one less violation of C1. The evaluation of a candidate structural description by the ranked constraints is referred to as that candidate's harmony. If candidate D1 fares better on the ranked constraints than candidate D2, then D1 is more harmonic than D2, denoted D1 $\succ$ D2. For a given linguistic input, the constraints of CON evaluate all of the candidate structural descriptions assigned to the input by GEN. The most harmonic candidate (the one which best satisfies the ranked constraints) is the grammatical structural description of that input for that ranking; it is the optimal candidate.

The domain of GEN implicitly defines the universal set of possible linguistic inputs. For our stress system, the set of possible inputs consists of all strings of heavy and light syllables. Each linguistic input to this system is presumed to be the syllables of a single prosodic word. The candidate structural descriptions assigned to an input by GEN are the possibilities that a grammar may choose among when selecting the grammatical description of an input. If a hypothetical structure is not one of the candidates defined by GEN for an input, then it is not a possible structural description for that input under any grammar of the system. For our stress system, the candidate structural descriptions of an input are

assignments of foot structure and stress levels to the syllables. This candidate set is subject to several conditions:

- A foot consists of either one or two contiguous syllables.
- Feet may not overlap.
- Each foot has precisely one head syllable.
- A head syllable bears stress (it is the most prominent syllable of the foot).
- Syllables which are not foot heads are unstressed.
- Each candidate must have at least one foot, which is the head of the prosodic word.
- The head syllable of the head foot bears main stress.
- The head syllables of other feet (not the head foot) bear secondary stress.
- Apart from the requirement that a prosodic word have a head foot, syllables may be unfooted.

For our purposes, syllables will be distinguished only by their weight: as light, denoted '◡', or heavy, denoted '—'. A word consisting of a light syllable followed by a heavy followed by two more light syllables would be denoted [◡ — ◡ ◡]. When syllable weight is not relevant, all syllables will frequently be denoted 'σ' to simplify.

Table 2 shows several (but not all) of the candidate structural descriptions for the input / ◡ — ◡ ◡ /.

Table 2

Some of the candidate structural descriptions for linguistic input / ◡ — ◡ ◡ /.

[(◡ —) ◡ ◡]	[(◡ —) (◡ ◡)]	[(◡ —) (◡ ◡)]	[◡ — (◡ ◡)]
[◡ (— ◡) ◡]	[(◡) (— ◡) (◡)]	[(◡) (—) ◡ (◡)]	[(◡ —) (◡) ◡]
[(◡) (— ◡) ◡]	[(◡ —) (◡ ◡)]	[◡ — ◡ (◡)]	[◡ — (◡ ◡)]

The constraints of CON evaluate the candidate structural descriptions in terms of violations. Each candidate violates a constraint zero or more times. Violation is always negative in evaluation: more violations of a constraint are worse than fewer violations, and no violations is the best that a candidate can possibly do with regard to a particular constraint. Our stress system has 10 constraints, shown in Table 3.

Table 3

The constraints of the stress system.

Constraint	**Description**
PARSE	a syllable must be footed
MAIN-RIGHT	align the head foot with the right edge of the word
MAIN-LEFT	align the head foot with the left edge of the word
ALL-FEET-RIGHT	align each foot with the right edge of the word
ALL-FEET-LEFT	align each foot with the left edge of the word
FOOT-NONFINAL	a head (stressed) syllable must not be final in its foot
IAMBIC	the final syllable of a foot must be the head (stressed)
FOOTBIN	a foot cannot consist of a single light syllable
NONFINAL	the final syllable of a word must not be footed
WSP	a heavy syllable must be stressed

The constraint violations assessed to candidates by the constraints are often represented in a tableau, such as is shown in Table 4; the constraint names are abbreviated to fit. Each asterisk '*' represents a single violation. The constraints MAIN-RIGHT, MAIN-LEFT, ALL-FEET-RIGHT, and ALL-FEET-LEFT are gradiently violable alignment constraints, meaning that the number of violations is sensitive to the extent to which the two relevant structures are misaligned. For example, the constraint MAIN-RIGHT is violated

once for every syllable that intrudes between the head foot (the foot containing main stress) and the right edge of the word. The constraint ALL-FEET-LEFT is violated once for each syllable intruding between a foot and the left edge of the word, and is evaluated separately for each foot of a description.

Table 4

A constraint tableau with some stress candidates for input /⏑—⏑⏑/.

	/⏑—⏑⏑/	PARSE	M-R	M-L	A-F-R	A-F-L	F-NF	IAMB	FTBIN	NF	WSP
(a)	[(⏑ —			**	**	**		**		*	*
(b)	[(⏑ —		**		**	**		**		*	*
(c)	[(⏑ —)⏑⏑]	**	**		**			*			*
(d)	[(⏑)(—		***		*****	***	**	*	*	*	
(e)	[(⏑—)⏑⏑]	**	**		**		*				
(f)	[(⏑—			**	**	**	**			*	
(g)	[⏑(—)⏑⏑]	***	**	*	**	*	*				
(h)	[⏑—⏑(⏑)]	***		***		***	*		*	*	*
(i)	[⏑—(⏑⏑)]	**		**		**	*			*	*

Observe that every candidate in the tableau has several constraint violations; this is normal for OT analyses. Which candidate is grammatical depends critically upon the constraint ranking used to define a particular grammar. Suppose that the constraints were ranked in the order that they appear in Table 4, from left to right, with PARSE being the highest-ranked constraint. This would be the ranking

PARSE ≫ M-R ≫ M-L ≫ A-F-R ≫ A-F-L ≫ F-NF ≫ IAMB ≫ FTBIN ≫ NF ≫ WSP

Of the candidates shown in Table 4, the one that is optimal on this ranking is candidate (a). The top-ranked constraint, PARSE, eliminates candidates (c), (e), (g), (h), and (i) right away. The competition then proceeds to the next constraint, M-R, which eliminates candidates (b) and (d). After the first two constraints, only candidates (a) and (f) remain viable. Those two candidates have equal violation of the next three constraints, and it is F-NF that finally eliminates (f) in favor of (a).

Not all candidates are possible grammatical forms. For example, candidate (h) of Table 4 is not optimal under any of the possible rankings of the ten constraints. To see why, compare it to candidate (i) below it. On every constraint, (h) has as many or more violations than (i). This is an example of harmonic bounding; no matter what the constraint ranking, the highest-ranked constraint on which (h) and (i) differ will eliminate (h) in deference to (i). This does not necessarily mean that (i) will be optimal; for the ranking used in Table 4, (i) loses to (a). But anytime (i) is not optimal, (h) will not be optimal either. Candidate (i) is a possible grammatical form, because it is optimal under a different ranking, one in which ALL-FEET-RIGHT and IAMBIC dominate all of the other constraints. Candidate (h) is not a possible grammatical form, for there is no ranking under which it is optimal. This does not make (h) an illegal candidate in any sense. GEN defines a very general space of candidates; it is the job of the constraints to determine which ones are possibly grammatical.

This system for metrical stress incorporates theoretical ideas from a number of sources, including (Hammond, 1990; Hayes, 1995; Kager, 1994; McCarthy & Prince, 1993; Prince, 1990; Prince & Smolensky, 1993). It is also less directly but no less significantly influenced by other work on metrical stress; to name a few, (Halle & Vergnaud, 1987; Hayes, 1980; Liberman & Prince, 1977; Prince, 1983).

2.2. Learning Constraint Rankings

Given that GEN and CON are universal, and what varies cross-linguistically is the ranking of the constraints, learning the core grammar of a language means learning the language-specific ranking. We

can assume that a language learner starts out with knowledge of GEN and CON. We now provide one of the necessary elements of IDL (as stated in section 1.3.2): an efficient procedure for determining a grammar corresponding to a consistent set of structural descriptions. This problem has been studied in detail, and strong algorithmic solutions have been developed (Tesar, 1995; Tesar & Smolensky, 1998).

2.2.1. Mark-Data Pairs

The first important observation is that a correct target candidate (one that is known to be grammatical in the target language) has implications for a constraint ranking not in isolation, but when compared to a competing candidate. The pairing of a target candidate with a competing candidate is known as a mark-data pair, with the target candidate referred to as the winner and the competitor referred to as the loser. Given such a pair, one can determine what needs to be true of a ranking such that the winner will be more harmonic than the loser.

Consider the pair of candidates shown in Table 5. The dashed lines separating the columns for the constraints indicate that no ranking relationships are yet known among the constraints (the rank ordering is what needs to be learned).

Table 5

Candidates of a mark-data pair

	/◡—◡◡/	PARSE	M-R	M-L	A-F-R	A-F-L	F-NF	IAMB	FTBIN	NF	WSP
winner	[(◡ —			**	**	**		**		*	*
loser	[(◡—)◡◡]	**	**		**		*				

The winner is the candidate that should win under the target ranking; in particular, it should beat the loser. This means that at least one constraint violated more by the loser, that is, a constraint which prefers the winner, should dominate all of the constraints violated more by the winner (constraints which prefer the loser). This relationship among constraints can be seen a bit more clearly if we represent this mark-data pair with a comparative tableau, as proposed by Prince (Prince, 1999). Table 6 presents the mark-data pair of Table 5 in this way.

Table 6

Comparative Tableau for a mark-data pair

Loser	**Winner**	PARSE	M-R	M-L	A-F-R	A-F-L	F-NF	IAMB	FTBIN	NF	WSP
[(◡—	[(◡ —	W	W	L		L	W	L		L	L

In a comparative tableau, a single row corresponds to a mark-data pair, and each constraint is marked in its column to indicate which candidate it prefers. A 'W' in the column for the PARSE constraint indicates that PARSE prefers the winner over the loser, meaning that the loser violates PARSE more times than the winner does. Given the strict domination nature of OT, how many more times the loser violates PARSE is not important, and so is not represented. The 'L' in the column for MAIN-LEFT (M-L) indicates that MAIN-LEFT prefers the loser. The blank in the column for ALL-FEET-RIGHT (A-F-R) indicates that it has no preference between the two candidates. Notice that this does not necessarily mean that both candidates satisfy the constraint (in this instance, both candidates violate ALL-FEET-RIGHT twice), only that both violate the constraint an equal number of times.

What this mark-data pair requires, then, is that at least one of the constraints with a 'W' in its column must dominate all of the constraints with an 'L' in their column:

(PARSE **or** M-R **or** F-NF) ≫ (M-L **and** A-F-L **and** IAMB **and** NF **and** WSP)

Notice that the 'W', or winner-preferring, constraints are disjoined, while the 'L' constraints are conjoined. That is the fundamental logic of strict domination. This concisely expresses the information about the target constraint ranking that is contained in the mark-data pair.

2.2.2. Recursive Constraint Demotion

The mark-data pair is the fundamental unit of information about a constraint ranking. An appropriate set of mark-data pairs should collectively express all that there is to say about the constraint ranking defining a particular target language. How can a learner efficiently combine the information contained in a set of mark-data pairs? The answer is an algorithm known as Recursive Constraint Demotion (Tesar & Smolensky, 1994, 2000). Recursive Constraint Demotion (RCD) can be illustrated using the set of mark-data pairs in Table 7.

Table 7

Initial input mark-data pairs for RCD

Loser	**Winner**	PARSE	M-R	M-L	A-F-R	A-F-L	F-NF	IAMB	FTBIN	NF	WSP
[(⌣⌣)]	[(⌣ ⌣)]						W	L			
[(⌣ ⌣)⌣]	[⌣(⌣ ⌣)]		W	L	W	L				L	
[(⌣)(⌣ ⌣)]	[⌣(⌣ ⌣)]	L			W		W		W		
[⌣⌣(⌣ ⌣)]	[(⌣ ⌣)(⌣ ⌣)]	W			L			L			
[⌣(⌣ ⌣)(⌣	[(⌣ ⌣)⌣(⌣ ⌣)]				L	W					
[⌣(—	[(⌣ —)⌣(⌣ ⌣)]				L	W					L
[⌣⌣⌣⌣(⌣ ⌣	[(⌣ ⌣)(⌣ ⌣)(⌣	W			L	L		L			

Given the data in Table 7, the learner starts out with 10 constraints that need to be ranked. RCD works by asking a very simple question: which of the constraints could be ranked at the top? Fortunately, the answer is also very simple: any constraint which does not have an 'L' in its column. If a constraint has an 'L', then ranking it at the top would cause the loser to beat the winner for the mark-data pair in the row containing the 'L'. RCD proceeds by creating a collection of constraints, called a stratum, and filling it with precisely those constraints having no 'L' in their column. This stratum will go at the top of the ranking. In this case, it is MAIN-RIGHT, FOOT-NONFINAL, and FOOTBIN that are ranked highest by the learner.

