Discovering Underlying Forms: Contrast Pairs and Ranking

by

Nazarré Nathaniel Merchant

A Dissertation submitted to the

Graduate School-New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Linguistics

written under the direction of

Bruce Tesar

and approved by

_________Alan Prince________

_________Paul de Lacy______

_________Joe Pater__________

_________Bruce Tesar______

New Brunswick, New Jersey

May, 2008

ABSTRACT OF THE DISSERTATION

Discovering Underlying Forms: Contrast Pairs and Ranking

By Nazarré Nathaniel Merchant

Dissertation Director:
Bruce Tesar

Phonological learners must acquire a lexicon of underlying forms and a constraint ranking. These must be acquired simultaneously, as the ranking and the underlying forms are interdependent. Exhaustive search of all possible lexica is intractable; the space of lexica is simply too large. Searching the underlying forms for each overt form in isolation poses other problems. A single overt form is often highly ambiguous among both underlying forms and rankings. In this dissertation I propose a learning algorithm that attends to pairs of overt forms that differ in exactly one morpheme. These pairs can exhibit less ambiguity than the isolated overt forms, while still providing a reduced search space.

The algorithm first assigns underlying values to occurrences of features whose surface realization never alternates; the other underlying features are left initially unset (Tesar et al., 2003). Pairs of overt forms that differ in one morpheme are then constructed. The algorithm then considers the possible values of unset features for each pair, processing pairs with the fewest unset features first. It uses inconsistency detection (Tesar, 1997) to test sets of values of unset features for viability. A set of values for the unset features is viable if it produces the correct overt forms under some ranking. Those feature values which are common across all viable solutions are then set. In the process of testing for inconsistency for each set of values of unset features a set of winner-loser pairs is generated. The learner determines the ranking restrictions jointly entailed by these sets of winner-loser pairs. These ranking restrictions are then maintained while processing all further contrast pairs. After all pairs have been processed, any still unset feature values are assigned default values. The general success of the algorithm depends upon these features being fully predictable in the output. A ranking is then obtained from this lexicon using Biased Constraint Demotion (Prince and Tesar, 2004).

Fixing all non-alternating features reduces the effective lexical search space. The algorithm further reduces the lexical search space by breaking up the search into tractable local pair searches. Extracting shared ranking information from winner-loser pairs generated from inconsistency detection restricts which featural combinations for future contrast pairs will be viable providing information that is otherwise unavailable to the learner.

Acknowledgements

The Rutgers Linguistics department, and in particular its phonology folk, have created a wonderful environment for pursuing linguistics, one in which I feel myself quite lucky to have participated. People's overall enthusiasm for their research and the encouragement of and interest in others' research contributed greatly to my enjoyment of my time in New Brunswick. It's a great intellectual environment.

I'm grateful to my committee, Bruce Tesar, Alan Prince, Paul de Lacy, and Joe Pater for their regular encouragement and enthusiasm for my research. In particular I'd like to thank Bruce Tesar and Alan Prince. Bruce's unflagging interest and steady guidance over the years provided me with the environment that allowed me to write this. It wouldn't have happened without him. I'm also greatly appreciative of Alan's contributions to and encouragements of this work. And going beyond the linguistic pursuits, I feel fortunate to have been exposed to his general spirit of inquiry. It's infectious.

Jane Grimshaw, Graham Horwood, Judy Bauer, Chi Luu, and Jessica Rett all contributed greatly to my enjoyment of my time at Rutgers. As for time spent in round one, I'm indebted to Jessica Sklar, Inga Johnson, and Katherine Brandl. They helped here more than they know. And thanks to Sarah Bray for helping in innumerable ways during those critical times when the work was getting done.

Table of Contents

List of Tables		vii
1	Contrast pairs and local lexica	1
2	Using contrast pairs and local lexica	26
3	Limits of local lexica	49
4	The Join	78
5	Ranking extraction applied	119
6	Palauan and implications about the lexicon	156
7	Comparison and conclusions	176
References		185
Curriculum Vita		188

List of Tables

1.7	Comparative tableau for the two overt forms ta.**ká:** and **tá:.**ka	20
1.8	Winner-loser pairs BCD will produce a hierarchy from	21
1.9	Winner-loser pairs and constraints remaining	21
1.11	Winner-loser pair produced from MRCD	23
1.13	Winner-loser pair produced from MRCD	24
2.9	Mapping of morphemes	36
2.10	Lexicon after initial lexical assignment	38
2.11	Results of error-driven learning on **tá**ka and **dáa**ka	39
2.12	Two local lexica for contrast pair **pá**ka ~ **dáaka**	41
2.15	Consistency check for two local lexica of pair **pá**ka ~ **dáaka**	41
2.16	ERCs from step 1	43
2.18	Winner-loser pairs from error-driven learning on the pair pa**sá** ~ **tá**sa	43
2.19	Inconsistent ERCs from error-driven learning	44
2.20	Results of error-driven learning on all four local lexica	44
3.11	ML dominates MR and Id(s) dominates ML	54
3.12	Non-stressed syllables do not surface as aspirated or long	55
3.13	Id(l) >> NoAsp, *V:, Id(a)	56
3.14	*[+s, -l, -a] >> NoAsp, Id(a), and Id(l) is dominated by Id(s)	56
3.15	Mappings of Language L1	57
3.16	Initial lexicon after the initial lexical assignment stage	58
3.17	Lexicon after contrast pair processing stage	59
3.24	Mappings of L2	63
3.27	Mappings of morphemes in L1	66
3.28	Mappings of morphemes in L2	66

3.29 URs of the suffixes in L1 after the processing of contrast pairs 70
3.30 Ranking information gained from previously unattested mapping 71
3.31 Set features of morphemes in contrast pair **<báa**ka, **t**^{h}**á**ka**>** 74
4.2 Fusion of two ERCs 81
4.4 ERC Set 1 84
4.5 ERC Set 2 84
4.6 Contradictory ranking requirements 85
4.10 Join of two ERCs 88
4.11 ERC Set 1 91
4.12 ERC Set 2 91
4.13 The join of all four ERCs 91
4.15 ERC Set 1 95
4.16 ERC Set 2 95
4.17 ERC Set 1 Result of joining ERC 1 and ERC3 together 96
4.18 Result of joining ERC 1 and ERC3 together 96
4.19 ERC Set 1 98
4.20 ERC Set 2 98
4.21 Joining ERCs containing an L in constraint C3 99
4.22 Algorithm applied to ERC Sets 1 and 2 100
4.23 ERC Set 1 102
4.24 ERC Set 2 102
4.25 The join of ERC 1 and 3 103
4.26 The fusion of ERC 1 and 2 104
4.27 ERC Set 1 105
4.28 ERC Set 2 105
4.29 Result of algorithm on ERC Set 1 and 2 105
4.30 ERC Set 1 108

4.31 Fusional closure of ERC Set 1 108

4.32 ERC Set 2 109

4.33 Pair-wise fusing of ERCs in ERC Set 2 110

4.34 ERC Set 1 112

4.35 ERC Set 2 112

4.36 Fusional closure of ERC Set 1 113

4.37 Fusional closure of ERC Set 2 113

4.38 Result of second stage on fusional closures 113

4.39 ERC Set 1 115

4.40 ERC Set 2 115

4.41 Globally true ranking restriction 115

4.42 Fusional closure of ERC Set 1 and global store 116

4.43 Fusional closure of ERC Set 2 and global store 116

5.11 Mappings of language 121

5.13 Ranking restrictions of overt forms **r**^{h}**á**sa, **ráa**sa, ra**s**^{h}**á**, and ra**sáa** 122

5.16 Mappings of language 130

5.17 Ranking restrictions of overt forms **r**^{h}**á**sa, **ráa**sa, ra**s**^{h}**á**, and ra**sáa** 132

5.18 Mappings of language 133

5.19 Lexicon after initial lexical construction stage 134

5.21 Four contrast pairs with two unset features 137

5.22 Local lexica for contrast pair paka ~ dáaka 137

5.23 Lexicon after processing of pairs without using shared ranking information 138

5.24 Lexicon after processing of pairs without using shared ranking information 139

5.25 Fixed lexical specifications for contrast pair 140

5.26 Local lexica for contrast pair 140

5.28 Winner-loser pair generated from first pass of error-driven learning 141

5.30 Winner-loser pair generated from parsing the second overt form 142

5.32 Resulting ERCs from error-driven learning on local lexicon two 143

5.34 Resulting ERCs from error-driven learning on local lexicon three 144

5.36 Resulting ERCs from error-driven learning on local lexicon four 144

5.38 ERC set 1 from local lexicon 1 145

5.39 ERC set 2 from local lexicon 2 145

5.40 ERC set 3 from local lexicon 3 145

5.41 ERC set 4 from local lexicon 4 145

5.42 Fusional closure of Set 1 146

5.43 Fusional closure of Set 2 146

5.44 Fusional closure of Set 3 146

5.45 Fusional closure of Set 4 146

5.46 Domination information for *V: 147

5.47 Domination information for MR 147

5.48 Domination information for Id(a) 147

5.49 Ranking entailments from the local lexica 148

5.50 Currently set lexical specifications for contrast pair **báa**ka ~ bas**^{h}á** 150

5.51 Local lexica for contrast pair **báa**ka ~ bas**^{h}á** 150

5.52 Ranking entailments from the local lexica and phonotactics 151

5.54 ERC created from incorrect parsing of /baaka/ 152

5.55 The inconsistency of ERCs 2, 5 and 6 152

6.1 Palauan stress placement 157

6.2 Stressed vowels: o, e, i, a, u, and unstressed ə 158

6.3 Root ‘cover opening’ with three affixes 159

6.4 Stress position does not predict vowel quality 160

6.5 Underlying forms for the four roots above 161

6.15 Mappings of morphemes in this language 166

6.18 Mappings of morphemes in this language 169

6.19 Results of feature assignment to non-alternating features 170

6.20 Underlying values after the processing of contrast pairs 171

6.21 Final lexicon 173

7.2 Mappings of morphemes in L1 178

Chapter 1. Contrast pairs and local lexica

To define a language optimality theoretic grammars require a ranking of the constraints, a set of underlying forms that the grammar acts upon, and a morphology for combining underlying forms into inputs. Presupposing a given morphology, a learner must determine both what the ranking is and what the underlying forms are to be said to have learned a language. These two learning tasks are intertwined: change the ranking and different underlying forms may be needed to produce the correct outputs; change the underlying forms and a different ranking may be needed to produce the correct outputs. These tasks cannot be done in isolation and because they are interdependent, information about the ranking and lexicon must be determined concomitantly. This dissertation will propose and investigate the properties of an algorithm that extracts ranking and lexical information from the overt forms of the language.

Besides the basic imperative of finding a correct grammar by learning the lexicon and ranking, any learning algorithm must be concerned with efficiency. The learner must successfully search a large space for a correct lexicon and ranking that will produce the overt forms of the language and do so in a reasonable amount of time. A naïve approach to learning is an exhaustive search. The learner checks each lexicon and ranking pair for correctness. Does a given lexicon and ranking produce the correct overt forms? If not, consider the next pair. Under realistic conditions this is hopelessly inefficient. The learner must focus its limited computational resources on an algorithm that is guaranteed to succeed and to do so quickly. To this end I propose that the learner restricts itself to focusing not on all of the forms of the language at once but on *contrast pairs* (Alderete et al. 2005, Tesar 2004), pairs of overt forms that differ in one morpheme. Contrast pairs will prove to serve two purposes: they will be informative and computationally tractable. By focusing on contrast pairs the learner will be provided information about both the lexicon and the ranking and this information will be able to be extracted efficiently.

This dissertation explores and articulates novel ways of learning underlying forms by focusing on computationally tractable sets of underlying forms, specifically contrast pairs. The research presented here demonstrates new ways of learning both the underlying forms of the language and the ranking that produces the overt forms of the language the learner is attempting to learn. This new approach, encompassed by the algorithm that I propose called the Contrast Pair and Ranking information algorithm (CPR), is capable of producing a correct lexicon and ranking in computationally feasible amounts of time from a learning state that includes no knowledge of the lexicon or the ranking.

This chapter will discuss the assumptions of the learning algorithm along with contrast pairs and local lexica. Chapter 2 will present a learning algorithm that focuses on extracting lexical information from contrast pairs. Chapter 3 will show the limitations of only extracting lexical information from contrast pairs. Chapter 4 will propose a method of determining shared ranking information across consistent local lexica. Chapter 5 will present a learning algorithm that both extracts lexical information and ranking information from contrast pairs. And finally, Chapter 6 will discuss some implications of this approach and languages that support it.

Section 1.1. Learning assumptions

Grammatical system assumptions

I assume a range of Optimality Theoretic systems (Prince and Smolensky 1993) with a limited typology of constraints. Constraints are assumed to fall into one of two classes, markedness constraints which only reference the output form of the candidate and faithfulness constraints which are limited to only ident-type constraints (McCarthy and Prince 1995). The restriction on the faithfulness constraints follows Tesar 2004 and is severely limiting. No Max or Dep constraints (McCarthy and Prince 1995) are allowed in the grammar and along with this prohibition on the type of constraints is the necessary restriction on Gen: it does not produce candidates with inserted or deleted segments. All candidates are segmentally identical to the input form. The faithfulness constraints are then restricted to the ident-type constraints. Only changes to the featural specification of an input candidate are allowed (and so no deletions, insertions, metathesis, nor coalescence are permitted). Though these are not realistic assumptions about naturally occurring grammars they serves two purposes.