{M-R F-NF FTBIN}

Once those constraints have been committed to the top of the hierarchy, RCD checks to see which mark-data pairs have now been accounted for. The answer is: any mark-data pair with a 'W' in the column of one of the constraints just ranked has been accounted for. The ranking of the 'W'-assigning constraint ensures that the winner will beat the loser: none of the constraints preferring the loser have been ranked yet, so the just-ranked 'W'-assigning constraint is guaranteed to dominate all of them. RCD acknowledges this by removing from the list all of the mark-data pairs now accounted for. Once these mark-data pairs have been removed, the columns for the just-ranked constraints can be removed, leaving only columns for constraints not yet ranked.

Table 8

Remaining mark-data pairs after the first pass of RCD

Loser	**Winner**	PARSE	M-L	A-F-R	A-F-L	IAMB	NF	WSP
[⏑⏑(⏑ ⏑)]	[(⏑ ⏑)(⏑ ⏑)]	W		L		L		
[⏑(⏑ ⏑)(⏑ ⏑)	[(⏑ ⏑)⏑(⏑ ⏑)]			L	W			
[⏑(—	[(⏑ —)⏑(⏑ ⏑)]			L	W			L
[⏑⏑⏑⏑(⏑ ⏑)	[(⏑ ⏑)(⏑ ⏑)(⏑	W		L	L	L		

Now RCD performs exactly the same steps recursively, on the remaining data. The question "what constraints among those remaining can be ranked highest?" is answered by "those constraints that no longer have an 'L' in their column." Those constraints (here, PARSE, M-L and NF) are placed in the second stratum, dominated by the constraints of the first.

{M-R F-NF FTBIN} ≫ {PARSE M-L NF}

When RCD removes the mark-data pairs now accounted for, two remain.

Table 9

Remaining mark-data pairs after the second pass

Loser	**Winner**	A-F-R	A-F-L	IAMB	WSP
[⏑(⏑ ⏑)(⏑ ⏑)	[(⏑ ⏑)⏑(⏑ ⏑)]	L	W		
[⏑(—	[(⏑ —)⏑(⏑ ⏑)]	L	W		L

The third pass ranks two more constraints, A-F-L and IAMB.

{M-R F-NF FTBIN} ≫ {PARSE M-L NF} ≫ {A-F-L IAMB}

When RCD removes the mark-data pairs accounted for by the third stratum, it removes both remaining pairs (A-F-L, now ranked, prefers both winners), thus guaranteeing that on the next pass none of the remaining constraints have an 'L' in their column. After this fourth pass, all constraints have been ranked, and RCD is complete, returning the following stratified hierarchy:

{M-R F-NF FTBIN} ≫ {PARSE M-L NF} ≫ {A-F-L IAMB} ≫ {A-F-R WSP}

A stratified hierarchy permits more than one constraint to be at the same level in the ranking. Constraints occupying the same stratum are not distinguished in ranking relative to each other. If the data do not determine a ranking relationship between two constraints, RCD does not make one, and returns a stratified hierarchy as a result.

The correctness of this stratified hierarchy for the mark-data pairs can been seen more directly if the constraint columns of Table 7 are reordered to reflect the hierarchy, as has been done in Table 10. Dashed lines separate constraints in the same stratum, while solid lines separate strata.

Table 10

Mark-data pairs with the final constraint hierarchy.

Loser	**Winner**	M-R	F-NF	FTBIN	PARSE	M-L	NF	A-F-L	IAMB	A-F-R	WSP
[(◡◡)]	[(◡ ◡)]		W						L		
[(◡ ◡)◡]	[◡(◡ ◡)]	W				L	L	L		W	
[(◡)(◡ ◡)]	[◡(◡ ◡)]		W	W	L					W	
[◡◡(◡ ◡)]	[(◡ ◡)(◡ ◡)]				W				L	L	
[◡(◡ ◡)(◡ ◡	[(◡ ◡)◡(◡ ◡)]							W		L	
[◡(—	[(◡ —)◡(◡ ◡)]							W		L	L
[◡◡◡◡(◡ ◡	[(◡ ◡)(◡ ◡)(◡				W			L	L	L	

Observe that, for each mark-data pair, the leftmost non-blank constraint cell is a 'W'. For each pair, the highest-ranked constraint distinguishing the winner from the loser prefers the winner.

2.2.3. Multi-Recursive Constraint Demotion

RCD operates on a given set of mark-data pairs. It assumes that appropriate target structural descriptions (descriptions that should be optimal in the target grammar) have been identified and that, for each target (the winner), a useful competitor (the loser) has been selected to form a mark-data pair. Some kind of intelligent selection of competitors is therefore required (selection of appropriate target descriptions, based upon overt data, is the topic of section 3). In typical OT systems, a linguistic input has a large (in many cases, infinite) space of candidate structural descriptions, so forming a separate mark-data pair for every competitor of a target candidate would be very computationally expensive, if not hopeless.

Fortunately, there is a very reasonable method for selecting useful competitors, and it fits well into an on-line learning framework. It requires that the learner maintain a working hypothesis grammar at all times (a circumstance quite familiar from the "learnability in the limit" framework (Gold, 1967)). The learner receives target structural descriptions one at a time. As each target is received, the learner extracts the linguistic input from the target, and parses the input according to its current grammar hypothesis. This procedure is known as production-directed parsing (Tesar & Smolensky, 1998), and it simply computes the function defined by the grammar. If the learner's current hypothesis assigns, as the optimal analysis, the very same structural description that was indicated as the target, then as far as the learner can tell, their current hypothesis is fine; no change needs to be made. If, on the other hand, the learner's current hypothesis assigns a different analysis, then the learner needs to change their hypothesis[6].

The detection of a mismatch between the target description and the currently optimal one tells the learner when to change their grammar hypothesis. As demonstrated above, learning in an OT system is likely to require the creation of a mark-data pair. Fortunately, the process of detecting when change is needed provides the necessary ingredients for a mark-data pair as a side effect. Specifically, the learner is guaranteed to make progress if they form a mark-data pair by adopting the current target as the winner, and adopting the description optimal under the learner's current hypothesis as the loser (Tesar, 1998a).

A learner can learn the language from a sequence of target structural descriptions by combining the mark-data pair formation strategy just described with RCD, forming an algorithm called Multi-Recursive Constraint Demotion (Tesar, 1997). The learner maintains at all times a list of mark-data pairs and a corresponding constraint hierarchy, which constitutes their grammar hypothesis. The corresponding

[6] This is in essence error-driven learning again, only the forms being provided to the learner are full structural descriptions, rather than overt forms.

constraint hierarchy is always the result of applying RCD to the list of mark-data pairs. At the outset, the learner has no mark-data pairs, so their hypothesis is the result of applying RCD to an empty list, namely a monostratal hierarchy with all constraints tied in a single stratum. As each target description is encountered, the learner consults its current hypothesis to see if it would select a different competitor as optimal for the same input. If it would, then the description selected by the learner's current hypothesis is paired (as the loser) with the target description (as the winner). This mark-data pair is added to the learner's list of mark-data pairs. RCD is then applied to the (newly extended) list to derive a new constraint hierarchy. Given this new grammar hypothesis, the learner will reparse the input to see if it now selects as optimal the target structural description. If not, it forms another mark-data pair to add to the list, repeating until it arrives at a hypothesis by which the target description is optimal. Once finished with that target, the learner then moves on to the next target description. Once the learner has reached a correct grammar hypothesis, mismatches stop occurring (the hypothesis generates every subsequent target description).

To give the flavor of how a hypothesis develops using MRCD, here are the first few steps in the learning of the correct grammar for stress in Polish (these are the first four mark-data pairs in the development of the correct hierarchy in the example given below in section 3.3). After the first mark-data pair is obtained, the learner applies RCD to obtain a two-stratum hierarchy. This is shown in Table 11.

Table 11

Ranking after 1st mark-data pair: {FTBIN PARSE M-R A-F-R F-NF WSP} ≫ {M-L IAMB A-F-L NF}

Loser	**Winner**	FTBIN	PARSE	M-R	A-F-R	F-NF	WSP	M-L	IAMB	A-F-L	NF
[(spoko j)ny]	[spo(ko jny)]			W	W	W		L	L	L	L

When the learner determines their second mark-data pair, it is added to the first, and RCD is re-applied to the list, resulting in a hierarchy in which Parse is demoted, relative to the previous hierarchy.

Table 12

Ranking after 2nd pair: {FTBIN M-R A-F-R F-NF WSP} ≫ { PARSE M-L IAMB A-F-L NF}

Loser	**Winner**	FTBIN	M-R	A-F-R	F-NF	WSP	PARSE	M-L	IAMB	A-F-L	NF
[(spoko j)ny]	[spo(ko jny)]		W	W	W			L	L	L	L
[(spo koj)(ny)]	[spo(ko jny)]	W		W	W		L	W		W	

Table 13 and Table 14 show the next two states in the learner's development.

Table 13

Ranking after 3rd pair: {FTBIN M-R F-NF WSP} ≫ { PARSE M-L A-F-L NF} ≫ {A-F-R IAMB}

Loser	**Winner**	FTBIN	M-R	F-NF	WSP	PARSE	M-L	A-F-L	NF	A-F-R	IAMB
[(spoko j)ny]	[spo(ko jny)]		W	W			L	L	L	W	L
[(spo koj)(ny)]	[spo(ko jny)]	W		W		L	W	W		W	
[propa(ga nda)]	[(pro pa)(ga nda)]					W				L	L

Table 14

Ranking after 4th pair: {FTBIN M-R F-NF WSP} ≫ { PARSE M-L NF} ≫ {A-F-R A-F-L IAMB}

Loser	Winner	FTBIN	M-R	F-NF	WSP	PARSE	M-L	NF	A-F-R	A-F-L	IAMB
[(spoko j)ny]	[spo(ko jny)]		W	W			L	L	W	L	L
[(spo koj)(ny)]	[spo(ko jny)]	W		W		L	W		W	W	
[propa(ga nda)]	[(pro pa)(ga nda)]					W			L		L
[(re wo)lucjo(ni sta)]	[(re wo)(lu cjo)(ni sta)]					W			L	L	L

MRCD has some important formal properties. First, if given as input a sequence of structural descriptions all of which are consistent with each other, MRCD is guaranteed to find a grammar generating all of the descriptions. It won't get stuck in a local optimum or fail to converge. Second, a strict upper bound can be placed on the data complexity (the amount of data required for the learner to find a correct grammar). If *N* is the number of constraints, it can be proven that not more than *N(N-1)/2* mark-data pairs are necessary to determine the ranking. That upper bound is less than $N^2/2$, and compares quite favorably with the number of distinct possible total rankings, *N!*. Furthermore, in practice this upper bound is usually a gross overestimate of the number of mark-data pairs that MRCD accumulates prior to reaching a correct hierarchy. See (Tesar & Smolensky, 2000) for proofs of these results, along with further analysis.

MRCD, then, provides one of the algorithmic requirements stated in section 1.3.2: an efficient procedure for determining a grammar corresponding to a consistent set of structural descriptions. Section 3 will demonstrate that another algorithmic requirement, an efficient procedure for determining when a set of structural descriptions is mutually inconsistent, falls out automatically from MRCD, and will demonstrate how MRCD can be used to overcome structural ambiguity.