First, they simplify the linguistic system so that the application of the learning algorithm presented here can be fully analyzed. By excluding insertion and deletion we are able to fully characterize both the linguistic system and the learning algorithm applied to the languages in the linguistic system. The application of the learning algorithm and our understanding of it rely on the learner having knowledge of the correct correspondences between both input and output segments and between output segments in different overt forms that share a morpheme. Inclusion of insertion and deletion in the assumptions about the linguistic system complicates the assignment of correct correspondences between input and output and between output and output segments. Though determination of input-output and output-output correspondences is a task the learner must accomplish, most likely concomitantly with determination of the lexicon and constraint ranking, by removing Max and Dep constraints (and insertion and deletion) we can focus on two aspects of the learning algorithm, lexicon specification and constraint ranking.

Second, even though I make the simplifying assumption that the learner does not insert nor delete segments and is given correspondences between segments the linguistic system is still complex enough to exhibit patterns that are attested in natural languages. In the linguistic systems presented here we find a rich enough system to exhibit contrast, alternating forms, ambiguous languages, and ambiguous underlying forms. In this simplified system some of the issues that arise from these types of complex behaviors can be solved. By addressing the issues in a less complex system we may gain insight into their behavior in more complex environments.

Before any learning occurs the learner is assumed to have full knowledge of the constraints, though no knowledge of the ranking of the constraints. Crucial to this learning approach is that the learner has access to all of the constraints of the language at all stages of learning; learning cannot proceed here unless the learner has the ability to construct a full grammar of the language. At any stage of learning, the learner can re-rank the constraints based on new information gained during the execution of the learning algorithm. The re-ranking of constraints need not be incremental, meaning that re-ranking does not need to only modify the relative ranking of two constraints; one new piece of information can yield a significantly different hierarchy from the previously postulated grammar. This type of potential complete re-ranking is accomplished using Biased Constraint Demotion (BCD) (Prince and Tesar 1999, see also Hayes 2004). A detailed discussion of BCD is given at the end of this chapter but importantly BCD has the property that given a hierarchy that produces a given set of input-output mappings, if the learner is presented with one new input-output mapping, BCD applied to the new (and old) input-output mappings may produce a significantly different ranking. There is no guarantee that only two constraints will be re-ranked in the application of BCD to one new form.

The overt forms of the language

I assume that the learner initially has access to all of the overt forms of the language. These overt forms are the words of the language and may be composed of more than one morpheme. Prior to any learning the learner is aware of the morphemic decomposition of the overt forms of the language. The learner is aware not just of the morphological categories of the language and the segmentation of the overt forms but of the morphemic identities of the morphemes in the overt forms of the language. The learner knows all of the morphemes of the language and their corresponding allomorphs as they appear in the overt forms of the language. This information comes to the learner by fiat; no learning occurs to determine morphemic identity. The learner just knows. How the learner determines them is beyond the scope of this dissertation but an issue that must be addressed. So, while the learner knows the morphemes of the language, it does not know what the underlying specifications for a particular morpheme or overt form are nor does the learner know what the ranking of the constraints are, that is, the learner does not know the grammar that maps the given (unset) underlying forms to the given overt forms.

The assumption that the learner obtains the morphemic decomposition of the overt forms of the language independent of learning the underlying forms and ranking is untenable for a comprehensive theory of learning. Possible morphemic composition of the overt forms of a language is affected by the grammar of the language and the underlying forms of the language and vice versa. Learning the morphemes of a language is intertwined with the learning of the grammar and the underlying forms. Even so, isolating the learning of underlying forms and the grammar from the learning of morphemic decomposition yields information about the nature of the learning problem. This in turn may give insight into the intertwined problems of learning the morphemic decomposition of words, their underlying forms, and the grammar that produces their overt forms.

The lexicon

The lexicon in this learning algorithm consists of a set of morphemes each of which is identified with a set of segments in the overt forms (the morpheme’s allomorphs) and with an underlying phonological specification. So morphemes are basic primitives in the lexicon and have an associated set of allomorphs and a underlying phonological representation. Prior to learning the learner is given the identity of the morphemes of the language and each morpheme's set of allomorphs but no information regarding the underlying phonological specifications of those morphemes. Throughout the learning algorithm underlying phonological forms for morphemes may remain unset with respect to some or all phonological features. Learning the lexicon in this situation means setting all phonological features for all of the morphemes. After all learning stages are completed the learner must have set each feature for each morpheme. A learned lexicon does not allow unset features.

The ban on unset features is a restriction on the final lexicon. Unset features are allowed throughout the learning stages. This is not to say that underspecified features are allowed. Underspecification is a distinct theoretic notion from unset features in the lexicon. I assume that underspecified features do not exist in the learned lexicon nor do they occur in the lexicon during learning. Underspecification is not allowed in this linguistic system. Features are allowed during learning to be unset, meaning that the learner does not know what the feature value is and must determine a value before the learning is said to be completed. Though I do not allow underspecification the results here could be extended to a system that does allow it. Underspecification of a feature can be viewed as simply another feature value for that feature; for example, in a system that allows underspecification, a vowel may be specified as long, short, or unspecified for length. In effect, length becomes a three-valued feature with underspecification. During learning, features initially are unset and learning a feature value for a morpheme in a system that allows underspecification would mean determining the value for that feature or whether it is underspecified in the learned lexicon. So with regards to the length example above, initially a vowel would be unset and the learner must determine whether the vowel is short, long, or unspecified for length. It is important to note that in the learning algorithm presented here an unset feature means an *unknown* feature value for the learner.

Section 1.2.1 New and not new

Learning underlying forms and the grammar that produces a language is not a new question and research continues on this topic today. A select subset of recent research includes Pater (to appear), Jarosz 2006, McCarthy 2005, Bermudez-Otero 2003, Albright 2002, Tesar et al. 2003, Tesar 2004 and Apoussidou 2007 each of whose approach either uses a different set of basic assumptions from the work pursued here or differs substantively in focus.

Pater's work uses inconsistency detection to create new constraints to account for lexical exceptionalism. His focus is on exceptional forms using an approach not incompatible with what is presented here. Jarosz posits a function that assigns probabilities to different grammars and a probability maximizing function that applies to the probabilistic form of Optimality Theory she assumes. These assumptions of the underlying grammar fundamentally differ from those assumed here. Bermudez-Otero uses a stratal OT theory and a technique of isolating forms that give rise to opaque behavior from the learner while the learner is attempting to learn the "core" grammar. This theory, while different from the one posited here and focusing on a different learning issue (that of opacity) is not necessarily incompatible with the theory given in this dissertation. Determination and isolation (until a later stage of learning) of forms that give rise to opaque phenomena could be a useful extension of the learning algorithm presenting herein. McCarthy also focuses on forms that yield opaque phenomena. His approach makes the assumption that non-alternating forms may have an underlying form that differs from its surface representation – an assumption at variance with the approach taken here. Apoussidou while looking at similar questions as those raised in this dissertation focuses on comparing the Gradual Learning Algorithm (GLA) (Boersma and Hayes 2001) and Constraint Demotion (CD) (Tesar 1995) to learn the lexicon and constraint ranking. Her focus on ambiguous structure and the use of GLA differentiates her approach from the one taken here. The Surgery Learning Algorithm of Tesar et al. while adopting a similar linguistic framework restricts the learner to single changes in the lexicon and allows the learner to set features incorrectly without knowledge of the locus of error, unlike the approach taken here. As discussed in ch. 7 below the Contrast Analysis approach of Tesar 2004 uses similar assumptions about the linguistic theory of the learner, though as will be shown this approach generalizes the capabilities of the Contrast Analysis learning procedure. Albright's restricted model of UR discovery (Albright 2002) requires that the underlying form of a morpheme match one of its surface realizations. This algorithm does not make that assumption and as discussed in ch. 6 languages like Palauan demonstrate that this is an assumption that should not be made.

While the approaches listed above use variously different assumptions in their theories there is a set of theories and tools that this dissertation relies fundamentally on and builds directly upon. Basic to this theory (and many of the above theories) is error-driven learning (Wexler & Culicover, 1980). The form that error-driven learning takes here is that of Multi-Recursive Constraint Demotion (MRCD) (Tesar 1997). In MRCD, for a given underlying form and its corresponding overt form, the underlying form is parsed using a given initial hierarchy. If the produced overt form is not the target overt from, an error has occurred and learning proceeds by the creation of a winner-loser pair (Tesar et. al. 2003) and the application BCD (Prince and Tesar 2004) to the resultant pair (along with all other previously created winner-loser pairs). In this manner a hierarchy is produced that generates the overt forms of the language.[1]

This dissertation proposes the novel application of this form of error-driven learning to a particular set of underlying forms, called local lexica (described below), for a contrast pair (Tesar 2004). This new approach to searching the lexical and ranking space will prove to greatly reduce the search space while providing useful information to the learner. In addition, this dissertation provides a new mechanism, called the join (defined in Chapter 4), for determining ranking information from ERCs (Elementary Ranking

[1] In this dissertation all applications of MRCD and error-driven learning were calculated by hand. While some work has been done to automate the production-directed parsing needed to use MRCD, the linguistic systems here have not been implemented in such automated systems.

Conditions) (Prince 2002) in the face of uncertainty. When a learner is presented with two ERCs one of which is true but not necessarily both and the learner is uncertain of which is true, the join operator will produce that ERC that captures the ranking information that both share. This join operator is then used to produce a new mechanism for extracting all ranking information from a collection of sets of ERCs, again, in the face of uncertainty. This procedure relies on modifying the given ERC sets using fusion (Prince 2002), a binary operation on ERCs that produces an ERC that is entailed by each individual ERC (Prince 2002). The mechanism proposed in this dissertation for extracting shared ranking information is maximally informative producing a set of ERCs that captures all shared ranking information.

Section 1.3 The stress-length linguistic system

The above assumptions are restrictions on linguistic systems that ensure that the methods presented below apply. In this section I discuss one such linguistic system, the stress-length system, and in the next chapter explore the consequences of applying the learning algorithm to languages in this system. Overt forms are severely restricted in both features and segmental composition here. This linguistic system only allows two features, stress and length, each of which is a binary valued feature. Each overt form consists exactly of one root morpheme and one suffix morpheme. Root and suffix morphemes also are restricted in length to exactly one syllable. So for any overt form in any language in this system there are at most sixteen possible underlying representations for the given overt form; the root may be short and unstressed, short and stressed, long and unstressed, and long and stressed; the suffix may also be underlyingly one of these four combinations and consequently, the overt form itself may come from any one of the sixteen combinations of the root and suffix. Given the set of constraints and features in this linguistic system, for any given language there can be at most four distinct roots and four distinct suffixes. For this reason, I restrict the number of morphemes in the lexicon for a given language to at most four roots and four suffixes.

For each of the two features there is a faithfulness constraint, Ident(stress) and Ident(length) (McCarthy & Prince 1995). The Ident(stress) constraint states that a stress feature on the surface must be identical to its underlying stress and the Ident(length) constraint states that a length feature on the surface must be identical to its underlying length. There are also four markedness constraints in this system. Two refer only to the placement of stress, MainLeft and MainRight (McCarthy & Prince 1993). MainLeft prefers stress to be on the leftmost syllable while MainRight prefers stress on the rightmost syllable.[2] One constraint disprefers long vowels, *V: (Rosenthall 1994), and the final constraint Weight-to-Stress principle (WSP) (Prince 1990) disprefers long, unstressed vowels. All the constraints are listed below.

1.	Ident(stress)	The stress value must be identical to its input correspondent
2.	Ident(length)	The length value must be identical to its input correspondent
3.	MainLeft	Stress must fall on the leftmost syllable
4.	MainRight	Stress must fall on the rightmost syllable
5.	*V:	Do not have a long vowel
6.	WSP	If a vowel is long it must be stressed

[2] Because stress in this idealized system falls on a syllable and forms are always bi-syllabic gradient evaluation considerations are obviated.

The language learner however does not attempt to learn a linguistic system; the assumption here is that knowledge of the linguistic system is innate. That is, the learner knows all of the constraints and, furthermore, is given the morphemic decomposition of the overt forms for the language the learner is attempting to acquire. Given the linguistic system and the overt forms of the language along with their morphological composition the learn must determine the ranking and the underlying forms of the language.

Section 1.4 Contrast pairs

Learning a language is a search problem. The learner must determine what the underlying forms of the language are and what the ranking of the language is; in effect the learner must search the space of possible lexica and rankings to find the correct grammar for the given overt forms the learner is presented with. Considering every possible lexicon and ranking combination is an untenable search strategy. Consider a simple language consisting of 15 constraints and 100 possible lexica. There are 15! x 100 $\approx$ 1.3 x 10^{14} possible grammars. Were the learner to consider each possible grammar in succession taking one second to evaluate any given grammar it would take only slightly over 4 million years to evaluate all of them. Clearly exhaustive search of the possible grammars cannot work in any sort of feasible time frame.

Some of the size of the search space comes from the combination of considering both possible lexica and possible rankings in combination. While a reduction in the search space size obtains by considering lexica and rankings in isolation it is not a tenable search strategy. A lexicon is only consistent with a set of overt forms given a particular ranking. And similarly a ranking is only consistent with a set of overt forms given a particular lexicon. This interdependence precludes searching the lexical hypothesis space independently of the ranking space (and vice versa). The learner must consider lexica and rankings together.

Even though lexica and rankings together must be tested for consistency it is not necessary for the learner to consider an entire lexicon in conjunction with a given ranking to determine information about the target grammar. The learner is able to extract ranking and lexical information by considering single overt forms and the possible lexica and rankings that produce that overt form. This is the type of approach used in the Prince & Tesar (1999). While able to extract some information it is not able to determine an entire grammar in most situations. A single form is too impoverished to yield the types of information needed to for the learner to learn the language. The learner must focus on a larger unit of data.

I propose that the learner uses contrast pairs to extract lexical and ranking information from the overt forms guiding the learner through the search space in an efficient manner.

Contrast pairs are pairs of overt forms that differ in one morpheme and contrast on some phonological feature. In English the word ‘cats’ ([kæts]) and the word ‘dogs’ ([dɔgz]) form a contrast pair. They both contain the plural morpheme, though with different surface realizations of that morpheme, and they differ on their root morphemes. They also contrast on the surface both in their root and suffix. Note that contrast pairs need not contrast minimally. The roots of these two forms differ on many features.