3. Learning with Inconsistency Detection

3.1. RCD Detects Inconsistent Sets of Mark-Data Pairs

When given a list of consistent mark-data pairs, RCD always returns a constraint hierarchy satisfying all of the pairs. When given an inconsistent set of mark-data pairs, RCD does something particularly useful: instead of running on and on in a futile effort to find a ranking that will work, it detects the inconsistency and grinds to a halt. Consider the list of (two) mark-data pairs in Table 15.

Table 15

Inconsistent mark-data pairs

Loser	Winner	A-F-R	A-F-L	IAMB	F-NF	M-R	M-L	NF	PARSE	FTBIN	WSP
[(⏑⏑)⏑]	[⏑(⏑ ⏑)]	W	L	L	W	W	L	L			
[⏑⏑(⏑ ⏑)]	[(⏑⏑)⏑⏑]	L	W	W	L	L	W	W			

RCD, as always, starts by identifying which constraints can be ranked highest, which means finding the constraints with no 'L' in their column. In this case, three constraints have empty columns, and can be ranked on the first pass. No mark-data pairs are removed on the first pass, however, because the ranked constraints prefer no winners. On the second pass, every remaining constraint has an 'L'. This means that the mark-data pairs collectively require that every remaining constraint be dominated by some other (remaining) constraint. That can't happen in a strict dominance hierarchy, immediately revealing that the

list is inconsistent. For the particular data involved, what it means is that, while there are some rankings by which [⏑ (⏑ ⏑)] is grammatical, and there are some rankings by which [(⏑ ⏑) ⏑ ⏑] is grammatical, there are no rankings in which [⏑ (⏑ ⏑)] and [(⏑ ⏑) ⏑ ⏑] are simultaneously grammatical.

Inconsistency detection has the same upper bound on data complexity as the learning of constraint rankings, meaning that by the time a set of mark-data pairs reaches a size equal to the upper bound, the set of mark-data pairs will either be inconsistent (and that fact will have been detected) or else it will determine a correct ranking.

3.2. Considering Multiple Interpretations of an Overt Form

We have now established that MRCD is capable of both efficiently determining if a set of structural descriptions is consistent, and efficiently finding a correct grammar for a consistent set of structural descriptions. Now we want to consider how to apply these properties in dealing with structural ambiguity of overt forms.

3.2.1. Parsing and Error Detection

The learner can perform error detection by using robust interpretive parsing (RIP). This is the same parsing procedure that the RIP/CD algorithm uses (see section 1.2.5). RIP, given an overt form, optimizes over the set of interpretations of the overt form (the set of full structural descriptions with an overt portion identical to that overt form). The optimal member of that set is the optimal interpretation of the overt form. The learner then applies production-directed parsing (PDP) to the input portion of the optimal interpretation, just as with MRCD. To check for an error, the learner compares the result of RIP (the optimal interpretation of the overt form) with the result of PDP (the current hypothesis' optimal analysis of the input portion). If the two don't match, there is an error, and learning must take place (what the learner does to change their hypothesis in response to an error is different from RIP/CD, and will be presented in the next section).

PDP and RIP can be likened to language production and language comprehension, respectively. They both optimize with respect to a constraint ranking in the same way. The difference is the definition of the set of competing candidates. With PDP, the candidate set is determined by a linguistic input: all candidates sharing that linguistic input compete with each other. With RIP, the candidate set is determined by an overt form: all candidates sharing that overt form compete with each other. For our metrical stress system, an overt form is a string of syllables with stress levels, such as [⏑ – ⏑ ⏑], while a linguistic input is a string of syllables without any stress levels, such as [⏑ – ⏑ ⏑]. Error detection applies RIP to an overt form, and then applies PDP to the corresponding linguistic input. For our metrical stress system, the set of candidates for RIP is a strict subset of the set of candidates for PDP. The candidates for RIP must match the observed stress levels of the overt form. The candidates for PDP need only match the corresponding linguistic input; they will match the overt form in the number of syllables and their weights, but may set the stress levels to be different from the overt form.

A simple illustration, using four constraints, is given in Table 16.

Table 16

Two forms of optimality: RIP and PDP

Overt Form:	[⏑ ⏑ ⏑]	A-F-L	FOOT-NONFINAL	IAMBIC	A-F-R
optimal interpretation (RIP)	[(⏑ ⏑) ⏑]		*		*
another interpretation	[⏑ (⏑ ⏑)]	*		*	
yet another interpretation	[⏑ (⏑) ⏑]	*	*		*
optimal description of the input (PDP)	[(⏑ ⏑) ⏑]			*	*

The ranking in the tableau, ALL-FEET-LEFT ≫ FOOT-NONFINAL ≫ IAMBIC ≫ ALL-FEET-RIGHT, can be used to impose an ordering upon any set of candidates. When the learner uses this ranking to pronounce a word, they use PDP, which optimizes over all possible structural descriptions for the input. When given the input [◡ ◡ ◡] and the indicated ranking, PDP selects the structural description [(◡ ◡) ◡]: this description is more harmonic than any of the other candidate structural descriptions for the input, thus it is the optimal description of that input. However, when the learner hears an overt form, they use RIP to interpret it. RIP optimizes over a restricted set of candidates, the possible interpretations of the overt form. When given the overt form [◡ ◡ ◡] and the indicated ranking, RIP selects the interpretation [(◡ ◡) ◡]: this structural description is more harmonic than any of the other interpretations, thus it is the optimal interpretation. For this overt form, RIP can only choose from among candidates that have main stress on the middle syllable (and no secondary stresses).

One important consequence of using Optimality Theory as the base linguistic theory is that it allows parsing to interpret overt forms that are ungrammatical according to the grammar (constraint ranking) being used to parse. Even if the learner's current ranking would assign a different stress pattern to the syllables of the observed overt form (in the example, putting main stress on the first syllable), the learner can still use their current ranking to select a "best guess" interpretation of the overt form, rather than simply halting with a conclusion of "ungrammatical form", as is the case with most non-optimizing grammatical theories. However, the learner has not lost any awareness of what is and is not grammatical. The learner can determine that the overt form [◡ ◡ ◡] is not grammatical according to their current grammar by computing, via PDP, the optimal description for the corresponding input [◡ ◡ ◡]. The optimal description [(◡ ◡) ◡] (with main stress on the first syllable) is more harmonic than any of the interpretations of the overt form (all of which have main stress on the middle syllable). It is this overt discrepancy between the results of RIP and PDP that indicates an error to the learner. When given an overt form that is grammatical according to the current grammar, RIP returns an interpretation that is the grammatical structural description for the input, that is, it matches the result of PDP.

No special mechanisms are needed for error detection under this approach. The interpretation of the overt form uses the same process as regular language comprehension, and the generation of the current hypothesis' analysis uses the same process as regular language production. Error detection amounts to the learner observing that what they heard is different from what they would have said.

For the experiment described in section 4, PDP and RIP were performed by dynamic programming algorithms developed for parsing with Optimality Theoretic grammars (Tesar, 1995, 1998c).

3.2.2. Considering Multiple Interpretations

What remains is to determine what the learner will do in response to the detection of an error. To put things in perspective, let us first consider a simple way that in general will not produce satisfactory results. MRCD can be applied to a set of full structural descriptions, and it will either determine that the set is inconsistent, or return a constraint hierarchy consistent with all of the descriptions. This means that we could try to deal with structural ambiguity by collecting a set of overt forms, and for each overt form generate all possible interpretations of the form. If the overt forms are adequate to determine a grammar, then there should be only one combination of interpretations, one interpretation per overt form, that is consistent, namely the set of interpretations assigned by the target grammar. All other combinations of interpretations should be inconsistent. Therefore, MRCD can be applied to each combination of interpretations; all but one combination should result in inconsistency, and the correct combination will return the learned constraint hierarchy.

A simple illustration of the notion 'combinations of interpretations' is given in Figure 1. Suppose we have three overt forms, OvertA, OvertB, and OvertC. Each overt form is ambiguous between two possible interpretations. The number of combinations of interpretations is 2 * 2 * 2 = 8 combinations.

3 Overt Forms:	OvertA	OvertB	OvertC
6 Interpretations:	InterpA1 InterpA2	InterpB1 InterpB2	InterpC1 InterpC2

8 Combinations of Interpretations:

InterpA1 InterpB1 InterpC1

InterpA1 InterpB1 InterpC2

InterpA1 InterpB2 InterpC1

InterpA1 InterpB2 InterpC2

InterpA2 InterpB1 InterpC1

InterpA2 InterpB1 InterpC2

InterpA2 InterpB2 InterpC1

InterpA2 InterpB2 InterpC2

Figure 1. Combinations of interpretations of overt forms. 3 overt forms, each 2-ways ambiguous, yields 2 * 2 * 2 = 8 combinations.

As long as there are a finite number of overt forms, and a finite number of interpretations of each overt form, this approach will work. But it will not in general be practical. The problem is that the number of combinations grows exponentially in the number of ambiguous overt forms. If there are M overt forms, and each has only two possible interpretations, the number of combinations is 2^M. Increasing the degree of ambiguity only makes things worse. Once the number of ambiguous overt forms to be used gets much larger than the number of constraints, the number of combinations of interpretations will grow much larger than the number of constraint rankings. At that point, it would be more efficient to simply test directly each total ranking of the constraints. Fortunately, there is a better way to make use of inconsistency detection. This approach has the further virtue that it does not require a fixed set of overt forms to be enumerated in advance; it can process overt forms one at a time, as it receives them.

The learner starts with a single grammar hypothesis, containing no mark-data pairs. When the first overt form is encountered, error detection takes place as described in section 3.2.1. If the form does not cause an error, no learning takes place, and the learner moves on to the next overt form. In all likelihood, an error will occur for the first overt form (it is possible for a form to not cause an error at the outset, but uncommon). At that point, the learner must contend with the structural ambiguity of the overt form. The learner needs to generate a set of interpretations of the ambiguous overt form such that the correct interpretation is guaranteed to be among them. The simplest way is if the learner generates every possible full interpretation of the overt form, and separately applies MRCD to each one. In each case, MRCD starts with the hypothesis the learner began with. If an interpretation is not a possible optimum (it cannot be optimal under any ranking), then MRCD will detect inconsistency, and that interpretation may be discarded. Any interpretation that is possibly optimal will result in a grammar hypothesis with one or more mark-data pairs. Each such hypothesis is retained by the learner when it goes on to consider the next

overt form[7]. This means that the learner must have the capacity to maintain more than one grammar hypothesis across forms.

The learner then proceeds to the next overt form. Error detection takes place again, only now the learner is separately checking the overt form for error against each grammar hypothesis it possesses. The learner proceeds as described for the first overt form, only with a parallel process for each grammar hypothesis. For each hypothesis on which an error occurs for the new overt form, all interpretations of the new overt form are considered.

Two things can happen at this point that have very real consequences for the learner. First, it is quite possible that the learner has more than one hypothesis from the first overt form, and that more than one of those hypotheses results in an error on the new overt form. This is combinatorial growth along the lines illustrated in figure 1 above: if the learner constructs separate hypotheses for InterpA1 and InterpA2, and each hypothesis results in an error on OvertB, then the learner will be trying to construct separate hypotheses combining the hypotheses supporting InterpA1 and InterpA2 with each of InterpB1 and InterpB2. If this growth remains unchecked, then even though the learner is only processing one overt form at a time, the number of hypotheses simultaneously maintained by the learner will grow rapidly.

The second thing that can happen provides just such a check. An interpretation of the second overt form could prove to be inconsistent with an interpretation of the first overt form. If MRCD is attempting to reconcile InterpB1 with the hypothesis supporting InterpA2, and the two together are inconsistent, then that combination is discarded. This means that the learner will never have to consider that combination of interpretations together with any of the interpretations of subsequent overt forms; a large piece of the combinatorial growth is thus cut off, saving the learner much effort. As more data (overt forms) are observed, the hypotheses maintained by the learner become more constrained: they must have been consistent with all overt forms observed thus far. More forms means more potential for contradiction to be detected. This is the Inconsistency Detection Learner (IDL): it determines the correct interpretations of the overt forms by detecting inconsistencies between incorrect interpretations and other data.

In a sense, IDL carries out a race between two forces, one trying to increase the number of hypotheses the learner must maintain, and the other trying to reduce the number of hypotheses maintained. The feasibility of IDL depends upon the relative strength of the two forces. If the linguistic theory being employed is such that the interpretations assigned to different forms are strongly interrestrictive, then IDL will be efficient, because when an error occurs, most of the interpretations of the overt form will be discarded due to inconsistency with data already observed. If the forms aren't so strongly interrestrictive, IDL may get swamped by an unrealistically large number of hypotheses to maintain. An advocate of IDL is thus someone who is "putting their money" on linguistic theory, that it will be restrictive enough to overcome the exponential growth implied by the ambiguity of the overt forms.

Apart from the combinatorial growth in the combinations of interpretations, there is the question of the number of possible interpretations of single overt forms. As stated above, the simplest and most general strategy is for the learner to generate and evaluate all possible interpretations of an overt form. In the experiment presented here, involving the metrical stress system described in section 2.1, that strategy was feasible and adequate. However, more sophisticated methods are possible in which the learner limits their effort to only a subset of the possible interpretations of an overt form, especially if a number of the interpretations are not possible grammatical forms (they are not grammatical under any constraint ranking). Such methods will likely be motivated when this learning approach is applied to linguistic

[7] In principle, all that the learner crucially needs to store are the interpretations (the winners of the mark-data pairs); the mark-data pairs and corresponding constraint hierarchy can be reconstructed via MRCD. It is the set of interpretations that is the basis for the hypothesis. In practice, that could mean a great deal of duplication of effort, and storing the mark-data pairs seems much more efficient, especially considering the modest number of mark-data pairs that will ever be associated with any single hypothesis.

domains where the degree of ambiguity of overt forms is greater than that of the stress system. Some proposals for such methods are discussed in section 5.4.

3.3. An Illustration: Learning Polish Stress

Here is a somewhat simplified illustration of IDL, learning the primary word stress pattern of Polish, a trochaic, quantity insensitive language. In Polish word stress (Rubach & Booij, 1985), main stress goes on the penultimate syllable, and secondary stresses iterate from the beginning of the word, starting with the first syllable. In words with an odd number of syllables, no secondary stress is placed on the antepenultimate syllable, avoiding stress clash.

Table 17

The Polish word stress pattern

Stress Pattern	**Example**	**Gloss**
[(σ σ)]	*domu*	'house'
[σ (σ σ)]	*spokojny*	'quiet'
[(σ σ) (σ σ)]	*propaganda*	'propaganda'
[(σ σ) σ (σ σ)]	*saksofonista*	'saxophone player'
[(σ σ) (σ σ) (σ σ)]	*rewolucjonista*	'revolutionary'
[(σ σ) (σ σ) σ (σ σ)]	*rewolucjonistami*	'revolutionary' (instr. pl.)

Suppose that the first word the learner hears is *spoko jny*. This is a tri-syllabic word with medial main stress. Under the current system, it has three possible interpretations, here labeled A1, A2, and A3:
A1: *spo*(*ko jny*)
A2: *spo*(*ko j*)*ny*
A3: (*spoko j*)*ny*
The learner attempts to construct a hypothesis for each of the three interpretations. It succeeds for A1 after accumulating two mark-data pairs, constructing hypothesis H(A1). It also succeeds for A3 after accumulating three mark-data pairs, constructing hypothesis H(A2). However, A2 proves to be inconsistent after one mark-data pair. In this case, interpretation A2 is not a possible optimal form; it will be suboptimal under any ranking. Thus, the learner can discard interpretation A2. After processing *spoko jny*, the learner is maintaining two hypotheses, H(A1) and H(A2).

The next overt form the learner encounters is *pro paga nda*. This overt form has five possible interpretations:
B1: (*pro pa*)(*ga nda*)
B2: (*pro pa*)(*ga n*)*da*
B3: (*pro*)*pa*(*ga n*)*da*
B4: (*pro*)(*paga n*)*da*
B5: (*pro*)*pa*(*ga nda*)
Hypothesis H(A1) does not definitively stress the word in the attested manner, so there is some learning to be done. First the learner attempts to separately reconcile each of the five interpretations with hypothesis H(A1). The learner succeeds with interpretation B1 after adding one additional mark-data pair, yielding the hypothesis H(A1B1). Each of the other four interpretations (B2-B5) proves to be inconsistent with H(A1), with a total of 5 learning steps needed to determine that. A learning step is the addition of a new mark-data pair to a list (along with the application of RCD to that list).

Hypothesis H(A3) also does not definitively stress the word in the attested manner, so the learner attempts to separately reconcile each of the five interpretations with hypothesis H(A3). The learner

succeeds with interpretation B4 after adding two additional mark-data pairs, yielding the hypothesis H(A3B4). Each of the other four interpretations (B1-B3,B5) proves to be inconsistent with H(A3), with a total of 9 learning steps needed to determine that.

Notice that inconsistency detection has cut down the possible combinations considerably. Over the first two forms, there are 3 * 5 = 15 possible combinations of interpretations, but only two combinations are actually consistent, A1B1 and A3B4. After only 23 learning steps, the learner has two hypotheses.

The next overt form encountered by the learner is *sa ksofoni sta*. This overt form has six possible interpretations (I won't bother to list them all here). Hypothesis H(A1B1), when used to generate a stress pattern this word, assigns the attested stress pattern. Thus, no learning need take place for H(A1B1) with this word. Not so hypothesis H(A3B4), which does not assign the attested stress pattern. The learner attempts to reconcile each of the six interpretations of *sa ksofoni sta* with H(A3B4). However, each of the six interpretations proves inconsistent with H(A3B4), after a total of six learning steps (one per interpretation). Thus, after only three pieces of data, and 29 learning steps, the learner has reduced the possibilities to a single hypothesis, H(A1B1).

Hypothesis H(A1B1) will remain unchanged until the learner encounters a longer form, like *re wolu cjoni sta*. H(A1B1) does not definitely stress this word as attested, so the learner must consider all 13 interpretations. Interpretation D1, (*re wo*)(*lu cjo*)(*ni sta*), is consistent with the earlier data, forming hypothesis H(A1B1D1) after just one additional mark-data pair. The other 12 interpretations are all inconsistent, which can be determined with 16 learning steps of effort.

The learner now possesses a hypothesis for Polish, H(A1B1D1), having taken a total of 46 learning steps to find it. The learner never had to carry more than two simultaneous hypotheses from one data form to the next. This example is somewhat simplified for purposes of illustration; in the experimental treatment with checking of prior winners and Block data order, 59 learning steps were used to learn the complete Polish pattern (see section 4).

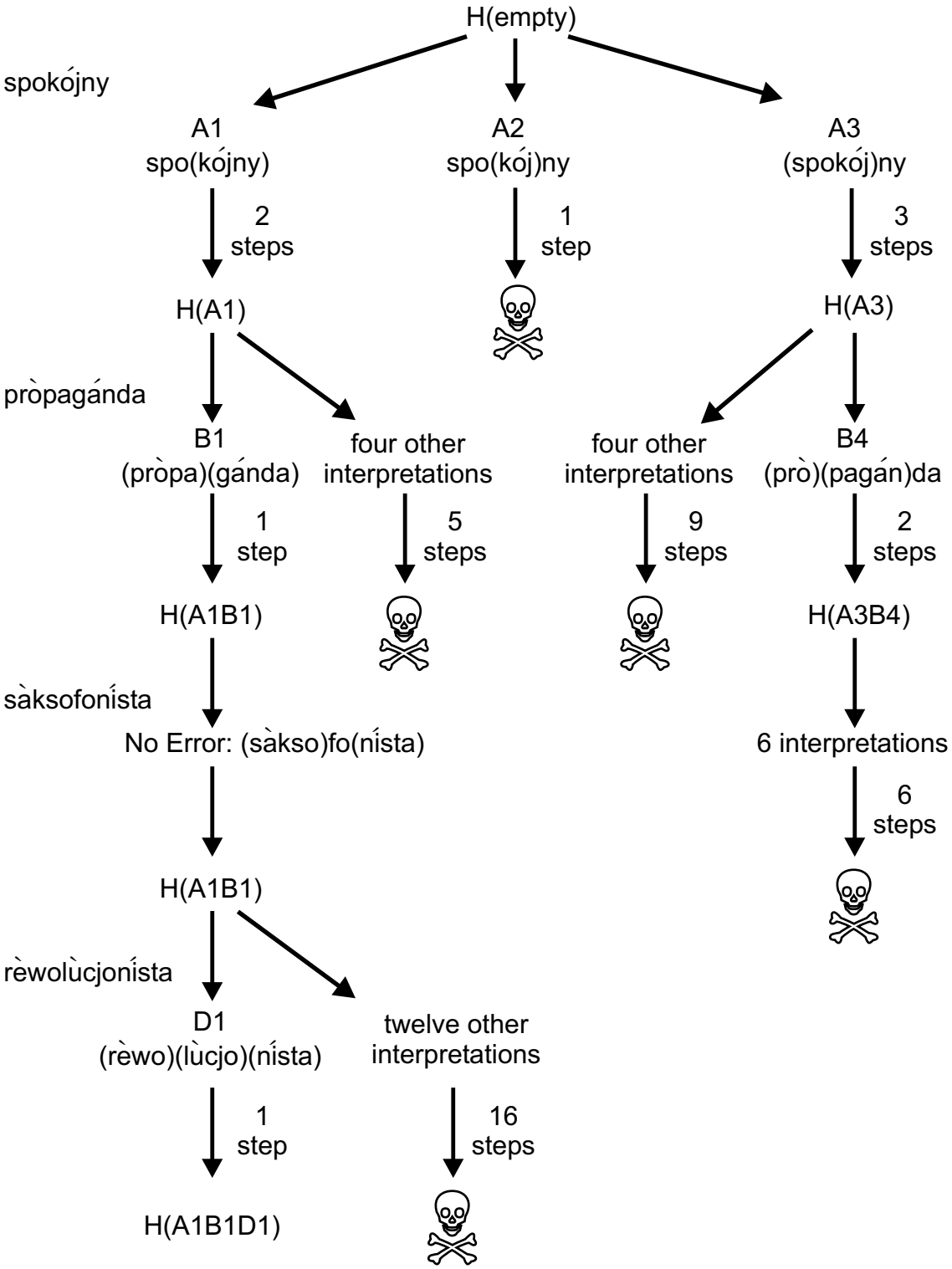

Figure 2. Learning Polish (46 learning steps are shown)

This illustrates the three ways in which IDL can cut down on the combinatorial explosion in combinations of interpretations:

- Eliminate interpretations which cannot possibly be optimal.

- Detect inconsistent combinations of interpretations.
- Correctly parse the overt form with a hypothesis, thus avoiding the consideration of multiple interpretations.

The first way is illustrated by candidate A2. This candidate cannot be optimal under any ranking. MRCD will always detect inconsistency when presented with such a form. It usually does so extremely quickly; in the Polish example, it took only one mark-data pair to demonstrate the futility of candidate A2. For the longer forms with larger numbers of interpretations, a correspondingly increasing number of the interpretations are not possibly optimal. Those candidates are guaranteed to be eliminated on the spot, and will not form combinations with interpretations of subsequent overt forms.

The second way, the elimination of inconsistent combinations of interpretations, is the heart of algorithm, doing the most work. For example, for the path under hypothesis H(A1), the combinations labeled "four other interpretations" includes (as one of the four) the combination of H(A1) with interpretation B4, (*pro*)(*paga n*)*da*. B4 is a possible optimum, and A1 is a possible optimum, but the two cannot simultaneously (in the same grammar) be optimal. Thus, the attempt to combine B4 with H(A1) terminates in inconsistency, and is discarded, avoiding any future expansion on that combination.