Contrast pairs will prove to be useful for two reasons: extracting information from them can vastly reduce the search space and they can be informative. Searching the entire lexical space for the correct set of underlying forms can be prohibitively time-consuming. Turning to the linguistic system presented above, a language of the system may have 8 morphemes in its lexicon (4 roots and 4 suffixes). Each of these morphemes can have one of four possible underlying representations (there are 2 binary features and each morpheme is specified for each of the features). This leads to a lexical space of size 4^8 = 65,536. If the learner attends only to contrast pairs and their lexica, a greatly reduced search space is encountered. Since each overt from consists of two morphemes and a contrast pair shares a morpheme across the overt forms, each contrast pair consists of 3 distinct morphemes. These 3 morphemes each may have 4 potential underlying forms leading to a lexical space to search of size 4^3 = 64, significantly smaller than the total space of 65,536. If information about the lexicon and ranking can be extracted from contrast pairs efficiently and effectively, this can lead to a great reduction in the lexical search.

In fact, information about both the lexicon and the ranking can be extracted from contrast pairs (Tesar 2006, Merchant and Tesar 2006b). Returning to the contrast pair ‘cats’ and ‘cads’, the plural morpheme surfaces differently in the morphological environment ‘cat’ than it does in the environment ‘cad’. Because the plural has only one underlying specification the difference in its surface representation comes about because of the grammar. The ranking of the constraints determines that the plural morpheme surfaces differently in these two environments and this contrast pair requires some grammatical account of contrast. Furthermore, because these two forms differ in only one morpheme (namely their root morphemes) and they differ on their surface we can determine the locus of lexical difference. The morphemes ‘cat’ and ‘cad’ must be different in their underlying specifications or else they could not differ on the surface in this contrast pair. Ranking information is needed to explain the different surface allomorphs of the plural morpheme and lexical information is needed to explain the different surface representations of the root morphemes. Both types of information are needed to explain the surface contrast of these two overt forms and importantly both types of information are extractable from contrast pairs.

Section 1.5 Local lexica

The extraction of lexical information and ranking information from a contrast pair stems from the investigation of possible underlying forms for the contrast pair in question. During learning the morphemes of a contrast pair may have some feature values that are set and some that are unset. A full assignment of values to the features of the morphemes of the contrast pair is called a *local lexicon* for the contrast pair (Merchant & Tesar 2006a). A local lexicon for a contrast pair is a hypothesis about what the underlying specifications of the morphemes of the contrast pair are. The collection of all possible assignments of underlying values to the unset features of a contrast pair constitutes the local lexica for that contrast pair.

Some local lexica for a contrast pair will be untenable. That is, a local lexicon for a contrast pair is a hypothesis about what the underlying lexicon is. Since the set of local lexica for a contrast pair consists of all possible hypotheses about the underlying lexicon some of the hypotheses may be incorrect. A necessary condition for a local lexicon to be the correct lexicon, though not sufficient, is for there to exist a ranking that maps the underlying forms with the local lexicon's specifications to the given overt forms. For some local lexica no ranking will map the underlying forms to the given overt forms. It is through testing local lexica for a given overt form that information about the lexicon and ranking will be extracted from contrast pairs.

Suppose the learner is attempting to acquire a language from the linguistic system given above. This system has two features and has overt forms consisting of a mono-segmental root and suffix. The learner encounters the two overt forms **dáa**.ka and ta.**ká** which constitute a contrast pair. The first overt form consists of a long, stressed root and a short, unstressed suffix while the second overt form consists of a short, unstressed root and a short, stressed suffix. For discernability reasons each syllable (which corresponds directly to a morpheme, either a root or suffix) is introduced by a consonant that represents morphemic identity. So these two overt forms share a suffix, namely the 'ka' suffix (though 'ka' surfaces unstressed in the first overt form and stressed in the second) and they differ on their roots, the first overt form having the root 'daa' and the second having ‘ta’. Because the two overt forms differ on one morpheme and surface differently they constitute a contrast pair. For this contrast pair, if all features of the three morphemes are unset there are $4^3 = 64$ local lexica for this contrast pair. The learner must determine which of the local lexica are consistent local lexica. That is, for which of the local lexica is there a ranking that maps those underlying forms with the local lexicon to the given overt forms. This determination of consistency must be done efficiently because it forms the crucial computational step that is repeated throughout the proposed algorithm. The learner uses biased constraint demotion (BCD) with multi-recursive constraint demotion (MRCD) (Tesar 1997) to determine the consistency of the local lexica, an algorithm guaranteed to produce a correct determination of consistency for each local lexicon.

Consistent local lexica for a given contrast pair may be used to correctly set underlying features of morphemes in the contrast pair and to determine ranking information about the target language. The issue faced by the learner though is that it does not know which local lexicon is correct; accepting ranking restrictions imposed by a consistent but incorrect local lexicon could yield incorrect information about the target language. This issue can be circumvented by determining what the ranking restrictions are that are shared across all the consistent local lexica. In this way the learner can determine ranking restrictions from the contrast pair. The learner can also extract lexical information from the consistent local lexica. If a feature value for a given morpheme has the same value in all consistent local lexica the learner knows that this is the correct value for that feature. Having the same value across consistent local lexica means having the same value as the correct lexicon because the correct lexicon *is* a consistent local lexicon.

Section 1.5 Inconsistency detection

Here I discuss how MRCD determines whether a given local lexicon is consistent by investigating its application to the contrast pair **dáa**.ka and ta.**ká**. Assuming that no features for the three morphemes are specified there are $4^3 = 64$ local lexica for this contrast pair. Consider the local lexicon that has each morpheme underlyingly unstressed and short. This local lexicon will be inconsistent for this contrast pair. This is clear; the overt forms differ on the surface and consequently must differ underlyingly. While clear to the analyst the learner needs an algorithm to determine this. Multi-recursive constraint demotion (MRCD) is such an algorithm.

Before describing MRCD some definitions are needed. A winner-loser pair (Prince & Tesar 1999) is a pair of overt forms and an input such that one of the overt forms, the "winner", is the optimal candidate that the grammar produces from the given input. A winner-loser pair can be identified with a comparative tableau (Prince 2000). A comparative tableau is a tableau that captures the preferences the constraints for the winning candidate, the losing candidate, or their indifference between the two candidates. It does not represent the number of constraint violations a candidate incurs. Consider the input of /ta.ka/ where both morphemes consist of short and unstressed segments and the two outputs of ta.**ká:** and **tá:**.ka where ta.**ká:** is the winning candidate. Then in 7 below is the comparative tableau created from this winner-loser pair. The markedness constraints WSP and *V: have no preference between the two output candidates; each does not violate WSP and each violates *V: once. Because they have equal violations for both markedness constraints (and not no violations) the tableau records the constraints' non-preference with an empty cell (in Prince 2000 this is represented with an 'e'). The winning candidate incurs one violation of ML and the losing candidate no violations of ML. This is represented with an 'L'. ML prefers the 'L'oser. The winning candidate violates MR not at all and the losing candidate violates MR once. This is represented with a 'W'. MR prefers the 'W'inner. The faithfulness constraints don't prefer either winner or loser since each violates both constraints equally.

7. Comparative tableau for the two overt forms ta.**ká:** and **tá:**.ka

		WSP	*V:	ML	MR	Id(s)	Id(l)
1	ta.**ká:** ~ **tá:**.ka			L	W		

Multi-recursive constraint demotion relies upon BCD to produce hierarchies. BCD itself relies upon winner-loser pairs to produce a stratified hierarchy. Given a set of winner-loser pairs BCD's first step is to inspect the constraints and determine if there are any that only either prefer the winner or have no preference in each of the given winner-loser pairs. Such constraints are now potentially rankable by BCD – that is, available for placement in the highest stratum. So in the example below with three winner-loser pairs and five constraints (four of which are markedness constraints, signified by the initial 'M', and one of which is a faithfulness constraint) BCD determines that both M1 and M2 are rankable.

8. Winner-loser pairs BCD will produce a hierarchy from

	M1	M2	M3	M4	F
1	W	e	e	L	e
2	e	e	W	L	W
3	e	W	L	e	L

These two constraints, M1 and M2, are then placed in the highest stratum together and the constraints M1 and M2 are removed from further consideration during the ranking procedure. Furthermore the winner-loser pairs that had a W in these now-ranked constraints are also removed from further consideration during the ranking procedure. The ranking of the constraints that have a W for these winner-loser pairs in effect "solves" these winner-loser pairs. The resultant hierarchy is now guaranteed to prefer the winner over the loser in these winner-loser pairs. After removing these constraints and winner-loser pairs BCD proceeds to apply the same steps to the remaining constraints and winner-loser pairs. In this case there remain three constraints and one winner-loser pair and BCD has constructed the incomplete stratified hierarchy {M1, M2}.

9. Winner-loser pairs and constraints remaining

	M3	M4	F
2	W	L	W

With the one remaining winner-loser pair BCD again inspects the constraints to determine if there are any that prefer either the winner or are indifferent and finds that in fact there are two, M3 and F. Under the constraint demotion algorithm both M3 and F would be placed in a stratum below the {M1, M2} stratum nearly completing the construction of the stratified hierarchy. But this is *biased* constraint demotion. The algorithm is biased against faithfulness constraints, so that when there is an option of ranking a set of constraints the algorithm will only choose markedness constraints. So here, the algorithm could rank M3 and F (or only F), but because it is biased against faithfulness constraints BCD selects only the markedness constraint M3 and places that in a stratum immediately below the previously constructed stratum {M1, M2}. This results in the partially constructed stratified hierarchy of {M1, M2} >> M3. BCD then proceeds merrily on its way by doing what it did before: remove the constraint M3 from consideration and that winner-loser pair which M3 preferred the winner over the loser in.

Remaining now are no winner-loser pairs and two constraints, M4 and F. BCD is stilled biased against faithfulness constraints so even though neither constraint disprefers the non-existent winner-loser pair BCD will rank M4 immediately below the previously constructed stratum and then will rank F at the bottom of the hierarchy producing the stratified hierarchy below.

10. Final hierarchy produced by BCD

    {M1, M2} >> M3 >> M4 >> F

Given a set of consistent winner-loser pairs BCD will always produce a hierarchy that will prefer all of the winners over all of the losers. BCD also has the property that given a set of *inconsistent* winner-loser pairs the algorithm will fail to produce a hierarchy. This, of course, is not a *failure* of BCD but a *benefit*. Given a large set of winner-loser pairs it can be exceedingly difficult for the analyst to determine if they are consistent or not – that is, to determine whether any hierarchy could produce the given set of input-output mappings captured in the winner-loser pairs. Furthermore, because BCD is an efficient algorithm for determining a hierarchy or for determining inconsistency (being O(n) where n is the number of constraints) it is a useful tool for the learner. Failing to produce a hierarchy is positive information; with a fixed set of constraints, a learner now knows that the current input-output mappings are wrong and must take corrective steps.

Now, multi-recursive constraint demotion with BCD first applies BCD to the empty list of winner-loser pairs producing a hierarchy with markedness constraints over faithfulness constraints. This hierarchy is then used to evaluate one of the overt forms, let us choose ta.**ká**. The initial hierarchy used to parse this form with the underlying specification of ‘ta’ being short and unstressed and ‘ka’ being the same produces the output **tá**.ka. This creates a winner-loser pair which is added to the (previously empty) list of winner-loser pairs upon which BCD is run. The winner-loser pair and resultant hierarchy are given below.

11. Winner-loser pair produced from MRCD

		WSP	*V:	ML	MR	Id(s)	Id(l)
1	ta.**ká** ~ **tá**.ka			L	W		

12. Resultant hierarchy from BCD

{WSP, *V:, MR} >> ML >> {Id(s), Id(l)}

This hierarchy then maps the input /ta.ka/ correctly to the output ta.**ká**. It is now used to parse the input /da.ka/. Recall that the three morphemes ‘ta’, ‘da’, and ‘ka’ all have the same underlying feature values: short and unstressed. MRCD is used recursively on the input /da.ka/ until a hierarchy that produces the correct output is achieved or until inconsistency is determined. The first parse of /da.ka/ unsurprisingly produces the same output as /ta.ka/ since they have the same underlying forms. This produces another winner-loser pair.

13. Winner-loser pair produced from MRCD

		WSP	*V:	ML	MR	Id(s)	Id(l)
1	ta.**ká** ~ **tá**.ka			L	W		
2	**dáa**.ka ~ da.**ká**		L	W	L		L

BCD is then applied to these two winner-loser pairs and reaches inconsistency. The first winner-loser pair requires that MR dominate ML while the second winner-loser pair requires that ML dominate MR (along with *V: and Id(l)). Clearly no ranking can satisfy these two conflicting requirements. MRCD has determined that this local lexicon is inconsistent. The learner will need to apply this approach to each of the local lexica for the contrast pair, determining in succession which of the local lexica are viable candidates for the underlying forms of the morphemes in the contrast pair.

Contrast pairs and their local lexica and means of extracting information from them are the main focus of this dissertation. In the next chapter a full learning algorithm is presented which demonstrates how the focus on contrast pairs and local lexica can be used to extract lexica information towards the goal of learning the languages of the given linguistic system.

Chapter 2. Using contrast pairs and local lexica

The question of how contrast and in particular how contrast pairs can yield information useful to the learner is best understood in the context of a learning algorithm. In this chapter an algorithm for learning that uses as its core information block the contrast pair is given. The learner is presented with the task of learning underlying forms for a given set of overt forms and learning the ranking that maps these underlying forms to the given overt forms. The two problems of determining the underlying forms and producing a constraint ranking are intertwined. What the underlying forms are restricts which rankings are possible and what the ranking is restricts what are possible underlying forms for a given set of overt forms.

As stated in the previous chapter, in this algorithm I assume that the learner has access to all the overt forms of the language, their morphological decomposition, and knowledge of all of the constraints of the grammar. The algorithm then must assign underlying forms to the given morphemes and produce a ranking that will map the underlying forms to the given overt forms.

Section 2.1. Overview of the algorithm

The algorithm I propose here, CPR: Contrast Pair and Ranking information,[3] consists of three parts, an initial lexical assignment, processing of contrast pairs, and final specification of underlying forms. The learner progresses through these stages sequentially and non-repetitively, the first stage of initial lexical assignment is completed before the learner progresses to the processing of contrast pairs, and similarly contrast pair processing is finished and not revisited once final specification of underlying forms is begun.

Section 2.1.1 Overview of Initial Lexical Assignment Stage

During the initial setting of featural values the algorithm attends to featural values that never alternate. Morphemes that have featural values that never alternate are given the underlying featural value that matches their surface realization (Tesar et al. 2003). Features that alternate remain unset at the end of the initial setting stage. The setting of non-alternating features is guaranteed to never set an incorrect featural value given the assumptions made here about the linguistic system (Tesar et al. 2003). Note that at the end of this stage it is not necessarily the case that any of the overt forms are fully specified. If at least one morpheme from each of the overt forms has an alternating feature then no overt forms will be fully specified. Also, the learner has no information

[3] The CPR algorithm presented here is a preliminary version. The full and final version of CPR is presented in Chapter 5 which consists of these three stages with a slight modification to the second stage along with phonotactic learning.

about the ranking. Since this algorithm is independent of any knowledge of phonotactics and no information about constraint rankings is produced all that is obtained from stage one is the setting of non-alternating features

Setting the feature values that do not alternate has the effect of reducing the lexical search space for the contrast pair processing stage. For each feature that is set for a morpheme there is a concomitant reduction of the number of local lexica by a factor of two for a contrast pair containing that morpheme. In this way the initial lexical construction stage reduces the search space for the contrast pair processing stage.

Section 2.1.2 Overview of Contrast Pair Processing Stage

The second stage of this learning algorithm consists of three parts, error-driven learning on the fully-specified overt forms, selection of the contrast pair with the least number of unset features, and setting of featural values for that contrast pair. These three steps are repeated until no further changes occur to the lexicon.

Step 1: Error-driven learning on fully-specified overt forms

The first part of this stage of the algorithm attends to those overt forms that are fully specified. At the completion of the first stage of the algorithm some of the overt forms may be fully specified. Information about the ranking of the target language can be gained from these forms. Error-driven learning using MRCD with Biased Constraint Demotion (Tesar 1997, Prince and Tesar 2004) is applied to these forms. This error-driven learning produces a set of winner-loser pairs that captures information about the target ranking. The Elementary Ranking Conditions (ERCs) (Prince 2002) represented by the winner-loser pairs generated from the error-driven learning restrict the possible rankings and for the learner reduce the search space by disallowing any rankings that do not conform to the ranking restrictions imposed by the ERCs. These winner-loser pairs are maintained throughout the remainder of the algorithm, both in the second stage of contrast pair learning and the final lexical assignment stage.

Step 2: Selection of contrast pair

In the second step of the contrast pair analysis stage the learner selects the contrast pair with the least number of unspecified features (that has at least one unspecified feature). In the case of a tie the learner randomly selects one of the minimally specified contrast pairs. Selecting the contrast pair with the least number of unset features has the potential to reduce the number of featural combinations considered in the third step of this stage. By ordering the contrast pairs by number of unset features the algorithm attempts to reduce the number of featural combinations considered. Furthermore, the setting of features for a given contrast pair, as may be accomplished in the third step, may reduce the number of unset features for later contrast pairs. In this way the total number of local lexica for a contrast pair considered may be reduced.

Step 3: Consistency check for local lexica and setting of underlying features

The third step of the contrast analysis stage is to determine which combinations of underlying features are consistent for the selected contrast pairs and to determine if any features may be set. First, for each combination of possible underlying features, that is, for each local lexicon, the learner uses error-driven learning with BCD to determine if that local lexicon is consistent. So, for the given local lexicon, the learner determines whether there is a ranking that produces these overt forms from the given underlying forms with the featural values given by the specified local lexicon. Error-driven learning on the fully-specified overt forms produces a set of winner-loser pairs that represent ranking restrictions on the target language. These ERCs are used during the application of error-driven learning on the local lexica. For some contrast pairs these ranking restrictions will have the effect of reducing the number of local lexica that are consistent. Only those rankings that adhere to the restrictions imposed by the ERCs generated in the first step of this stage are possible for a given local lexicon. In this way knowledge gained about the ranking from forms that are fully specified is maintained throughout the algorithm; the learner attempts to use previously gained knowledge about the ranking while determining information about the underlying forms.

After determining which local lexica are consistent, for each morpheme and each unset feature in the contrast pair, the learner inspects the values of that feature in the consistent local lexica. If, for a given morpheme, the feature for that morpheme has the same value in all consistent local lexica the learner sets that value for that feature in the lexicon. That featural value is no longer unset henceforth. In this manner does the learner set underlying forms using contrast pairs.

A feature that remains unset after processing of a contrast pair may not be redundant. If a feature is unset after processing of a contrast pair the only thing that can be concluded is that there was insufficient information gained from this contrast pair at this time to set that feature. The feature may need to be set to a particular value for the learner to successfully learn the language. All that can be concluded is that current knowledge about the ranking and lexicon is insufficient to determine that. In fact, a feature that remains unset after processing of a contrast pair may be set while processing the same contrast pair later in the algorithm. This will occur when further information about the lexicon and ranking cause some previously consistent local lexica to be inconsistent allowing the feature in question to be set.

This procedure is guaranteed to only set correct featural values. Indeed, for a given contrast pair all possible featural combinations are considered by the learner, one of which is the correct featural setting for the target language. The local lexicon that has the same featural settings as the target language is guaranteed to be consistent since the target ranking will always produce the correct overt forms and since the ranking restrictions generated from the fully-specified overt forms cannot be inconsistent with the target ranking. Now, since a local lexicon with featural settings matching the target lexicon is consistent, if the learner sets a featural value it must match the target lexicon. This is because the learner only sets a featural value if it has the same value across *consistent* local lexica and because the target lexicon is always guaranteed to be consistent. Hence no incorrect featural values will be set using this method of setting uniform featural values across consistent local lexica.

It should be noted that it is possible that no featural values for a given contrast pair will be set when considering local lexica. This in fact happens in the example language presented below.

After completing this third step the learner repeats the three steps of the contrast pair processing stage, applying error-driven learning on the fully specified forms, selecting the contrast pair with the least number of unset features, and the setting of consistently valued features among the consistent local lexica. The learner repeats these three steps until no further feature values are set in the lexicon, that is, until no changes have been made to the lexicon and all contrast pairs have been considered since the last lexical modification.

Some performance improvements could be introduced to algorithm presented here. Contrast pairs that have been processed in the contrast pair processing stage need not be re-processed unless one of two things have happened, either further ranking information has been gained from the error-driven learning on the fully-specified overt forms, or further lexical information has been gained about morphemes contained in the given contrast pair. If no features have been set for any of the morphemes in a contrast pair and no further ranking information is known, the processing of the contrast pair will proceed as it did previously producing no new information. By not considered these contrast pairs some efficiency gains may occur.

Section 2.1.3 Overview of Final Lexical Assignment and Ranking Determination

After the completion of the contrast pair processing stage the learner proceeds to the final stage, the setting of remaining feature values and the determination of a ranking for the language. At the end of the second stage the learner may not have set all of the feature values for all of the morphemes. The final stage assigns a value for all remaining unset features. A default value is assumed to exist for all features. The algorithm assigns this default value for all unset features. At this point all morphemes are fully specified. Error-driven learning using BCD is used on all of the overt forms (which are now all fully specified). This produces a final ranking assuming that the default value assignment did not assign any incorrect featural values. The learner is now done, having produced a lexicon and a ranking that produces the given overt forms.

In an online version of learning, one in which forms are presented not en masse at the beginning of the algorithm but appear throughout the learning process and all forms need be fully specified at all times, an alternative approach to a feature being unset is compatible with this algorithm. Instead of having unset features all features would receive a default value initially and be marked as having a default value. Those features that were not set (or changed as it would be in an online version) during the contrast pair processing stage would receive their default values. The conclusions about the efficacy of this approach would still hold in this type of learning environment.

Section 2.2 Description of entire algorithm in pseudocode

Below is the algorithm in its entirety given in pseudocode.

1. Pseudo-code of CPR.

```
Stage 1.  Initial lexical assignment
        Non-alternating features are set to their surface realization
Stage 2.  Contrast pair processing
        Step 1.  Processing of fully specified overt forms
                Error-driven learning using BCD is applied to all overt forms that
                contain only morphemes that are underlyingly fully specified.  The
                winner-loser pairs generated from this are used throughout the
                remainder of learning.
        Step 2.  Selection of contrast pair
                The contrast pair that has the least (though non-zero) number of
                unset features and that has not been chosen since the last lexical
                modification is chosen.
        Step 3.  Setting of featural values of uniformly valued features of
                consistent local lexica
                For each local lexicon of the contrast pair error-driven learning
                using BCD is used to determine whether the local lexicon is
                consistent.  Feature values that have the same value across all
                consistent local lexica are set in the learner's lexicon.
        Repeat steps 1 – 3 until all contrast pairs have been processed since the
        last lexical change.
Stage 3.  Final lexical assignment and ranking determination
        A default value is assigned to all unset featural values.  Error-driven
        learning using BCD is applied to the fully specified overt forms to produce
        a final ranking.
```

Section 2.3 Application of algorithm

In this section I apply the above algorithm to one of the languages from the linguistic system presented in Chapter 1. The linguistic system is given again below.

Recall that the system contains two features, stress and length, each of which has a binary value. Overt forms are bi-morphemic consisting of a root and a suffix. There are four markedness constraints and two faithfulness constraints. The markedness constraints are MainLeft, MainRight, WSP, and *V: and the faithfulness constraints are Ident(stress) and Ident(length). These constraints are given below.

2.	Ident(s)	The stress value must be identical to its input correspondent
3.	Ident(l)	The length value must be identical to its input correspondent
4.	MainLeft	Stress must fall on the leftmost syllable
5.	MainRight	Stress must fall on the rightmost syllable
6.	*V:	Do not have a long vowel
7.	WSP	If a vowel is long it must be stressed

Section 2.3.1 The language being learned

As discussed before, the learner is not attempting to learn this linguistic system. The linguistic system is known to the learner. The learner is attempting to learn a language of the linguistic system. The language the learner is attempting to learn in this example using the above algorithm is given by the ranking below.

8. WSP >> Id(s) >> ML >> MR >> Id(l) >> *V:

In this language WSP is undominated; no long unstressed vowel surfaces. Stress placement is determined by the underlying stress feature. In the case where both root and suffix are identically specified for stress, stress is placed on the root since ML >> MR. Alignment and underlying stress completely determine stress placement, underlying length never does. But Id(l) outranks *V: so when an underlyingly long syllable is stressed either because of underlying stress or because it is the root (and the suffix is unstressed) that syllable may surface long. When a syllable is underlyingly long but not stressed it will never surface as long even though Id(l) outranks *V:. This is because WSP is undominated; an unstressed vowel can never be long.

This language has four unique roots and three unique suffixes that combine to give twelve distinguishable overt forms. All the root and suffix combinations and their outputs under this ranking are given in the table below. Typographically in the table below, the initial consonant of a syllable represents morphemic identity. Roots are introduced by p, b, t, d and suffixes are introduced by k, g, s, z.[4] Long vowels on the surface are represented by 'aa' and short surface vowels by 'a'. Surface accent is represented by bolding of the syllable and by the acute accent diacritic.

9. Mapping of morphemes

/stress, length/	ka /–X/	sá /+–/	záa /++/
pa /––/	**pá**ka	pa**sá**	pa**záa**
baa /–+/	**báa**ka	ba**sá**	ba**záa**
tá /+–/	**tá**ka	**tá**sa	**tá**za
dáa /++/	**dáa**ka	**dáa**sa	**dáa**za

The necessary underlying specifications for each of the morphemes are also given in this table. So the root pa must be underlyingly unstressed and short. The first suffix, ka, has the property that its underlying value for length does not matter. This is represented by an 'X' in the first row. What this means is that a suffix that is underlyingly unstressed and long will have the same surface realizations in all environments as a morpheme that

[4] This orthographic representation was suggested by Alan Prince.

is underlyingly unstressed and short. In this language a suffix (but not a root) that is underlyingly unstressed will always surface as unstressed and short, hence there is no way to distinguish underlying length on suffixes that are underlyingly unstressed. While no features are globally neutralized in this language, length is neutralized on suffixes that are unstressed.

The underlying values given above are necessary specifications for each of the morphemes in this language; to correctly map these underlying forms to their respective surface realizations each of the morphemes must have precisely these underlying values with the exception of ka which must be either unstressed and short or unstressed and long. The learner is presented with these twelve overt forms and the morphological decomposition of these forms. The learner is given the knowledge that there are four roots and three suffixes and that each combination of root and suffix maps to the given respective overt form. The learner does not have any prior knowledge of the underlying values nor of the ranking that will produce these mappings. The learner knows precisely what is given in the table above except for the underlying values of the morphemes. It is the task presented to the learner to determine a lexicon and a ranking that will reproduce the paradigm as restrictively as possible.