The third way is illustrated when hypothesis H(A1B1) encounters the third overt form, *re wolu cjoni sta*. The optimal description for the underlying form, under hypothesis H(A1B1), stresses the word exactly as given in the overt form, so no error occurs (in the sense of error-driven learning). The learner then concludes that it is interpreting the form correctly, and does not bother to consider alternative interpretations of the overt form, thus avoiding any combinatorial growth due to the number of interpretations of that overt form.

3.4. Algorithm Details

This section addresses a couple of less obvious issues that arise regarding the use of IDL with Optimality Theory. The details of how these issues are handled algorithmically are spelled out here.

3.4.1. Parsing with Stratified Hierarchies

RCD produces stratified hierarchies, and commonly some of the strata will be occupied by more than one constraint. The error-driven learning approach requires the learner to be able to perform both production-directed parsing and robust interpretive parsing using stratified hierarchies. Basic OT defines the optimizations for total rankings. For stratified hierarchies, more must be said: an interpretation of stratified hierarchies must be provided, one that determines the relative harmony of any pair of candidates. The interpretation that was presumed in the earliest formulations of constraint demotion is the one sometimes called "pooling the marks": all constraints of a given stratum are treated as equal, and two candidates are compared relative to that stratum by adding together ("pooling") the violations of each of the constraints in the stratum. The candidate with fewer total violations for the stratum is the more harmonic (the better candidate), relative to that stratum. If two candidates have the same total number of violations of the constraints in a stratum, they tie on that stratum (regardless of how the violations are distributed among the constraints in the stratum), and evaluation continues with the next stratum down in the hierarchy.

Mark pooling is illustrated in Table 18, where five descriptions compete with respect to the stratified hierarchy {C1} ≫ {C2 C3} ≫ {C4}. Candidates D1 and D2 "tie" on the middle stratum; each has a total of one violation on the stratum. Candidates D3 and D4 are eliminated at the second stratum, because they have more than the minimal total number of violations on the stratum. Candidate D5 plays no role in the competition in the middle stratum, because it has already been eliminated due to its violation of C1, which occupies the first (highest) stratum. The candidates that are still in contention after the second stratum are still competing, and the decision is made by the remainder of the hierarchy (in this case, the stratum containing C4).

Table 18

Illustration of Mark Pooling (C2 and C3 occupy the same stratum): D1 is the most harmonic

	C1	C2	C3	C4	C5
D1		*			*
D2			*	*	
D3		**			
D4		*	*		
D5	*				

This interpretation of tied constraints does allow the possibility of ties in harmony between candidates with non-identical constraint violations, if those candidates have an equal number of violations on each stratum.

The mark pooling interpretation (which we will call MPool) of stratified hierarchies is well-defined, and works well with the dynamic programming algorithms used for parsing in the experiment of section 4. However, it can sometimes cause error-driven learning to stop before reaching a hierarchy in which the desired winner beats every competitor by constraint domination alone. This happens when the winner and a candidate loser conflict on constraints in the same stratum, but the loser has more overall violations within the stratum. This is illustrated in Table 19.

Table 19

Pooling marks obscures the conflict between C2 and C3

	C1	C2	C3	C4
winner		**		*
loser			***	

Constraints C2 and C3 are in the same stratum. The winner and the loser conflict on C2 and C3, but the winner wins under this hierarchy because it has fewer total violations. However, for the winner to beat the loser properly, C2 (or perhaps C4) must strictly dominate C3. If the winner and the loser were to be combined into a mark-data pair and added to the list, RCD would ensure that C3 appeared below C2 in the resulting hierarchy, and more information (restrictions on the hierarchy) would have been extracted from this winner. Because of the fluke fact that the winner violates C2 few times than the loser violates C3, error-driven learning using MPool will fail to construct this mark-data pair at that point. Failing to construct and add this mark-data pair means less work for the learner at that instant, but it is potentially costly to the learner down the road: extra information increases the likelihood of detecting inconsistency with other forms, and early detection of inconsistency is what the learner is relying on to avoid the combinatorial growth in the combinations of interpretations.

To deal with this, a second kind of interpretation of stratified hierarchies is also employed, one in which "conflicts tie". Under this interpretation (which we will call CTie), when two candidates are being compared on a stratum, if the candidates have conflicting outcomes on different constraints in the stratum, then the two candidates are declared to tie on the entire hierarchy. Precisely, two candidates have conflicting outcomes on a stratum if one of the constraints of the stratum prefers one candidate, while another of the constraints in the stratum prefers the other candidate. Because error-driven learning explicitly strives to eliminate ties in harmony (it doesn't stop until the winner is the sole optimum), if the winner and a loser are declared to be tied, the learner will form a mark-data pair with them, and benefit from the additional information.

Comparison of two candidates using CTie still proceeds from the top stratum down. If at least one constraint of the stratum prefers candidate D1 to D2, and none of the constraints of the stratum prefer D2,

then D1 is the better candidate (just like with MPool). If all constraints of the stratum are indifferent (D1 and D2 violate each constraint the same number of times), then the comparison moves down to the next stratum of the hierarchy (again just like with MPool). If at least one constraint of the stratum prefers D1, and at least one constraint of the stratum prefers D2, then the comparison is complete, and D1 and D2 are declared tied (this differs from MPool). It is important that in this last case, the comparison does not proceed down to subsequent strata; that would result in the comparison being determined by one of the subsequent strata, and the higher unresolved constraint conflict could remain undetected, just as with MPool.

CTie can detect certain constraint conflicts that MPool will miss. However, it has one property that make it a bad replacement for MPool in general. The comparison relation determined by CTie is not transitive: it is possible that D1 $\approx$ D2 (D1 ties with D2), and D2 $\approx$ D3, but D1 $\not\approx$ D3. An example is illustrated in Table 20; specifically in this example, D1 is less harmonic than D3 (D1 $\prec$ D3).

Table 20

Comparison with CTie: D1$\approx$D2, D2$\approx$D3, D1$\prec$D3

	C1	C2
D1		**
D2	*	
D3		*

The same relationships among the three candidates, translated in to the comparative tableau format, are depicted in Table 21.

Table 21

Pairwise comparison with CTie

Loser	**Winner**	**C1**	**C2**	**Relative Harmony**
D1	D2	L	W	conflict, so tie
D2	D3	W	L	conflict, so tie
D1	D3		W	no conflict

This makes parsing a bit more complicated[8], because it is less obvious what it means to say that a set of candidates are the optimal candidates under a stratified hierarchy. The definition employed here is that an optimal candidate is one that is not less harmonic than any other candidate. This effectively means that an optimal candidate must be in a tying conflict with every other optimal candidate. For a set of candidates to all be optimal, it must be the case that any pair of candidates from the set are in a tying conflict as defined by CTie (or else have identical violation marks).

For the experiment described in this paper, CTie is used sparingly. When the learner is first presented with a new overt form, MPool alone is used to determine if an error has occurred, and if any learning needs to be done. If an error is detected, the learner pursues a separate hypothesis for each interpretation of the overt form. In testing each interpretation, MPool is used to construct new mark-data pairs until either an inconsistency is detected or the interpretation is optimal according to MPool. At that

[8] Specifically, if during parsing a parser constructs a set of candidates that tie and might be optimal, and a new candidate is then constructed, the new candidate cannot simply be compared to one member of the set to determine how it compares with all members of the set (as is the case with MPool). The new candidate must be compared to all members of the set, to check if the new candidate is less harmonic than any members of the set, and to eliminate any set members that are less harmonic than the new candidate.

point, the learner re-parses using CTie to check for constraint conflicts that MPool might have missed. So, CTie is only employed when the learner is already actively engaged in modifying its grammar hypotheses. This is done for two reasons. First, it holds down the amount of extra parsing the learner does. Second, it remains true to the spirit of a psychological model of a learner whose primary goal is making sense of their world. If the first attempt to parse what they hear seems to fit (it makes sense given what the learner already knows about the world), then the learner presumes that interpretation to be correct. It is when an error of some kind occurs on the first parse attempt that the learner is motivated to scrutinize more carefully for unresolved constraint conflicts.

3.4.2. Checking Prior Winners in a Hypothesis

When MRCD is applied to a given interpretation and a given grammar hypothesis, it adds mark-data pairs (with the given interpretation as the winner) until the resulting hypothesis holds the interpretation to be optimal (unless inconsistency is detected). However, it is not guaranteed that that winner will still be optimal after the addition of further mark-data pairs to the hypothesis in response to other winners for other overt forms. This is another way in which error-driven learning may fail to obtain all the information that is in principle available from an interpretation when it first encounters it. In other words, when MRCD is applied to an interpretation, the mark-data pairs it accumulates with that interpretation as winner are necessary conditions upon a hierarchy, but they may not constitute sufficient conditions to guarantee that the winner will always be optimal.

Here is an illustration of this subtle phenomenon. Suppose the target language is one where main stress is antepenultimate if the antepenultimate syllable is heavy and the penultimate syllable is light, otherwise main stress is penultimate. The learner has initially accumulated the following mark-data pairs, shown in Table 22 in the order that they were added, top to bottom. The winner in the bottom-most mark-data pair, [(— —) ◡], is optimal with respect to the ranking resulting from this list of mark-data pairs.

Table 22

At this point, candidate [(— —) ◡] is optimal

Loser	**Winner**	NF	M-L	A-F-L	FTBIN	WSP	PARSE	M-R	A-F-R	IAMB	F-NF
[(◡ ◡)]	[(◡) ◡]	W			L		L	L	L		
[(◡ ◡)]	[(◡) ◡]	W					L	L	L	W	L
[(◡ —)]	[(◡) —]	W			L	L	L	L	L		
[(— ◡)	[(— ◡)					W				L	W
[(— —)	[(— —)									W	L

However, after two more mark-data pairs have been added, the learner obtains a hierarchy which is consistent with all of the mark-data pairs, but which no longer holds [(— —) ◡] to be optimal. This is shown in Table 23.

Table 23

The ranking after the addition of two more mark-data pairs

Loser	**Winner**	NF	FTBIN	WSP	M-R	A-F-R	PARSE	M-L	A-F-L	IAMB	F-NF
[(∪ ∪)]	[(∪) ∪]	W	L		L	L	L				
[(∪ ∪)]	[(∪) ∪]	W			L	L	L			W	L
[(∪ —)]	[(∪) —]	W	L	L	L	L	L				
[(— ∪) ∪]	[(— ∪) ∪]			W						L	W
[(— —) ∪]	[(— —) ∪]									W	L
[(∪ ∪) ∪ ∪]	[∪ (∪ ∪) ∪]				W	W		L	L		
[(∪ ∪) (∪) ∪]	[∪ (∪ ∪) ∪]		W			W	L	W	W		W

In particular, candidate [(— —) ∪] is no longer more harmonic than its competitor [(—) (—) ∪]. The two candidates conflict on the second stratum, as shown in Table 24. When this winner was seen earlier, all the forms were short enough that the sole foot aligned with the left edge of the word, so the left-aligning constraints A-F-L and M-L remained at the top of the hierarchy. This was sufficient to ensure that the desired winner, [(— —) ∪], was more harmonic than competitor [(—) (—) ∪], the latter having a second foot that is not strictly left-aligned. The high-ranking left alignment constraints dominated WSP, which prefers the competitor placing stress on each heavy syllable. However, when the longer four syllable form is observed, the learner is provided evidence that left alignment cannot be undominated, and A-F-L and M-L are demoted down to below M-R and A-F-R in the hierarchy. This has the side effect of also demoting A-F-L and M-L below WSP, eliminating the ranking relationship that earlier was sufficient to ensure the optimality of the desired winner. To restore the optimality of the desired winner, the learner needs to demote WSP to below A-F-R.