Section 2.3.2 Stage one: initial lexical specification

The first stage of the algorithm assigns underlying values to those featural values that do not alternate. Here note that every morpheme except the root baa and the suffix záa have at least one feature that does not alternate. Consider the morpheme pa. In all environments, ka, sá, and záa, the morpheme pa surfaces as short. Because it always surfaces as short the learner assigns for the length feature for pa the underlying value of short. The root pa does alternate with respect to stress and so the learner leaves the underlying value for stress for pa as currently unset. The learner then repeats this inspection for each of the other featural values for each of the other morphemes. The results of the initial lexical assignment are given below.

10. Lexicon after initial lexical assignment, '?' means value is currently unset

	/stress len/
pa	/?–/
baa	/??/
tá	/+–/
dáa	/++/

	/stress len/
ka	/––/
sá	/?–/
záa	/??/

After the initial lexical assignment of non-alternating features three of the morphemes are fully specified, tá, dáa and ka; their values are completely known to the learner. Both features, stress and length, for morphemes baa and záa alternate and hence the learner has set no underlying values for baa and záa; their values are completely unknown to the learner at this time. Since there are two roots and one suffix that are completely specified the learner knows the underlying specification for two overt forms, **tá**ka and **dáa**ka. The learner will be able to use this information in the first step of contrast pair analysis.

Section 2.3.3 Stage two: contrast pair processing stage

Step 1: processing of fully-specified forms

Having specified the initial lexicon in the first stage of CPR the learner moves on to the second stage, contrast pair processing. This consists of three steps that are repeated until no further changes occur in the lexicon. The first step is applying error-driven learning to those forms that are fully specified. After the initial lexical assignment only two overt forms are fully specified, **tá**ka and **dáa**ka. Error-driven learning using BCD is applied to both of these forms producing ranking information that the learner maintains throughout the algorithm. The results of this error-driven learning are the two winner-loser pairs below.

11. Results of error-driven learning on **tá**ka and **dáa**ka

			WSP	*V:	ML	MR	Id(s)	Id(l)
1	/dáaka/	**dáa**ka ~ **dá**ka		L				W
2	/táka/	**tá**ka ~ ta**ká**			W	L	W	

These winner-loser pairs impose ranking restrictions on the learner. The first row, resulting from the application of error-driven learning to /dáaka/ yields the information that Id(l) must dominate *V: in any ranking considered by the learner. The second row imposes the restriction on the final language that either ML or Id(s) must dominate MR. These restrictions will reduce the number of consistent local lexica for certain contrast pairs in the following stage allowing the learner to set more features by ruling out inconsistent local lexica. As more overt forms become fully specified the ranking restrictions imposed by error-driven learning by these overt forms have the ability to further constrain which local lexica are consistent, allowing the learner to set more features.

Step 2: selection of contrast pair

The learner, after applying error-driven learning to the fully specified overt forms, selects the contrast pair with the least number of unset features with the requirement that it have at least one unset feature. Selecting a contrast pair with no unset features would not yield any new information because the overt forms of such a contrast pair are fully specified and hence no features need be set for the morphemes of the contrast pair. For the first pass over these contrast pairs there are precisely two contrast pairs with exactly one unset feature, the contrast pair **pá**ka ~ **dáaka** and the pair **tá**sa ~ **dáa**sa. The first pair has the feature stress in pa unset; all other features in the morphemes pa, ka, and dáa are already set in the lexicon. The second pair only has the stress feature of sá unset; tá and dáa are entirely set and sá's length is set to short. All other contrast pairs that do not have all of their morphemes fully set have more than one unset feature. The algorithm then randomly selects one of these two minimally unset contrast pairs, say **pá**ka ~ **dáaka**.

Step 3: processing of contrast pair

The three morphemes in this pair, pa, ka, and dáa, have between them only one unset feature, the stress of pa, the other features being set during the initial lexical assignment stage. Because there is only one unset feature there are two local lexica (one for each value of the unset feature) for the learner to test consistency of. The two local lexica are given below.

12. Two local lexica for contrast pair **pá**ka ~ **dáaka**

a. Local lexicon 1

	/stress len/
pá	/+–/
dáa	/++/

	/stress len/
ka	/––/

b. Local lexicon 2

	/stress len/
pa	/––/
dáa	/++/

	/stress len/
ka	/––/

The learner now applies error-driven learning on this contrast pair for each of these local lexica. For the first local lexicon with pa having the underlying specification of plus stress and minus length the learner determines that it is consistent. That is, the learner can produce a hierarchy that produces these two overt forms from local lexicon 1. The learner then proceeds to check the consistency of local lexicon 2 determining that that lexicon is also consistent. The BCD rankings that confirm consistency are given below.

13. Local lexicon 1 ranking

WSP >> F(s) >> ML >> MR >> F(l) >> *V:

14. Local lexicon 2 ranking

WSP >> F(s) >> ML >> MR >> F(l) >> *V:

15. Consistency check for two local lexica of pair **pá**ka ~ **dáaka**

	pa stress	Consistent
LL1	−	Yes
LL2	+	Yes

So the learner determines that both local lexica for this contrast pair are consistent (in this case with the same ranking, though this may not always be the case), and since both are consistent no setting of features occurs because the unset feature in this contrast pair, the stress of pa, does not have the same value across consistent local lexica. After determining that no setting of features occurs, the learner proceeds through the steps of the contrast pair processing stage again.

Repetition of steps 1 – 3

There are no new fully specified overt forms and so no new winner-loser pairs are generated in the first step. The learner selects the next contrast pair that has the least number of unset features, in this case it is **tá**sa ~ **dáa**sa. A consistency check again determines that both local lexica for this contrast pair are consistent. No underlying forms are set.

The learner then turns to the contrast pair pa**sá** ~ **tá**sa which has two unset features, pa stress and sa stress. These two unset features yield four local lexica for the learner to test for inconsistency.

First consider the local lexicon that has pa underlyingly stressed and sá underlyingly unstressed. Error-driving learning is applied to the two overt forms of the contrast pair. BCD first produces a hierarchy from the ERCs from the first step of contrast pair analysis. These were the ERCs derived from error-driven learning on the fully-specified overt forms. The ERCs and resultant hierarchy are given below.

16. ERCs from step 1

	WSP	*V:	ML	MR	Id(s)	Id(l)
1		L				W
2			W	L	W	

17. Hierarchy produced from these ERCs using BCD
{WSP, ML} >> MR >> Id(l) >> *V:

Error-driven learning starting with this hierarchy determines that this local lexicon is inconsistent. The first winner-loser pair results from parsing /pasá/ with the hierarchy above. This hierarchy produces the incorrect output **pá**sa yielding the winner-loser pair below.

18. Winner-loser pairs from error-driven learning on the contrast pair pa**sá** ~ **tá**sa

			WSP	*V:	ML	MR	Id(s)	Id(l)
1	/pa**sá**/	pa**sá** ~ **pá**sa			L	W	L	

At this point, error-driven learning applies BCD to all three ERCs (two from the first step and the newly generated ERC). BCD tells the learner that these three ERCs are inconsistent and consequently this local lexicon is not a possible local lexicon for the contrast pair. The inconsistency stems directly from the new information gathered from the newly created ERC. This ERC is inconsistent with the first ERC generated from the fully-specified overt forms. Both ERCs are given below shorn of their input and output information.

19. Inconsistent ERCs from error-driven learning

	WSP	*V:	ML	MR	Id(s)	Id(l)
16.1			W	L	W	
18.1			L	W	L	

ERC 16.1 was generated from the fully-specified overt forms and requires that either ML or Id(s) dominate MR. The ERC just created while processing this local lexicon require that MR dominate both ML and Id(s). Clearly no ranking can satisfy both of these requirements and consequently the local lexicon is inconsistent.

This procedure is then repeated for the remaining three local lexica. The results of applying error-driven learning to the four different local lexica are given below.

20. Results of error-driven learning on all four local lexica

	pa stress	**sá stress**	**Consistent**
Local lexicon 1	−	−	yes
Local lexicon 2	−	+	yes
Local lexicon 3	+	−	no
Local lexicon 4	+	+	no

For this contrast pair only two of the four local lexica yield consistent rankings, the local lexicon with pa unstressed and sá unstressed and the local lexicon with pa unstressed and sá stressed. Across these two local lexica the value of pa is the same, unstressed, and the learner sets the value of pa in the lexicon to unstressed. The value of sá is not the same across consistent local lexica and hence no setting of its featural value occurs. In this case the learner has determined from this contrast pair that pa stress necessarily must be set to unaccented for there to exist a ranking that can produce the two given overt forms. From the investigation of consistent local lexica the learner has gained information about the lexicon, though the learner does not at this stage gain any information about the ranking that produces these overt forms, only that pa must be unstressed to produce these two forms.

After processing of this contrast pair the learner proceeds to process all remaining contrast pairs until no further featural values can be set. For this language upon completion of the contrast pair processing stage the learner will have set all featural values for all of the morphemes in the language. The final stage of the learning algorithm then needs not set any default featural values (this will not always be the case as an example in the next chapter will show) and only needs to produce a final ranking from the given now fully specified overt forms. The final stage uses BCD to produce a final ranking. The learner applies MRCD using BCD all of the overt forms (that are now all fully-specified) to produce a final ranking. The ranking produced is below.

21. Final ranking produced by error-driven learning on all forms
WSP >> Id(s) >> ML >> MR >> Id(l) >> *V:

This is precisely the ranking the learner is attempting to learn. The learner has now successfully learned this language having produced a set of underlying forms for the given morphemes and a ranking that produces the given overt forms from the underlying forms the learner has specified.

Section 2.4 Algorithm performance

CPR succeeds in learning this language and does so by considering only a tiny subset of the possible lexica for the given morphemes of this language. Initially the learner is presented with seven morphemes each of which could have four possible underlying forms. So the learner initially has 4^7 = 16,384 possible lexica to choose from. In the execution of this algorithm the learner only considers a total of 24 lexica throughout the contrast pair processing stage. This is slightly less than two tenths of one percent of the total possible lexica the learner needs to discriminate amongst. Granting that the learner correctly assigns featural values to non-alternating features (as the learner does in this algorithm in the initial lexicon construction stage) there still remain six unset features for a total of 2^6 = 64 possible lexica for the learner to consider. This algorithm only considers slightly more than a third of those. In fact, for all of the possible languages in this linguistic system the learner only considers a very small subset of possible lexica for any given language. The exhaustive searching of local lexica at the contrast pair level has the potential to greatly reduce the overall search of lexica at the language level.

The linguistic system given above has precisely 24 unique languages (Brasoveanu 2004). For any one of the 24 languages the algorithm learns the language; that is, given the overt forms of one of the 24 languages the algorithm will produce a correct lexicon and a ranking that will produce the given overt forms from the lexicon the algorithm constructs.

The first two stages of this algorithm will never set an incorrect lexical value. The first stage of initial lexical assignment does not set incorrect values. This follows from the assumptions of the linguistic system and is shown in Tesar et al. 2003. The second stage of this algorithm also does not set incorrect lexical values. For a given contrast pair the learner considers all of the possible lexica for that contrast pair one of which is the correct lexicon. Consistency is determined for each of the local lexica. A feature is set in the lexicon during this stage if and only if it has the same value in the consistent local lexica. Now one of the local lexica is the target lexicon and this lexicon is guaranteed to be consistent. So if a feature is set it must agree with the target lexicon. Hence no features are incorrectly set during the contrast pair processing stage.

In the linguistic system detailed above the third stage of the algorithm, final lexical assignment and ranking determination, does not set any features incorrectly. For any given language in this system that the algorithm does not set all of the features for during the first two stages any remaining unset features are set correctly via default assignment. Furthermore, in every case these features that are not set in the first two stages are not contrastive. So *any* setting of these features in the third stage of the algorithm would result in a correct lexicon. The learner did not just get lucky by having the default values be the necessary values for these features. All features that had necessary values were set in the first two stages of the algorithm.

The first two stages of this algorithm are guaranteed to be correct and in this linguistic system the third stage unavoidably assigns correct underlying forms. The second stage of the algorithm need not set all contrastive features though. In this linguistic system it is the case that all contrastive features are set in the second stage but as shown in the next chapter it is possible for a contrastive feature to not be set in the second stage. Attempting to determine a ranking in the third stage will lead to inconsistency highlighting the fact that the contrast pair processing stage did not set a contrastive feature.

The next chapter demonstrates precisely such a linguistic system. The algorithm presented here only extracts lexical information from the contrast pairs and not ranking information. As will be shown it is the lack of extraction of ranking information that leads the current version of the contrast pair processing stage to not set a contrastive feature and the use of that extracted ranking information that will allow the learner to determine more contrastive features of the language.

Chapter 3. Limits of local lexica

The process of considering all local lexica for a given contrast pair yielded complete and correct results for the linguistic system presented in the previous chapter. But what are the limits of this algorithm? In a more feature-rich environment with constraints that impose a more complex interaction amongst those features the previous algorithm can fail to yield a complete lexicon and concomitant ranking.

In the previous system when there was ambiguity over which feature was driving the surface contrast the algorithm was able to resolve the ambiguity by considering only lexical information conveyed by the contrast pair at hand and the ranking information from the fully-specified overt forms. In the linguistic system presented below this information is not sufficient to resolve this type of ambiguity. By introducing a third feature and introducing a markedness constraint that relates all three features (amongst other constraints) there will be contrast pairs whose surface contrast on two of the features can be explained by either of two features in isolation but not together. That is, for any given contrast pair either of two features could explain the surface contrast but the contrast pairs all together require precisely one of the features to be set a certain way. This cross contrast pair implication is lost by only considered lexical information from a contrast pair. This chapter will demonstrate how this information can be lost by applying the algorithm from the last chapter to a system that has this property and will show that the information that is needed comes from extracting ranking information from contrast pairs that are not fully specified. The following chapter will present an algorithm for extracting this shared ranking information.

Section 3.1 A linguistic system with three features

This linguistic system builds on the system presented in the previous chapter. It contains three features, two contained in the previous system, stress and length, and a third binary feature, aspiration. Underlying morphemes (which are restricted to single segments) are then specified for stress, length, and aspiration.[5] The constraints from the previous system are present here along with three new constraints, one of which is a faithfulness constraint, the other two markedness constraints. The full set of constraints for this system is given below.

The faithfulness constraints are listed in 1 – 3.

1.	Id(s)	The stress value must be identical to its input correspondent
2.	Id(l)	The length value must be identical to its input correspondent
3.	Id(a)	The aspiration value must be identical to its input correspondent

The markedness constraints are listed in 4 – 9.

4.	ML	Stress must fall on the leftmost syllable
5.	MR	Stress must fall on the rightmost syllable
6.	WSP	If a vowel is long it must be stressed
7.	*V:	Do not have a long vowel
8.	NoAsp	Do not have aspiration (Hammond 2005)
9.	*[+s, –l, –a]	Do not have a segment that is stressed, short, and not aspirated

[5] I assume that it is the mono-syllabic morpheme that is specified for these features and it is not either an underlying vowel or consonant. These features are an idealized representation of the linguistic features of stress, length, and aspiration.

The first three constraints above are faithfulness constraints; the last six are markedness constraints. The first two faithfulness constraints are the Ident constraints for the features stress and length. The new faithfulness constraint is the faithfulness constraint for the new feature aspiration; it says simply be identical to the underlying aspiration specification for a given segment.

Of the six markedness constraints the first four are identical to the markedness constraints from the previous linguistic system. The fifth markedness constraint, NoAsp, says that +aspiration is the marked value of aspiration and that, all other things being equal, the surface value of aspiration for a given segment should be –aspiration. This is directly analogous to *V:, which is a constraint that could be reformulated precisely as *+l, do not be long. The final markedness constraint is *[+s, –l, –a] which states that a segment should not have the feature combination of +s, –l, and –a, that is do not be stressed, short, and unaspirated. This somewhat analogous to the constraint WSP which can be restated as *[–s, +l], do not be both stressed and long. If one views both length and aspiration as prominence enhancers the constraint *[+s, –l, –a] can be viewed as saying that a stressed segment should be have at least one of the prominence markers of length or aspiration if it is stressed. So, if a segment is stressed it should also be long or be aspirated to enhance its prominence (or it could be both long and aspirated). A segment only violates this constraint when it is stressed and lacking in one of the other two prominence features in the language.[6]

In defining this linguistic system the same assumptions about word form are made as in the system from the previous chapter. Every overt form consists of exactly one root and one suffix, and every root and every suffix contain exactly one segment. So, since there are only three features in the language and each root consists of exactly one segment there can be at most $2^3 = 8$ unique roots, one for each combination of specifications of the three binary features. Similarly, since suffixes also consist of exactly one segment, there can be at most 8 unique suffixes for any given language in this system. These eight roots and suffixes then can combine to produce at most 64 unique forms for those most faithful languages though the ones investigated in depth here will have significantly fewer forms. Although this system is limited in its lexicographic bounty it is sufficiently rich in its complexities to shed light on certain limits of the processing of contrast pairs.

Section 3.2 A description of a particular language in this system

In this section the algorithm is applied to a particular language of this linguistic system. The algorithm will succeed in setting correctly a large majority of the features of the given overt forms but will leave unset at the end of the contrast pair processing stage two crucial features. The final stage of the algorithm will assign a value to these features incorrectly and will produce an incorrect ranking for the language.

[6] English uses aspiration as a prominence enhancer on stressed syllables. Its relation to the proposed constraints here will be discussed further in the final chapter.

One of the unset features remains unset because there is not enough information in all of the overt forms to determine what language the learner is learning. The language presented below is in a subset relationship with another language of this linguistic system. Because of this, the overt forms themselves do not contain enough information to distinguish these two languages. The other crucially unset feature could be set if enough information were gathered from the overt forms. Attending only to the lexical information given by the contrast pairs does not suffice to set this feature though. The ranking for this language is given below.

10. WSP, *[+s, –l, –a] >> Id(s) >> ML >> MR >> Id(l) >> NoAsp, *V: >> Id(a)

Describing this language phonotactically, an unstressed segment is always short and unaspirated. Stressed segments are always long and unaspirated or short and aspirated. Stress itself is free to appear on either roots or suffixes. Putting these facts together, there are precisely four phonotactically distinct overt forms in this language, **r^{h}á**ra, **ráa**ra, ra**r^{h}á**, ra**ráa**.

In this language WSP and *[+s, –l, –a] are both undominated (they do not interact since WSP is a condition on unstressed syllables and *[+s, –l, –a] is one on stressed syllables and neither plays a determining role in stress placement in this language). Stress placement is determined by underlying stress; if both root and suffix are stressed (or unstressed) the root is stressed since ML >> MR and ML is immediately dominated by Id(s) as shown below (omitted constraints in the following tableaux do not prefer the winner or the loser). In the next few tableaux the underlying representation of a morpheme is depicted by a series of three pluses or minuses representing, in sequence, the underlying value of stress, length, and aspiration. So in row 1 of the tableau below the input consists of two identical morphemes both of which are stressed and are underlyingly short and unaspirated represented by the /+--/ /+--/ in the input column. The winning output for this input is [+-+][---], an initial stressed, short, and aspirated root (represented by [+-+]) and an unstressed, short, and unaspirated suffix.

11. ML dominates MR and Id(s) dominates ML

	Input	Winner~Loser	ML	MR	Id(s)
11.1	/+--/ /+--/	[+-+][---] ~ [---][+-+]	W	L	
11.2	/---/ /+--/	[---][+-+] ~ [+-+][---]	L		W

Morphemes that are unstressed always surface as short and unaspirated. This comes about because of two facts. First, NoAsp >> Id(a) as established in row 12.1 in the tableau below. This ensures that in non-stressed syllables +aspiration will not surface. Now, non-stressed syllables always surface short even though Id(l) >> *V: (tableau 13 establishes this fact). This pattern obtains because WSP is undominated and states that unstressed syllables can never be long.

The behavior of length and aspiration is more complicated under stress. The constraint *[+s, –l, –a] is undominated in this language and so a syllable cannot be stressed and both –l and –a. Since stress must be placed on some syllable and *[+s, –l, –a] is undominated a stressed syllable must be either long or +a to satisfy *[+s, –l, –a]. And since Id(l) >> NoAsp, *V: and Id(a), underlying length will determine the repair mechanism that is called upon to satisfy *[+s, –l, –a]. If a segment is stressed and underlyingly long then the syllable will surface as [+s, +l, –a]. The underlying specification of aspiration in this case does not matter since NoAsp >> Id(a) and *[+s, –l, –a] is satisfied since the segment is surfacing long. Similarly, if the segment that is stressed is underlyingly short the segment will surface [+s, –l, +a]. In this way it satisfies Id(l) and satisfies *[+s, –l, –a] by being +a. Again, the underlying specification of aspiration does not matter since length is determining the means of satisfaction of *[+s, –l, –a]. So even though NoAsp dominates Id(a) and this configuration prohibits the surfacing of +a on segments that are not stressed, the constraint *[+s, –l, –a] and faithfulness to length can cause a segment to surface as +a when the stressed segment is underlyingly short. In this particular language the value of aspiration is completely predictable from the values of stress and length; it is not contrastive in any environment.

12. Non-stressed syllables do not surface as aspirated or long

	Input		WSP	*V:	NoAsp	Id(l)	Id(a)
12.1	/+-+/ /--+/	[+-+][---] ~ [+-+][--+]			W		L
12.2	/+-+/ /-+-/	[+-+][---] ~ [+-+][-+-]	W	W		L	

13. Id(l) >> NoAsp, *V:, Id(a)

	Input		*V:	NoAsp	Id(l)	Id(a)
13.1	/-++/ /---/	[++-][---] ~ [+-+][---]	L	W	W	L
13.2	/---/ /---/	[+-+][---] ~ [++-][---]	W	L	W	L

14. *[+s, -l, -a] >> NoAsp, Id(a), and Id(l) is dominated by Id(s)

	Input		*[+s,-l, -a]	*V:	No Asp	ML	MR	Id(s)	Id(l)	Id(a)
14.1	/+--//---/	[+-+][---] ~[+--][---]	W		L					L
14.2	/-+-//+--/	[---][+-+] ~[++-][---]		W	L	L	W	W	L	L

The tableaux 11, 12, 13, and 14 establish the necessary rankings of the language given above. Specifically required and demonstrated are that WSP >> Id(l) >> NoAsp >> Id(a), that Id(l) >> *V:, that *[+s, -l, -a] >> NoAsp, that Id(s) >> ML >> MR, and that Id(s) >> Id(l). Applying BCD to the above winner-loser pairs produces the hierarchy given for this language.

Of the eight possible roots and eight possible suffixes in this linguistic system this language only exhibits four distinct root behaviors and three distinct suffix behaviors. The roots distinguish all possible combinations of underlying values of stress and length and do not distinguish underlying aspiration. This yields precisely four roots, one for each value of stress and length. The suffixes only distinguish length when the suffix is underlyingly stressed. In an unstressed suffix the segment will never attract stress and hence will always surface as short and unaspirated. The full behavior of the four roots and three suffixes are given in the table below. Typographically, the same notation for stress and length is used as in the last chapter. Stressed segments are marked with the acute diacritic and long segments are signified by ‘aa’. An aspirated segment is marked by the superscript ^{h} on the onset. Here again the onset consonant signifies only morphemic identity, the four roots are denoted with {p, b, t, d} and the three suffixes with {k, s, z}. The underlying specifications for each of the morphemes are given in the order stress, length, and aspiration. An ‘X’ means that there is no necessary underlying specification for that feature for that morpheme. The ‘X’ notation is used for these forms because richness of the base precludes the analyst from positing a unique underlying representation for these morphemes given the ranking the learner is attempting to learn. Now, for example, tá has an underlying specification of /+–X/. This means that underlying it is stressed, short, and the value of aspiration can be either plus or minus. Either value produces the morphemic behavior given in the table below.

15. Mappings of Language L1

/stress, length, asp/	ka /–XX/	sá /+–X/	záa /++X/
pa /––X/	**p**^{h}**á**ka	pa**s**^{h}**á**	pa**záa**
baa /–+X/	**báa**ka	ba**s**^{h}**á**	ba**záa**
tá /+–X/	**t**^{h}**á**ka	**t**^{h}**á**sa	**t**^{h}**á**za
dáa /++X/	**dáa**ka	**dáa**sa	**dáa**za

The underlying values for this language given above are the necessary underlying specifications that the analyst has determined, not the learner. The learner has the separate task of determining underlying forms for this language without the analyst’s knowledge.

Section 3.3.1 The application of the algorithm to this language

The learner is given the twelve overt forms of this language and the morphological decomposition. The first stage of the learning algorithm assigns underlying values to features that do not alternate. The result of this stage is given below.

16. Initial lexicon after the initial lexical assignment stage

	/s l a/
pa	/?–?/
baa	/??–/
t^há	/+–+/
dáa	/++–/

	/s l a/
ka	/–––/
sá	/?–?/
záa	/??–/

Three of the seven morphemes are now completely specified. The roots t^há and dáa do not alternate in this language and the learner completely specifies them correctly in the initial lexical assignment stage. Similarly the suffix ka does not alternate and the learner correctly specifies its underlying form in this initial stage.

At this point there are now two overt forms that are completely specified, **t^há**ka and **dáa**ka. Moving on to the contrast pair processing stage, the learner applies error-driven learning on these two forms producing a set of winner-loser pairs. Contrast pair processing proceeds and all contrast pairs are processed resulting in the setting of four of the eight features that were not set during the initial lexicon setting stage. The resulting lexicon after the contrast pair processing stage is given below.

17. Lexicon after contrast pair processing stage

	/s l a/
pa	/−−?/
baa	/−?−/
tʰá	/+−+/
dáa	/++−/

	/s l a/
ka	/−−−/
sá	/+−?/
záa	/?+−/

Four features have been set beyond what was set in the initial lexical assignment. The morpheme pa is set to unstressed, baa is set to unstressed, sá is set to stressed, and záa is set to long. The contrast pairs in which these features figure are given below.

18. Features set during contrast pair processing stage and contrast pairs that yielded the featural settings

pa /**−**,−, ?/	Stress is set to **−** in contrast pair **<**pa**záa, tʰá**za**>**
baa /**−**, ?, −/	Stress is set to **−** in <ba**záa, tʰá**za**>**
sá /**+**, −, ?/	Stress is set to + in **<báa**ka**,** ba**sʰá>**
záa /?, **+**, −/	Length is set to + in **<** **pʰá**ka**,** pa**záa>**

During the contrast pair processing stage no further morphemes are fully set and consequently no further ranking information is gained from error-driven learning on the fully specified overt forms. Of the four features that remain unset, the aspiration values of both pa and sá need not be set to any particular value to successfully learn this language. Aspiration is fully predictable from the specifications of stress and length and the underlying value of aspiration is immaterial to the surface realization of this feature for every overt form. The aspiration feature in the target lexicon can be any value for any morpheme, hence if a default value is assigned to aspiration in the final stage no impediment to acquiring the target language will occur. A pernicious problem does arise in the lack of specification of the other two features.

The length of baa must be set to long for the learner to acquire the correct grammar. Under the target ranking, repeated below, if baa's length were set to short, the underlying form **bá**ka would map incorrectly to **b**$^{\mathbf{h}}$**á**ka, when it should surface as **báa**ka. Similarly, záa must be underlyingly stressed. The suffix záa in conjunction with root pa will also map to an incorrect output under the target ranking. This is shown below.