Table 24

The earlier winner [(— —) ∪] is no longer optimal, conflicting with a competitor on the second stratum

Loser	**Winner**	NF	FTBIN	WSP	M-R	A-F-R	PARSE	M-L	A-F-L	IAMB	F-NF
[(—) (—) ∪]	[(— —			**L**		**W**		W	W		W

During learning, the learner eventually will encounter each overt form multiple times. Suppose the learner encounters an overt form a second time, and checks it with respect to a hypothesis containing at least one mark-data pair with a winner for that overt form. One of several things may happen.

- Most of the time, the winner from the mark-data pair is still optimal, and no learning need take place.
- Sometimes, no interpretation of the overt form is currently optimal, due to mark-data pairs that have been added since the last time the overt form was processed. Learning then takes place, but the learner will re-consider all possible interpretations of the overt form, which in a sense duplicates some of the work done previously.
- On occasion, an interpretation of the overt form is optimal, but it is a different interpretation from the one that is the winner of the earlier overt form. In this case, the learner does not add any mark-data pairs, but learner will not detect that the interpretation of the form has shifted.

One technique was investigated for forcing hypotheses to remain consistent with the interpretations that gave rise to them. That technique has the learner check the winners of its mark-data pairs every time a new mark-data pair is added to the list, and a new hierarchy is derived via RCD. If any prior winner is found to no longer be optimal, MRCD is applied to that prior winner, adding more mark-data pairs to the list until either the winner is again optimal, or inconsistency has been detected.

This technique does not cause the learner to review every form observed by the learner, but only the winners present in the actual mark-data pairs stored with the hypothesis. This keeps the amount of additional effort exerted by the learner under control, as it is bounded by the number of mark-data pairs a hypothesis can ever contain. The learner is not storing in memory the assigned interpretation for every form they hear, only those interpretations used to form mark-data pairs. Nevertheless, because checking prior winners does involve a certain amount of extra parsing of forms, experimental treatments were run both without and with the use of checking prior winners. This allows us to see the performance of the basic algorithm without checking prior winners, and to evaluate the impact of checking prior winners on learning.

4. The Computational Experiment

4.1. Characteristics of the Stress System

The OT stress system defines languages which assign stress patterns to words of arbitrary length. For the experiment, 62 words of each language were presented: words with length between 2 and 5 syllables using all possible ordered combinations of light and heavy syllables, a word of 6 light syllables, and a word of 7 light syllables.

Given this base of 62 linguistic inputs, in this system there are 2,140 distinct sets of structural descriptions (languages), each realizable with different constraint rankings. However, some languages completely match in overt forms: there are cases where two different grammars, determined by two different constraint rankings, define languages with identical sets of overt forms, even though the structural descriptions for some of the overt forms differ. Because the learner is only given overt forms as data, these languages cannot be distinguished by the learner. To avoid uninformative duplication in the experiments, the entire set of possible languages was evaluated. For each collection of languages with duplicate sets of overt forms, all but one were removed, so that in the resulting set of languages, only one language remained for each possible set of overt forms. With the duplicates removed, a total of 1,527 languages remained for the learner to be tested on. The learner was separately tested on all 1,527 languages.

The overt forms vary widely in their degree of ambiguity. A form like [◡ ◡] is only two-ways ambiguous, permitting interpretations [(◡) ◡] and [(◡ ◡)]. The form [◡ ◡ ◡ ◡ ◡ ◡ ◡] is 21-ways ambiguous, permitting 21 different interpretations.

Now suppose for a moment that every form in a language is only two-ways ambiguous. Then the number of combinations of interpretations that could be realized is $2^{62} \cong 5 * 10^{18}$. If the number of learning steps the learner must take to learn a language via inconsistency detection scales at all like the number of combinations, then this approach will be hopelessly intractable. Every overt form in every language has at least two possible interpretations, so 2^{62} is a lower bound on all languages in the system. For languages including forms that are more than two-ways ambiguous, the number of possible combinations will be much, much larger still. The number of total rankings[9] of the constraints is tiny by comparison, 10! = 3,628,800. For IDL to be an improvement over simple exhaustive search of all total rankings, it will have to completely escape the combinatorial explosion of combinations of interpretations.

4.2. Purpose and Design of the Experiment

The primary purpose of the computational experiment is to see, using a linguistically plausible OT system, whether IDL can overcome the combinatorics of the system and efficiently learn correct

[9] For those interested in combinatorial comparison with parametric theories, the number of total rankings of 10 constraints, 10! = 3,628,800, is less than the number of distinct parameter setting combinations for a system with 22 binary parameters, 2^{22} = 4,194,304.

grammars from sets of overt forms. IDL is guaranteed to eventually find a correct grammar, so the issue is the amount of computational effort required by IDL to learn. In the terms of the discussion at the end of section 3.2.2, the experiment is run to see which force wins the race, the combinatorial growth in the combinations of interpretations of overt forms, or the combinatorial reduction resulting from the detection of mutually inconsistent interpretations. The measures of computational effort that were used are described in section 4.2.2.

A secondary purpose of the experiment is to examine the effects of two factors upon the performance of IDL, the checking of prior winners and the presentation order of the data. These two factors are discussed in section 4.2.1.

4.2.1. Two factors: checking prior winners, and data order

The experiment contains a total of four treatments. Two factors were varied independently, each with two levels. One of the factors was described in section 3.4.2, the checking of prior winners. For this factor, two levels were used: one level ran the algorithm without the checking of prior winners, while the other level ran the algorithm with the checking of prior winners.

The other factor was the order of presentation of the data to the learning algorithm. Each language consists of a set of overt forms. The overt forms within a language can vary in degree of ambiguity, and it is intuitively plausible that using forms with lower ambiguity early could benefit learning. With metrical stress, degree of ambiguity correlates to some extent with the length (in syllables) of the form: more syllables means more ways of grouping syllables into feet if there are secondary stresses present. Thus, the length of a form is an overtly available indicator that a learner could use to select low ambiguity forms earlier.

For the data order factor, two levels were used. In the first level, the overt form order was randomized prior to presenting them to the learner; this level is labeled “Random” in the results below. In the other level, the overt forms were grouped into blocks by length: the forms of lengths of 2 and 3 syllables were the first block, the forms of 4 syllables were the second block, the forms of 5 syllables were the third block, the 6 syllable form was the fourth block and the seven syllable form was the fifth block. The order of the forms within each block was randomized, and the overall order was established by ordering the blocks (first block first, up to the fifth block last). This level is labeled “Block” in the results below. This is a simple approximation to a strategy of the learner selecting short forms early.

In all four treatments, for each language, once an ordering of the overt forms for the language was established, the forms were presented to the learner in that order. Once the end of the list of forms was reached, presentation continued starting over at the beginning of the list of forms (using the same order). This continued until the learner made a pass through the forms without encountering an error (a mismatch between the overt form and the optimal candidate for one of the learner’s hypotheses) on any of the overt forms. As guaranteed, the learner successfully terminated for every language in each treatment.

4.2.2. Measuring Learning Efficiency

Two quantities were measured during the learning of each language. The first is intended to be a measure of cumulative computational effort by the learner. The unit of measurement is the act, by the learner, of adding a mark-data pair to a hypothesis and generating (via RCD) a new hierarchy. That act is here termed a “learning step.” The sum of learning steps includes not just the number of mark-data pairs in the hypothesis that survives to the end as the learned grammar, but all such operations performed on other hypotheses prior to their being discarded as inconsistent. Detecting inconsistency in a hypothesis necessarily requires at least one mark-data pair, so if the learner must consider a large number of hypotheses during learning, there will be a correspondingly large number of learning steps added to the total for the language.

The other quantity used to evaluate learning performance is intended to measure the maximum load imposed on the learner by the requirement of maintaining multiple hypotheses. This quantity is the

maximum number of simultaneous hypotheses maintained by the learner from one overt form to the next during learning. After processing an overt form, the learner possesses some number of hypotheses in memory. That number was examined after the processing of each form during the learning of a language, and the maximum value over the entire learning process was determined.

For each language, in each treatment, the total number of learning steps and the maximum number of simultaneous hypotheses were recorded. In each treatment, the results are summarized by reporting, across all the languages of the system, the mean, median, and range of each of the two measures.

4.3. The Experimental Results

The results of all four treatments are presented in Table 25.

Table 25

Experimental Learning Results: N = 1,527

	Total Learning Steps			**Max # Simultaneous Hypotheses**		
Data Order	**Mean**	**Median**	**Range**	**Mean**	**Median**	**Range**
No prior winner checking						
Random	83.2	71	2..475	3.4	3	1..16
Block	51.1	43	4..260	2.9	2	1..13
Prior winner checking						
Random	62.6	57	2..328	2.5	2	1..8
Block	39.8	37	4..141	2.2	2	1..6

The algorithm proves to be quite fast on this OT system, even when no checking of prior winners is performed. For each treatment, a majority of the languages required fewer total learning steps than the square of the number of constraints ($10^2 = 100$). IDL's performance compares quite favorably to the scale of the core hypothesis space (10! = 3,628,800 total rankings). It clearly has escaped the combinatorial growth in number of combinations of interpretations. For the treatments with no prior winner checking, the higher end of the learning step range consists of relatively few outliers. For the (Random / No prior winner checking) treatment, there were only 11 languages that took more than 300 learning steps, and all but one of those were cases where the learner learned more than one grammar for the same overt forms.

Both factors affected the number of learning steps, with both the Block data order and the Prior winner checking levels causing improvement. The number of languages (N = 1,527) leaves little room for doubt of the statistical significance of the differences. Four two-treatment comparisons were performed to verify this, each comparison between treatments that agreed on the level of one factor and differed on the level of the other. Because the same set of languages was used in each treatment, the difference in learning steps between the compared treatments was computed for each language, and then the mean of the differences was computed, thus eliminating cross-language variance (which is considerable). With a null hypothesis that the mean of the differences in learning steps is zero, the Z-values and the p-values for the means tests are given in Table 26.

Table 26

Number of Learning Steps, means test results: N = 1,527

Fixed Level	**Tested Difference**	**Z-value**	**p-value**
No prior winner checking	Random – Block	23.8	$< 10^{-124}$
Prior winner checking	Random – Block	27.6	$< 10^{-167}$
Random	No win check – win check	26.7	$< 10^{-156}$
Block	No win check – win check	21.1	$< 10^{-98}$

Biasing the learner to select shorter forms early (the Block level) has a noticeable effect. Not only is the difference in the number of learning steps statistically significant, but the Block level median number of learning steps is only about 65% of the median number of steps for the corresponding Random level (both with and without prior winner checking). Not surprisingly, the highest performance occurs with the Checking prior winners level / Block level treatment. In fact, in that treatment there were only 14 languages (out of 1,527) on which the learner took more than 100 learning steps.

In even in the most basic treatment (Random / No prior winner checking), for most of the languages the learner never maintained more than 3 simultaneous hypotheses, although there are a minority of languages where the maximum number of simultaneous hypotheses is notably larger. This reflects two things. First, for many of the overt forms with large numbers of possible interpretations, quite a number of those interpretations are not possibly optimal. IDL detected those forms quickly, and discarded them. Second, the overt forms are heavily interrestrictive. A simple combination of two interpretations for one form with two interpretations for another form yields 4 combinations, already more than were ever maintained in a majority of the languages, indicating that most of the time interrestrictiveness has effects as soon as two forms are considered. The learner is able to capitalize immediately upon the interactions between forms imposed by linguistic theory.

Another pronounced effect of checking prior winners is the reduction of the range of the number of simultaneous hypotheses. The high end of the range of the number of simultaneous hypotheses is cut in half, for both data orders. A little extra effort on the part of the learner each time a hypothesis is modified can pay off when inconsistencies are detected earlier.

5. Discussion

5.1. The Role of Optimality Theory

Nothing about IDL is specific to metrical stress. The approach could certainly be applied to other linguistic phenomena. IDL offers a strong approach to a problem endemic to language learning, the structural ambiguity of overt forms. In principle, the IDL approach does not even require that linguistic knowledge be analyzed in Optimality Theoretic terms. What IDL requires are:

- an explicit space of possible grammars.
- an explicit set of possible interpretations for any overt form.
- a method for determining when a set of interpretations is inconsistent.
- a method for finding a grammar that licenses a consistent set of interpretations.