19. Target ranking
    WSP, *[+s, –l, –a] >> Id(s) >> ML >> MR >> Id(l) >> NoAsp, *V: >> Id(a)

20. Incorrect mapping of **báa**ka with baa set to /–––/
    /baaka/ → **b**$^{\mathbf{h}}$**á**ka

21. Incorrect mapping of pa**záa** with záa set to /–+–/
    /pazáa/ → **p**$^{\mathbf{h}}$**a**za

So default value assignments of unstressed and short given to baa and záa respectively after the contrast pair processing stage will preclude the learner from determining the target language. The two features the learner incorrectly specifies in this language represent two distinct types of errors a learner can encounter.

Section 3.3.2 Two types of missed information

As stated above the algorithm fails to specify correctly two features and the failure to set these two features stems from two distinct types of missed information. The first type is related to the subset/superset problem; there are two different grammars both of which admit all of the observed surface data, the target language and another language which the target language is in a type of subset relationship with. This superset language has a wider range of morphemic behaviors and the learner has no information from the presented forms to distinguish the two. In the incorrect language having zaa be underlyingly unstressed produces the same overt forms as in the target language with záa underlyingly stressed. All of the other morphemes in the target language behave identically to their corresponding morphemes in the superset language and have the same underlying specification. Because of the identical behavior and identical underlying specifications (except for the stress of záa) the learner cannot determine the underlying value of stress from záa from positive data alone.

The second problem is the incorrect specification of length of the morpheme baa. The overt forms given to the learner are not consistent with the morpheme baa being underlyingly specified as short and yet when the learner encounters a contrast pair containing baa there is always a ranking consistent with baa being underlyingly short (and, of course, a ranking consistent with baa being underlyingly long, specifically the target language). The learner has failed to determine that no ranking and lexicon with baa underlyingly short is consistent with all of the overt forms using the technique of testing local lexica for consistency. This technique does not convey any ranking information from contrast pair to contrast pair except when overt forms are fully specified. The learner does not extract any ranking information from a contrast pair and consequently no information besides the lexical information is shared across contrast pairs. It is the lack of discernment of shared ranking information across contrast pairs because the crucial contrast pairs are never fully specified that causes the learner to not set the necessary feature of length for baa.

These two problems, lack of discernment of subset/superset relations and lack of extraction of relevant ranking information present in the contrast pairs (and the full set of overt forms) are presented in detail in the following two sections. An outline of a solution to both problems is also presented. The following chapter will then continue with a full solution to the second problem described here.

Section 3.4 Another language of the system and its relationship to the previous language

The subset problem is usually construed as the problem of selecting between two languages when one of the languages is phonotactically a proper subset of the second language and the data presented is consistent with both languages (Angluin, 1980, Baker, 1979). That is, when all of the overt forms from language A can be overt forms from language B and all overt forms encountered by the learner so far are consistent with language A the learner cannot determine which of the two languages it is attempting to learn using inconsistency detection. Ideally, the learning algorithm would have a bias towards selecting the subset language (Prince and Tesar 1999). Though this is the typical construal of the subset problem, the subset/superset problem appellation also captures other types of subset/superset relations that obtain between languages.

Two languages can have the identical phonotactics and be in a *morphological* subset/superset relationship (Tesar et. al. 2003). In this type of relationship there one language allows more morphological environments. Because the superset language has more morphological environments it also allows more distinctions between the morphemes of the language in these environments. In this way, the subset language contains a subset of morphological environments of the morphological environments of the superset language. The language from the previous section and another language in this system, presented below, are in such a subset/superset relationship. The ranking of the language from the previous section is repeated below.

22. Language L1
WSP, *[+s, –l, –a] >> Id(s) >> ML >> MR >> Id(l) >> NoAsp, *V: >> Id(a)

Language L1 is in an environment subset relationship with the language defined by the ranking given in 20, call this language L2.

23. Language L2
WSP, *[+s, –l, –a] >> Id(s) >> Id(l) >> NoAsp, *V: >> Id(a) >> MR >> ML

The mappings of all of the morphemes in this language are given below.

24. Mappings of L2

/s, l, a/	ka /−−−/	sá /+−−/	zaa /−+X/	rá /+−+/	láa /++X/	xa /−−+/
pa /−−+/	**p^{h}á**ka	pa**s^{h}á**	pa**záa**	pa**r^{h}á**	pa**láa**	pa**x^{h}á**
baa /−+X/	**báa**ka	ba**s^{h}á**	ba**záa**	ba**r^{h}á**	ba**láa**	**báa**xa
tá /+−+/	**t^{h}á**ka	**t^{h}á**sa	**t^{h}á**za	ta**r^{h}á**	ta**láa**	**t^{h}á**xa
dáa /++X/	**dáa**ka	**dáa**sa	**dáa**za	**dáa**ra	da**láa**	**dáa**xa
fa /−−−/	fa**k^{h}á**	fa**s^{h}á**	fa**záa**	fa**r^{h}á**	fa**láa**	fa**x^{h}á**
cá /+−−/	**c^{h}á**ka	ca**s^{h}á**	**c^{h}á**za	ca**r^{h}á**	ca**láa**	**c^{h}á**xa

Language L2 has six distinct roots and six distinct suffixes as shown in the table above. These twelve morphemes give rise to 36 overt forms. Language L2 in many ways is similar to L1. The phonotactics of L2 are identical to the phonotactics of L1. There are four phonotactically distinct overt forms in both L1 and L2: **r**$^{\mathbf{h}}$**á**ra, ra**r**$^{\mathbf{h}}$**á**, **ráa**ra, ra**ráa**. In L2 the constraints WSP and *[+s, –l, –a] are undominated ensuring that unstressed syllables will always be short and that stressed syllables will all either be long or +a (or both). Lexical specification of stress determines the stress placement since Id(s) is only dominated by WSP and *[+s, –l, –a]. The languages diverge in that length alone can determine placement of stress in L2. In L1, underlying specification for length can never influence the placement of stress. This is because ML >> Id(l). In L1, if stress placement hasn't been determined by underlying specification of stress then the alignment constraint ML requires stress to be on the leftmost syllable. Only under stress can a syllable surface as long in L1. L2 is a different matter though. Because Id(l) outranks both alignment constraints and *V:, if stress placement hasn't been determined by underlying specification of stress, then length can determine stress placement. If, for a given overt form, only one of the underlying morphemes is long then Id(l) will prefer to have only that morpheme long. WSP being undominated in L2 will ensure that that morpheme receives stress (as long as Id(s) does not have a conflicting preference). This behavior obtains in **báa**ka which has initial stress even though this language ranks MR above ML. Another key difference between the languages is that in L1 ML outranks MR while in L2 MR outranks ML. So the default stress pattern in L1 is leftmost syllable stressed while in L2 the rightmost syllable is stressed.

25. Language L1
WSP, *[+s, –l, –a] >> Id(s) >> **ML >> MR** >> **Id(l)** >> NoAsp, *V: >> Id(a)

26. Language L2
WSP, *[+s, –l, –a] >> Id(s) >> **Id(l)** >> NoAsp, *V: >> Id(a) >> **MR >> ML**

The key ranking differences stem from the relative ranking of Id(l) with respect to the alignment constraints and the ranking relations between ML and MR. In L1 Align >> Id(l) and hence length cannot determine placement of stress, while in L2 Id(l) >> Align and hence length can determine placement of stress. Default stress placement is reversed in L1 and L2 because in L1 ML >> MR while in L2 MR >> ML.

Though the two languages differ significantly in their morphemic behavior they have an interesting property. There is an injective mapping from the morphemes in language L1 to the morphemes in language L2 such that the overt forms produced by L1 are identical to the overt forms produced by L2 of the L1 morpheme's images under the injective mapping. That is, there is a way to associate all of the morphemes in L1 with morphemes in L2 so that any combination of root and suffix of these identified morphemes will have the same overt form under the mapping of L1 or L2. The mappings of L1 and L2 are repeated below to illustrate this phenomenon.

27. Mappings of morphemes in L1

/s, l, a/	ka /−XX/	sá /+−X/	záa /++X/
pa /−−X/	**p^{h}á**ka	pa**s^{h}á**	pa**záa**
baa /−+X/	**báa**ka	ba**s^{h}á**	ba**záa**
tá /+−X/	**t^{h}á**ka	**t^{h}á**sa	**t^{h}á**za
dáa /++X/	**dáa**ka	**dáa**sa	**dáa**za

28. Mappings of morphemes in L2

/s, l, a/	ka /−−−/	sá /+−−/	zaa /−+X/		rá /+−+/	láa /++X/	xa /−−+/
pa /−−+/	**p^{h}á**ka	pa**s^{h}á**	pa**záa**		pa**r^{h}á**	pa**láa**	pa**x^{h}á**
baa /−+X/	**báa**ka	ba**s^{h}á**	ba**záa**		ba**r^{h}á**	ba**láa**	**báa**xa
tá /+−+/	**t^{h}á**ka	**t^{h}á**sa	**t^{h}á**za		ta**r^{h}á**	ta**láa**	**t^{h}á**xa
dáa /++X/	**dáa**ka	**dáa**sa	**dáa**za		**dáa**ra	da**láa**	**dáa**xa
fa /−−−/	fa**k^{h}á**	fa**s^{h}á**	fa**záa**		fa**r^{h}á**	fa**láa**	fa**x^{h}á**
cá /+−−/	**c^{h}á**ka	ca**s^{h}á**	**c^{h}á**za		ca**r^{h}á**	ca**láa**	**c^{h}á**xa

Language L2 exhibits exactly the same distribution for morphemes pa, baa, tá, and dáa and for ka, sá, záa as does L1. That is, by identifying pa in L1 with pa in L2 and ka in L1 with ka in L2 and similarly for baa, tá, and dáa and ka, sá, and záa (záa in L1 is identified with the nearly orthographically identical zaa in L2) the overt form for any choice of root and suffix in L1 will be the same as the overt form for the associated root and suffix in L2. So, for example, /baa + záa/ → ba**záa** in L1 and /baa + zaa/ → ba**záa** in L2.

It is important to note that this behavior of identical overt forms under this morphemic identification occurs because there is *morphemic* identity and only morphemic identity. These morphemes can be identified, producing the identical outputs, but it is not the case that their corresponding underlying forms are identical. Nor in fact is it possible for their underlying forms to be identical in all cases. There are three types of morphemic correspondences, those morphemes that are in correspondence with morphemes that have the same underlying form, those that are in a subset relationship with their corresponding morpheme, and those that are different and cannot be in a subset relationship with their corresponding morpheme.

So first, some of the identified morphemes from language L1 and L2 may in fact be identical. For example, baa in L1 may have either of the following two underlying forms: /−+−/ or /−++/. The corresponding morpheme baa in L2 may have exactly one these two underlying forms also. Whether they have exactly the same underlying form after learning does not matter, what matters here is that they have an identical range of potential underlying forms.

Now pa in L1 can be said to have a greater range of underlying forms than its corresponding morpheme pa in L2. So, pa in L1 may have either of the following two underlying forms: /−−−/ or /−−+/. For any given suffix, with either underlying form, L1 maps pa to exactly the same overt form. Now its associated root pa in L2 must be /−−+/. The larger set of potential underlying forms for pa in L1 is a result of a contrast in L2 that is not present in L1. In L2 an unstressed, short, aspirated segment, pa, in the environment of an unstressed, short, and unaspirated segment, ka, surfaces stressed and aspirated as the overt form **p^{h}á**ka demonstrates. In L2 an unaspirated, short, unstressed root in the same environment does not receive stress as the form fa**k^{h}á** shows. L2 is contrastive with respect to aspiration in this environment while L1 is not. This causes the underlying form subset/superset relationship to obtain.

Finally, the corresponding morphemes may have necessarily different underlying forms. This is the case for the morpheme záa in L1 and its corresponding morpheme zaa in L2. The suffix záa in L1 must be stressed and long. Its aspiration value is immaterial under the ranking of L1. Its associated morpheme in L2, zaa, must in fact be *unstressed* and long (and its aspiration value is also immaterial). This is because length is contrastive in an underlyingly unstressed suffix in L2 while in L1 it is not. In L1 length is not contrastive in underlyingly unstressed suffixes since ML dominates both MR and Id(l) and since undominated WSP precludes length in an unstressed position. In L2 length is contrastive in underlyingly unstressed suffixes since Id(l) dominates the alignment constraints.

So the language L1 can have its morphemes identified with the morphemes in L2 and have the same distributional pattern. Furthermore, all of the underlying forms in L1 can be identical to the underlying forms which they are associated with in L2 with the exception of precisely one feature of one form, the stress feature of the suffix záa. This is the reason that the stress feature of záa is not and cannot be set by the algorithm presented above. Even if all other features are set correctly by a learning algorithm given the data of language L1 the learner, attending only to the overt forms, cannot determine from inconsistency detection alone which of the two languages, L1 or L2, is generating the attested overt forms. This is not the fault of contrast pairs; all of the given forms will fail to yield information sufficient to determine the target language because the forms themselves are ambiguous between L1 and L2. It is a limit of inconsistency detection.

Section 3.5 Towards a solution of the restrictiveness issue

The distinction between the overt environments that a morpheme appears in and the necessary underlying specifications of those overt environments appears to lead to a potential solution to the restrictiveness issue articulated above. In this section I will discuss an approach that assumes that non-alternating forms are in fact non-alternating even in unattested environments. This will allow the learner to restrict possible outputs for unattested forms allowing the learner to choose a more restrictive language. This approach may lead to the learner being able to distinguish these two languages. The full investigation of these ideas will be left to later work.

Neutralization of stress and length occur in suffixes in the context of an underlyingly stressed root in language L1. It is this contextually dependent neutralization that has not been learned and knowledge of which would distinguish for the learner the languages L1 and L2. Consider the non-alternating morphemes tá and dáa in language L1.

Morphemes ka, tá, and dáa in L1 do not appear in as many environments as in L2. These environments are defined by the features that surface overtly. So, for example, tá and dáa only appear in the environment [sa] and not in [**s**^{h}**á**] or [**sáa**]. In the above approach of processing contrast pairs the learner has determined that the underlying values of the three suffixes after the processing of contrast pairs are those given below.