OT makes the space of possible grammars quite explicit: it is the set of grammars that arise from total rankings of the constraints. This explicitness is not notably different from some other approaches to linguistic theory which are typologically rigorous, such as the principles and parameters framework.

However, the other requirements make Optimality Theory particularly well-suited to the IDL approach to learning. Because OT is defined in terms of optimization over a space of candidate structural descriptions, a precise specification of the possible structural descriptions of various linguistic inputs and overt forms is already a central part of the theory. Furthermore, the possible interpretations of a given overt form can be computed using the same type of computational machinery already in place for parsing. Robust interpretive parsing optimizes over the set of interpretations of an overt form with respect to some constraint hierarchy. If an empty constraint hierarchy is used, all candidates "tie", because no violations are assessed. Performing robust interpretive parsing with an empty constraint hierarchy returns all of the tied candidates: it returns all of the interpretations of the overt form[10].

[10] In the computational implementation used in the experiment, this is precisely how the set of possible interpretations for an overt form was computed, retaining all of the computational benefits of dynamic programming as a consequence.

OT's reliance on strict domination among constraints is the key to the guaranteed efficiency of the MRCD algorithm. MRCD works both for determining whether a set of interpretations is inconsistent, and for finding a grammar (a constraint hierarchy) licensing a consistent set of interpretations. The same algorithm satisfies both requirements because the inconsistency detection occurs in the process of attempting to find a constraint hierarchy for a set of interpretations. The upper bound on the number of mark-data pairs applies to both inconsistency detection and grammar finding, and is a direct consequence of strict domination among constraints.

Looked at more broadly, OT provides particularly strong support for an approach to learning that gains information from interaction between different forms, because OT is a linguistic theory based upon interactions between candidates, rather than one based solely on the evaluation of candidates in isolation.

5.2. The Value of Candidate Interpretations

It can be instructive to ask why the learner needs to bother with considering different interpretations of overt forms. Why not just test the compatibility of overt forms with hypotheses directly? After all, that is how error-driven learning has been more traditionally employed: the learner has a hypothesis grammar, attempts to parse an overt form, and if parsing fails, you know the grammar is wrong, and should be changed. But that immediately raises the question: *how* should the grammar be changed? Merely observing that the current grammar is incorrect leaves one with the (exponentially large) set of alternative grammars to choose from in attempting to find one that will parse the overt form. Focusing on different interpretations of the overt form allows one to shrink from a decision space of the possible grammars to a (much, much smaller) decision space of the plausible interpretations of an overt form. A candidate interpretation is quite specific in its requirements of a grammar, enough that the learner can limit the space of possible grammars consistent with the interpretation in a few steps. In the metrical stress system examined here, the space of possible interpretations for an overt form is small enough to be plausibly exhaustively searched. This way, the overt form causing an error can provide much more information than just the fact that the learner's current grammar is incorrect; it can also give strong indications of what the correct grammar must be like, and in a way that can be used in a computationally efficient manner.

It is this shift in the immediate focus of the learner, from grammars to interpretations, that allows IDL to avoid the endless drift in grammar space of the TLA, and the massive search task faced by the STL's superparser. The fact that the space of possible interpretations of an overt form can be accessed using the same parsing mechanism as used for regular parsing obviates the need for extra learning-specific mechanisms, such as the cue searching mechanisms of the cue learner and the superparser of the STL.

With regard to the computation involved, the evaluation of the different candidate interpretations can take place independently; the computation for evaluating one candidate interpretation does not depend upon the outcome of the evaluation of another. This opens the way for parallel computation: the evaluation of the different candidate interpretations can be performed simultaneously. The same applies to the evaluation of a form by multiple hypotheses: the different hypotheses independently interpret the overt form, and those computations could be performed in parallel. As long as the number of hypotheses simultaneously maintained by the learner remains reasonably small, there will not be excessive demands on computational resources.

5.3. The Structure of Hypotheses

Any serious approach to learning must make commitments about the nature of the intermediate information states, or hypotheses, that the learner will have during the course of learning. The structure of a learner's hypotheses will be closely connected to how the learner acts, and can play a big role in the effectiveness of a learner. The structure of the hypotheses used by the OT-based IDL is somewhat unusual, and it may not be immediately obvious how it contributes to the success of the learner. Comparisons with some of the proposals discussed in section 1.2 may be helpful.

In the traditional learnability in the limit framework, the learner always has a single fully specified grammar as a hypothesis. Learning steps consist of transitions from one full grammar to another. This view of learning hypotheses is preserved faithfully in the Triggering Learning Algorithm, which at any time has a setting for each parameter. This straight-forwardly allows error-driven learning, because the learner always has a full specification of parameter settings, so parsing can be applied to new overt forms for acceptance or rejection. However, the learner has no way of knowing which of its parameter settings are supported by previously observed empirical evidence, and which are just arbitrary initial selections. By requiring full parameter settings at all times, the TLA invites this confusion.

To avoid such confusion, the cue learning approach prefers to "set no parameter before its time." At intermediate stages, the learner will have assigned settings to some parameters, while others are unset. Some parameters may have default settings, but it is important to cue learning that the learner always know which parameters in its hypothesis are definitively set. This makes error-driven learning difficult, and cue learners avoid the problem of parsing with some parameters unset by not doing it, abandoning error-driven learning. The price for this is a bunch of other learning-specific processing mechanisms, which take the place of the regular parser in analyzing overt forms for the learner. These cue scanning mechanisms duplicate some of the activities of full-grammar parsers, but are necessarily distinct, designed to be able to function with only certain parameters set.

The Structural Triggers Learner is an attempt to have it both ways, in a sense. The STL is designed so that the intermediate hypotheses held by the learner only have parameter settings justified by data already seen, with the other parameters remaining unset. However, it is envisioned that the STL will still be able to conduct error-driven learning, and to reliably set parameters based upon overt forms. This depends upon two parsing-related claims, the viability of which remain to be seen. The first claim is that it should be possible to parse overt forms using hypotheses with few (initially none) of the parameters set. This is dependent upon the linguistic theory being of an appropriate form, that of each parameter setting being instantiated in a treelet (an actual piece of a structural description). If such a parser can be constructed, it will serve for detecting errors. Setting new parameters in response to errors requires a more ambitious proposal, the superparser. The superparser is a separate mechanism capable of considering all possible settings of all unset parameters in the process of finding a set of parameter settings that will parse the overt form. Further, the superparser must be capable of determining, for all possible interpretations of the overt form, what if any parameter settings are required by all of them, for it is only those settings that may be adopted with confidence. It is not yet known whether such a mechanism can actually be constructed in a form that is at all efficient in operation; that would clearly be a significant achievement. If such a superparser can be constructed, then the STL will successfully be able to combine error-driven learning with hypotheses that only have some parameters set. The adoption of only a single hypothesis at any one time, however, forces the learner to wait indefinitely for overt forms which are unambiguous in their requirements of settings for the unset parameters.

In the OT-based IDL learner presented in this paper, a hypothesis is based upon a consistent set of interpretations (full structural descriptions), and for working purposes is represented by a set of mark-data pairs and a corresponding stratified constraint hierarchy. The mark-data pairs are a specific representation of what the hypothesis is committed to. This avoids the overcommitment problem of the TLA; the learner keeps track of what has been learned from prior data. The stratified hierarchy, resulting from the mark-data pairs, puts the information of the mark-data pairs into a form that can be used for parsing. Stratified hierarchies need not be fully formed grammars (recall that a grammar is fully formed if it follows from at least one total ranking), and during learning they frequently are not; a stratified hierarchy may have two conflicting constraints in the same stratum, where any full grammar would require that one of the constraints dominate the other. Nevertheless, by pooling the marks, the learner is always able to use a stratified hierarchy to parse overt forms. Further, the learner is able to do this with the same optimizing parse mechanism that is used to parse with full grammars. The use of stratified hierarchies in hypotheses

allows the learner to parse overt forms during learning, without having to make premature full grammar commitments, and without needing separate, learning-specific parse mechanisms.

OT-based IDL is able to pull this off because a stratified hierarchy is not a perfect representation of the knowledge in the mark-data pairs of the hypothesis. Any attempt to represent exactly what total rankings are consistent with a set of mark-data pairs will typically result in quite large and disjointed sets of grammars. One cannot uniquely recover the mark-data pairs from a stratified hierarchy. But the stratified hierarchy is a good enough representation of the information in the mark-data pairs to work effectively for error-driven learning. The imperfect nature of the approximation provided by the stratified hierarchy does have an effect under certain circumstances. As explained in section 3.4.1, parsing with mark pooling (MPool) can occasionally miss the fact that a conflict between two constraints in the same stratum needs to be resolved in order to properly complete error-driven learning; this was addressed through occasional use of the CTie interpretation of stratified hierarchies.

Nothing in life is absolutely free. For IDL, the price of basing hypotheses upon consistent sets of interpretations is that sometimes more than one hypothesis must be simultaneously maintained in memory. Uncertainty resulting from unresolved ambiguity of interpretation of overt forms is represented not by introducing further complexity into an individual hypothesis, but by creating different hypotheses for the different interpretations. This allows the learner to benefit from the form of the individual hypotheses without having to prematurely commit to a particular interpretation, as is done in the RIP/CD algorithm. It also allows the learner to proceed with learning in the face of (temporary) unresolved ambiguity, rather than waiting indefinitely for fully unambiguous forms. The risk of this approach is the possibility of the learner maintaining a large number of hypotheses simultaneously. The experimental results suggest that this risk may be quite manageable for spaces of human linguistic grammars.

5.4. Directions for Current and Future Research

In the stress system experiment presented in section 4, the learner exhaustively checked all possible interpretations of an overt form that caused an error on some hypothesis. For this system, the number of interpretations of the overt forms is reasonable enough to permit this. When more complex linguistic systems are attempted, it is likely that the learner won't realistically be able to separately generate and examine all possible interpretations of overt forms, and will instead have to use some means for selecting the most plausible interpretations to try.

One basis for rejecting a candidate interpretation is if the interpretation is not possibly optimal. As explained above, MRCD is guaranteed, when attempting to find a ranking supporting an interpretation, to detect if that interpretation is not possibly optimal. Sometimes, MRCD isn't even necessary: if the interpretation is harmonically bound by a single competitor, and that competitor happens to be selected as the losing candidate for the formation of a mark-data pair, the learner can tell just from the single mark-data pair that the interpretation is doomed (no possible ranking can cause a description to beat a competitor that harmonically bounds it). In the experiment reported here, such cases were still counted as applications of MRCD, so that the reported counts more fully reflected the extent of the generation and evaluation of candidate interpretations of the overt forms, but in practice simple tests of harmonic bounding might significantly reduce the number of interpretations actually subjected to MRCD.

If linguistic systems with larger numbers of interpretations for some overt forms are considered, there may be other efficient means of identifying and eliminating interpretations that are not possibly optimal. See (Samek-Lodovici & Prince, 1999) for interesting and relevant work on the structure of candidate spaces, in particular the relationship of possibly optimal candidates to others.

One phenomenon that will not yield to any simple-minded exhaustive search of possible interpretations is that of underlying forms. Given the principle of richness of the base in Optimality Theory, an overt form can be used to construct candidates with any of an infinite number of possible underlying forms (most of them involving vast deletion of underlying material). This makes exhaustive search out of the question. Fortunately, for any given overt form, the vast majority of 'possible'

underlying forms are easily dismissed and in fact need not be considered at all by the learner; unmotivated deletion is always punished (and not possibly grammatical) by any reasonable OT analysis. Regarding the remaining quite finite space of plausible underlying forms for an overt form, there appear to be general strategies available that may allow the learner to further restrict active consideration to just a few possible underlying forms for each morpheme. The inclusion of underlying forms as another dimension of ambiguity in structural descriptions, and the consequences for learning, are a topic of current research.

Related to underlying forms is the question of grammatical restrictiveness. MRCD is guaranteed to give the learner 'a' constraint hierarchy consistent with all of the data presented to it. However, in some systems it is possible for one grammar to generate a language which is a strict subset of the language generated by another grammar. This arises quite frequently in phonotactic distributions: for example, some grammars may not permit the sound [N] to appear anywhere in the language, others may permit it only in syllable coda positions, and still other grammars may permit it in either syllable onset or coda positions. In the stress system of this paper, every underlying form maps to a distinct overt form (distinct from that of any other underlying form), identifiable by the weights of the syllables in the form, so the concern about more/less restrictive grammars does not arise. Neutralization does not occur in this system. But it could if the system were expanded to include lexical stress systems, where different words with the same syllable weight pattern can have somewhat different stress patterns, or to include the possibility of faithfulness-violating changes to syllable weight, exhibited in phenomena such as iambic lengthening. The possibility of grammars of differing restrictiveness poses a challenge due to a familiar property of language learning: the lack of comprehensive, reliable negative evidence. A child learning a language in which [N] may appear only in syllable codas is not likely to be told this specific fact, and they probably wouldn't know what it meant (consciously) if they were. The learner must infer it from the lack of [N] in syllable onsets in the data they hear. Thus, given a choice from among grammars of differing restrictiveness, the learner must choose the most restrictive grammar consistent with their data. Learning algorithms have been proposed for OT systems that attempt to bias the learner towards the most restrictive grammar; for details, see (Hayes, to appear; Prince & Tesar, to appear; Smolensky, 1996). The incorporation of this kind of learning into an IDL learner is a topic of current research.

Section 5.2 discussed the fact that different grammar hypotheses being simultaneously entertained by the learner can be processed in parallel. Continuing that line of thought, after parallel evaluation the learner may choose to compare the existing hypotheses with respect to various criteria, including restrictiveness; only the most restrictive hypotheses consistent to date would be maintained, with the others being discarded even though they are not strictly inconsistent with the data. This is one way of incorporating a bias towards more restrictive grammars into an IDL learner. This way also makes more apparent the symmetry with respect to Optimality Theory itself. Just as candidate structural descriptions for a linguistic input compete in parallel, so too do candidate grammar hypotheses, with the "optimal" candidate hypothesis (the most restrictive one consistent with the data) being the ultimate goal of the learner.

Other challenging topics not directly addressed here include robustness of learning to input with ungrammatical utterances, and the modeling and acquisition of linguistic variation. These pose challenges for any approach to learning. For IDL in particular, input including ungrammatical utterances could cause the algorithm to eliminate all hypotheses, concluding that no grammar is consistent with all of the data. An approach to this would be to have the learner work toward finding a grammar consistent with the largest possible subset of the input data (another dimension of hypothesis competition). One can imagine several ways to attempt this; formalizing and evaluating them will require significant future research. For a promising approach to dealing with ungrammatical input and variation in the problem of learning OT constraint rankings from full descriptions, see (Boersma, 1998; Boersma & Hayes, 1999).

6. Conclusions

One way or another, a human language learner must combine information from multiple overt forms in order to resolve structural ambiguity and infer the correct grammar. The Inconsistency Detection Learner does this in a particularly direct way, by attempting to construct hypothesis grammars for combinations of interpretations of the overt forms, and discarding those combinations that are shown to be inconsistent. Prior work developed formal results for Constraint Demotion proving that, for Optimality Theoretic systems, inconsistency among a set of full structural descriptions will be detected within a relatively small number of learning steps. The new experimental results presented in this paper show that, for a linguistically reasonable OT system for metrical stress, the IDL algorithm overcomes structural ambiguity without requiring the learner to evaluate more than a reasonable number of sets of structural descriptions. Further, IDL is not defined in a way that is specific to metrical stress, so it is quite reasonable to believe that IDL could be effectively applied to language learning in a variety of linguistic domains, and that this approach may play an important part in the acquisition mechanisms of human learners.

References

Bertolo, S., Broihier, K., Gibson, E., & Wexler, K. (1997a). Characterizing learnability conditions for cue-based learners in parametric language systems, *Fifth Meeting on Mathematics of Language*. Saarbrucken, Germany.

Bertolo, S., Broihier, K., Gibson, E., & Wexler, K. (1997b). Cue-based learners in parametric language systems: Application of general results to a recently proposed learning algorithm based on unambiguous 'superparsing', *Proceedings of the 19th Annual Meeting of the Cognitive Science Society*. Stanford, CA.

Boersma, P. (1998). *Functional Phonology*.: Holland Academic Graphics.

Boersma, P., & Hayes, B. (1999). *Empirical Tests of the Gradual Learning Algorithm* (Ms.): University of Amsterdam and UCLA.

Chomsky, N. (1995). *The Minimalist Program*. Cambridge, MA: MIT Press.

Clark, R. (1992). The Selection of Syntactic Knowledge. *Language Acquisition, 2*, 83-149.

Clark, R., & Roberts, I. (1993). A Computational Model of Language Learnability and Language Change. *Linguistic Inquiry, 24*, 299-345.

Daelemans, W., Gillis, S., & Durieux, G. (1994). The Acquisition of Stress: A Data-Oriented Approach. *Computational Linguistics, 20*(3), 421-451.

Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society B, 39*, 1-38.

Dresher, B. E. (1999). Charting the Learning Path: Cues to Parameter Setting. *Linguistic Inquiry, 30*(1), 27-67.

Dresher, B. E., & Kaye, J. (1990). A Computational Learning Model for Metrical Phonology. *Cognition, 34*, 137-195.

Echeverria, M. S., & Contreras, H. (1965). Araucanian Phonemics. *International Journal of American Linguistics, 31*, 132-135.

Fodor, J. D. (1998). Unambiguous Triggers. *Linguistic Inquiry, 29*(1), 1-36.

Fodor, J. D. (to appear). Parsing to learn. *Journal of Psycholinguistic Research*.

Gibson, E., & Wexler, K. (1994). Triggers. *Linguistic Inquiry, 25*(4), 407-454.

Gold, E. M. (1967). Language Identification in the Limit. *Information and Control, 10*, 447-474.

Gupta, P., & Touretzky, D. (1994). Connectionist Models and Linguistic Theory: Investigations of Stress Systems in Language. *Cognitive Science, 18*(1), 1-50.

Halle, M., & Vergnaud, J.-R. (1987). *An Essay on Stress*. Cambridge: MIT Press.

Hammond, M. (1990). *Deriving Ternarity* (Ms.). Tucson: Linguistics Dept., University of Arizona.
Hayes, B. (1980). *A Metrical Theory of Stress Rules.* Unpublished PhD., MIT, Cambridge.
Hayes, B. (1995). *Metrical Stress Theory: Principles and Case Studies*. Chicago: The University of Chicago Press.
Hayes, B. (to appear). Phonological Acquisition in Optimality Theory: The Early Stages. In J. P. René Kager, and Wim Zonneveld (Ed.), *Fixing Priorities: Constraints in Phonological Acquisition*.: Cambridge University Press.
Joanisse, M., & Curtin, S. (1999). *Dutch Stress Acquisition: OT and Connectionist Approaches* (Ms.): Linguistics Dept., USC.
Kager, R. (1994). *Ternary Rhythm in Alignment Theory* (Ms.): Utrecht University.
Liberman, M., & Prince, A. (1977). On Stress and Linguistic Rhythm. *Linguistic Inquiry, 8*, 249-336.
McCarthy, J., & Prince, A. (1993). Generalized Alignment. In G. Booij & J. Van Marle (Eds.), *Yearbook of Morphology* (pp. 79-154). Dordrecht.
Niyogi, P., & Berwick, R. C. (1996). A Language Learning Model for Finite Parameter Spaces. In M. R. Brent (Ed.), *Computational Approaches to Language Acquisition* (pp. 161-193). Cambridge, MA: MIT Press.
Osborn, H. (1966). Warao I: Phonology and Morphophonemics. *International Journal of American Linguistics, 32*, 108-123.
Pinker, S. (1987). The Bootstrapping Problem in Language Acquisition. In B. MacWhinney (Ed.), *Mechanisms of Language Acquisition*. Hillsdale, NJ: Erlbaum.
Prince, A. (1983). Relating to the Grid. *Linguistic Inquiry, 14*, 19-100.
Prince, A. (1990). Quantitative Consequences of Rhythmic Organization. In K. Deaton & M. Noske & M. Ziolkowski (Eds.), *CLS26-II: Papers from the Parasession on the Syllable in Phonetics and Phonology* (pp. 355-398).
Prince, A. (1999). *A Proposal for the Reformation of Tableaux* (Ms.). New Brunswick: Department of Linguistics, Rutgers University.
Prince, A., & Smolensky, P. (1993). *Optimality Theory: Constraint Interaction in Generative Grammar* (Ms.): Rutgers University and the University of Colorado.
Prince, A., & Tesar, B. (to appear). Learning Phonotactic Distributions. In R. Kager & J. Pater & W. Zonneveld (Eds.), *Fixing Priorities: Constraints in Phonological Acquisition*.: Cambridge University Press.
Pulleyblank, D., & Turkel, W. J. (1998). The Logical Problem of Language Acquisition in Optimality Theory. In P. Barbosa & D. Fox & P. Hagstrom & M. J. McGinnis & D. Pesetsky (Eds.), *Is the Best Good Enough? Optimality and Competition in Syntax* (pp. 399-420). Cambridge, MA: MIT Press.
Rubach, J., & Booij, B. E. (1985). A grid theory of stress in Polish. *Lingua, 66*, 281-319.
Sakas, W., & Fodor, J. D. (in press). The structural triggers learner. In S. Bertolo (Ed.), *Parametric Linguistics and Learnability: A Self-Contained Tutorial for Linguists*. Cambridge, UK: Cambridge University Press.
Samek-Lodovici, V., & Prince, A. (1999). *Optima* (Rutgers Center for Cognitive Science Technical Report RuCCS TR-57). New Brunswick: Rutgers University.
Smolensky, P. (1996). *The Initial State and "Richness of the Base" in Optimality Theory* (Technical Report JHU-CogSci-96-4). Baltimore, MD: The Johns Hopkins University.
Tesar, B. (1995). *Computational Optimality Theory.* Unpublished PhD., University of Colorado, Boulder, CO.
Tesar, B. (1997). *Multi-Recursive Constraint Demotion* (Ms.). New Brunswick, NJ: Linguistics Dept., Rutgers University.

Tesar, B. (1998a). Error-Driven Learning in Optimality Theory via the Efficient Computation of Optimal Forms. In P. Barbosa & D. Fox & P. Hagstrom & M. J. McGinnis & D. Pesetsky (Eds.), *Is the Best Good Enough? Optimality and Competition in Syntax* (pp. 421-435). Cambridge, MA: MIT Press.

Tesar, B. (1998b). An Iterative Strategy for Language Learning. *Lingua, 104*, 131-145.

Tesar, B. (1998c). Robust Interpretive Parsing in Metrical Stress Theory. In K. Shahin & S. Blake & E.-S. Kim (Eds.), *The Seventeenth West Coast Conference on Formal Linguistics* (pp. 625-639): CSLI.

Tesar, B., & Smolensky, P. (1994). The Learnability of Optimality Theory. In R. Aranovich & W. Byrne & S. Preuss & M. Senturia (Eds.), *The Thirteenth West Coast Conference on Formal Linguistics* (pp. 122-137): CSLI.

Tesar, B., & Smolensky, P. (1998). Learnability in Optimality Theory. *Linguistic Inquiry, 29*(2), 229-268.

Tesar, B., & Smolensky, P. (2000). *Learnability in Optimality Theory*. Cambridge, MA: MIT Press.

Wexler, K., & Culicover, P. (1980). *Formal Principles of Language Acquisition*. Cambridge, MA: MIT Press.