29. Underlying values of the suffixes in L1 after the processing of contrast pairs.

	/s l a/
ka	/−−−/
sá	/+−?/
záa	/?+−/

The suffix ka is underlyingly unstressed short and –a; sá is stressed short and the aspiration value is undetermined; zá is long, –a, and its stress value is undetermined. Recall that these specified underlying values for ka, sá, and záa (except for the a values of ka and záa) are precisely those that are necessarily entailed by the overt forms of language L1. The learner then restricts attention to only those forms whose underlying specification is complete, namely ka. The suffix ka combined with roots that are completely specified yield ranking information that the learner then further uses to restrict potential underlying forms. Because the learner only uses forms that are fully specified to restrict local lexicon searches the underlying specification of stress for záa will remain unset. Indeed, roots tá and dáa only appear in the overt environment [sa] and since ka is the only suffix that is fully specified the learner only gains ranking information by considering tá and dáa in the underlying environment defined by /ka/ (which is to say with the morpheme that has the underlying specification of unstressed, short, and –a). If the learner knew the behavior of tá and dáa in underlying environments /s^{h}á/, /sáa/, then it would be possible for the learner to distinguish the languages L1 and L2. This is because under L1 /tá + s^{h}á/ → **t^{h}á**sa and hence the learner would know that ML >> MR since these two morphemes would have the same underlying form. This ranking information would be captured in the winner-loser pair given below.

30. Ranking information gained from previously unattested mapping

/tá+ s^{h}á/	WSP	*[+s-l-a]	NoAsp	*V:	ML	MR	Id(s)	Id(l)	Id(a)
t^{h}ása ~ tas**^{h}á**					W	L			

So if the learner could have access to the behavior of these morphemes in these (underlying) environments the languages L1 and L2 could be distinguished. A problem arises though in determining what the behavior of these morphemes would be in these environments. The linguist can see that in L1 the roots tá and dáa do not alternate and that the output of /tá + s^{h}á/ would be **t^{h}á**sa. The learner on the other hand does not have this information. The learner does know though that roots tá and dáa do not alternate. This may be the crucial piece of information the learner could use to further restrict lexical possibilities for other morphemes. The learner could assume that non-alternating morphemes in fact do not alternate in any environment either attested or unattested. This assumption, that non-alternating morphemes do not alternate in unattested environments is already implicitly made. Initial lexicon construction requires that non-alternating morphemes truly do not alternate, else the initial specifications of the non-alternating features could be set incorrectly. By assuming that non-alternating morphemes do not alternate ever the learner has an anchor to explore unattested underlying environments for these non-alternating morphemes.

Though the learner would know what the output of the non-alternating morpheme is the output for the environment morphemes is still not known. So, for example, tá does not alternate and hence the learner would know that tá in combination with the morpheme /s^{h}á/ would surface as [**t^{h}á**]. What the learner would not know is what the morpheme s^{h}á would surface as. The linguist knows that under L1 it must surface as [sa] but the learner would not. What the learner would know is that since tá surfaces as [**t^{h}á]**, the two morphemes /tá + s^{h}á/ must surface as either **t^{h}á**sa or **t^{h}ás**^{h}a. That is, the suffix must be short and unstressed and either +a or –a. The learner knows this because all other output forms for this input that have the root stressed are harmonically bounded by these two output forms. By restricting the output of a particular morpheme, namely tá in this case, further restrictions are placed on the morphemes whose output correspondents are not known. These potential input-output mappings then could lead to further restrictions on the ranking if the learner had a means of extracting shared ranking information across potential input-output mappings. The next chapter develops just such a technique.

The approach outlined here, while no means an algorithm, suggests an approach to determining further ranking information. Non-alternating forms are assumed to be in fact non-alternating. This assumption allows the learner to restrict potential outputs for unattested and possibly alternating forms. This restriction of potential input-output mappings allows the learner to gain ranking information from forms that are not attested. A full investigation of this approach needs to address whether in general an efficient way of generating both the unattested underlying forms and unattested overt forms exists.

Section 3.6.1 Failure to extract ranking information

The morphological subset/superset issue for the learner learning language L1 stemmed from the learner being unable to distinguish between L1 and L2, where L1 is more restrictive than the other. This caused the learner to fail to set the feature for stress for záa. The learner also failed to set the featural value of length for the root baa. This did not stem from a failure to choose the most restrictive language but from a failure to extract necessary ranking information from the given overt forms of language L1. Below discusses the types of ranking information that the learner fails to extract and demonstrates this failure in one of the contrast pairs the learner is attempting to extract information from.

The forms presented to the learner require that the length value of baa be long; the learner has failed to reach this conclusion though. The information the learner is attending to in the contrast pairs does not necessitate this featural specification. The learner while processing contrast pairs is only attending to the consistency of local lexica. There is more information though present in the contrast pairs. Failure to set the length feature of baa comes about because the learner is not attending to ranking information that is present in the contrast pairs that are not fully specified.

In each contrast pair in which baa occurs, baa's length value remains unset. This means that for each contrast pair there are (at least) two local lexica, one with baa's length short and one with baa's length long, that are consistent. In each case there are two types of ranking solutions for the incorrect setting of length (rankings that produce the correct output but have baa length set to short). The two ranking solutions require rankings that are not allowed in language L1. The two (incorrect) solutions require that either Id(a) outrank NoAsp or require that Id(l) outrank ML and MR (and some of the incorrect solutions require both). Language L1 requires both NoAsp >> Id(a) and ML, MR >> Id(l), but in the contrast pairs that contain baa this required ranking information is not present. This lack of ranking information prevents the learner from determining that the underlying value of short for baa is inconsistent with the remaining overt forms.

The only time Id(l) >> ML, MR is required is when a contrast pair contains both záa and baa. If záa's stress were set correctly to + then the non-target language ranking of Id(l) >> ML, MR would not be required. Focusing on baa here, in every consistent local lexicon for a contrast pair containing baa in which baa's length is short it is the case that a ranking requirement of Id(a) >> NoAsp is imposed. Exhaustive investigation of each contrast pair containing baa has demonstrated this. Below the case of the contrast pair **<báa**ka, **t^{h}á**ka**>** is investigated and shown to have these properties.

Section 3.6.2 Failure to set required features on a contrast pair

Under the target ranking L1, /báa + ka/ → **báa**ka and /tá + ka/ → **t^{h}á**ka. The underlying specifications for the three morphemes in this contrast pair as set by the algorithm by the end of the contrast pair processing stage are given below.

31. Set features of morphemes in contrast pair <**báa**ka, **t^{h}á**ka>

	/s l a/
baa	/–?–/
tá	/+–+/

	/s l a/
ka	/–––/

All the featural values of these three morphemes are specified with the exception of the length of baa and hence there are two local lexica for the learner to check for consistency. The local lexica with baa's length set to long are consistent under the target ranking of L1. The local lexica with baa's length set to short are consistent with the ranking given below (amongst others).

32. WSP, *[+s, −l, −a] >> Id(s) >>ML >> MR >> Id(a) >> *V:, NoAsp >> Id(l)

The fact that baa surfaces long and tá surfaces short in the same environment means that an underlying difference in their specifications must be accounting for the difference. Stress cannot be the feature that explains the difference since they both surface stressed. If two morphemes surface stressed in the same environment but differ on aspiration and length, underlying stress will never account for that difference. This is because no constraint says that if underlyingly stressed behave one way (while stressed) and behave another way if underlyingly unstressed (while stressed), nor do any combination of the constraints. This is the type of constraint interaction that what would be needed for aspiration and length to surface differently under stress because of a differing underlying specification of stress. So either aspiration or length must be different for the two root morphemes. First consider the case were baa is underlyingly short.

Assuming that baa is short and −a and it surfaces long, either Id(a) or NoAsp must dominate both Id(l) and *:V. Now tá surfaces as +a and short and therefore Id(a) must dominate NoAsp, Id(l), and *:V. I.e. Id(a) >> NoAsp, *+l, Id(l). This subranking holds because baa is underlyingly short (and surfaces long) and tá is underlyingly +a (and surfaces +a). Because *[+s, –l, –a] is undominated, under stress a morpheme must either be +a or +l, the differing "repair" mechanisms for this constraint must come from differing underlying specifications. So, baa being underlyingly short requires the subranking shown below.

33. Id(a) >> NoAsp, *:V, Id(l)

Now, in the target language there is a crucial ordering between these four constraints that conflicts with the ranking in 29. This subranking is given in 30.

34. Id(l) >> NoAsp, *:V >> Id(a)

In fact, in every contrast pair containing baa where baa surfaces stressed baa can be specified *short* provided the following the incorrect subranking of 29 above is allowed. So if the learner knew that the ranking in 30 obtained for the target language the conflicting ranking in 29 would not be allowed and baa's length would be set correctly.

These ranking requirements are in fact imposed by other contrast pairs but because the morphemes of the contrast pairs that require the ranking in 30 are not fully specified these ranking requirements are not imposed on other contrast pairs. It is the lack of discernment of these requirements while processing the contrast pairs that causes the failure to set the length of baa. In the next chapter I will develop a technique for extracting ranking information from the contrast pairs even when the morphemes of the contrast pairs are not fully set. This ranking information then can be used in a slightly modified version of the learning algorithm presented in chapter 2 to set the length feature of baa.

Chapter 4. The Join

Attending to consistent local lexica for a given contrast pair allows the learner to learn a significant amount of information about the target lexicon. But as seen in the last chapter this approach missed some pertinent information that would be available to the learner by attending to more overt forms simultaneously because ranking information imposed by more than two overt forms is not imposed by any given contrast pair. In this chapter an approach is given that maintains the learner's focus on contrast pairs only but extracts further ranking information. By attending to not just consistent local lexica for a contrast pair but the ranking restrictions imposed by those consistent local lexica the learner can glean ranking information about the target ranking that restricts which local lexica are consistent for further contrast pairs.

Checking for consistency of local lexica produces a set of winner-loser pairs for each local lexicon. These winner-loser pairs represent ranking restrictions imposed by the input-output mapping defined by that local lexicon. For a given contrast pair one of the local lexica is correct and hence any ranking information that is shared across consistent local lexica is shared with the correct local lexicon. Enabling the learner to extract this shared ranking information will allow the learner to learn ranking information about the target language. This ranking information is gained from the consistent local lexica even though the learner does not necessarily have enough information to determine which of the consistent local lexica is the correct one. In this way the learner can extract ranking information from possible input-output mappings.

Ranking information imposed by a local lexicon is generated through error-driven learning using MRCD. This ranking information is captured by the winner-loser pairs that the error-driven learning generates and these winner-loser pairs themselves represent the ranking information as ERCs.

Each consistent local lexicon can potentially generate a non-empty set of ERCs. From these sets the learner may be able to extract ranking information. Each set of ERCs, one for each consistent local lexicon, imposes its own ranking restrictions and these ranking restrictions may be in conflict with other restrictions placed by other consistent local lexica. Even though ranking information across local lexica may be contradictory is it possible to determine precisely what ranking conditions satisfy all of the local lexica's ranking restrictions.

Section 4.1. Fusion

A tool that will be used throughout this section is that of fusion (Prince 2002). Fusion is a binary operator on the set of ERCs for a given constraint set mapping into that set of ERCs that has many properties that are supremely useful for reasoning about rankings. Two such properties are that the fusion of two ERCs is entailed by each of the fusands and that two ERCs that have contradictory ranking requirements will fuse to an ERC consisting only of Ls and es signifying the infelicitousness of the conjunction of the two ERCs.

The fusion operator is defined component-wise over the set {W, e, L}. Define the component-wise fusion of any two elements of the set {W, e, L} as follows where the operator ‘o’ signifies the component-wise fusion.

1. Definition of fusion over set {W, e, L}
    - W o e = e o W = W
    - W o L = L o W = L
    - e o L = L o e = L
    - X o X = X (where ‘X’ means any of {W, e, L})

So fusion over the set {W, e, L} is commutative, idempotent, and associative. This definition stems from placing a three-valued logic on the set {W, e, L} where L entails e and e entails W and then for two elements in {W, e, L} the fusion of the two elements is the strongest entailment-wise element of the two. So, one can view L as dominant since it entails all other elements and hence L fuse anything will always be L.

This above fusion is defined only on the set {W, e, L} while what is desired is fusion on two ERCs. Fusion of two ERCs then is defined as the component-wise fusion of the ERCs. So in the following example in the table below with the ERCs <W, L, W, W> and <W, e, L, e>, their fusion is <W, L, L, e> since the component-wise fusion of W and W is W (for C1), L o e = L (for C2), W o L = L (for C3), and W o e (for C4).

2. Fusion of two ERCs

	C1	C2	C3	C4
ERC 1	W	L	W	W
ERC 2	W	e	L	e
1 o 2	W	L	L	W

Beyond having the properties that the fusion of any two ERCs is entailed by both ERCs and that inconsistent ERCs fuse to L^+, fusion allows for the construction of a set of ERCs' fusional closure. The fusional closure of an ERC set will be described and discussed in detail later in this chapter but its most useful characteristic (for present purposes) is that any ERC entailed by the original set is entailed by a single ERC in the set's fusional closure.

Section 4.2. Overview of ranking extraction algorithm

An overview of the algorithm to extract shared ranking information is given here. The algorithm consists of two parts, first, a processing stage on the ERCs generated from testing for consistency of the local lexica and then, second, extraction of shared ranking information across the processed sets of ERCs. The first stage, the processing of ERC sets, to be described in detail in a section below, is necessary to ensure that the extraction of shared information in the second stage is maximally informative. The second stage of extraction of shared ranking information has the property that if there is ranking information to be extracted from a completely arbitrary set of ERC sets it will extract *some* shared ranking information but not necessarily *all* of the shared ranking information. The first stage modifies the ERC sets in such a way as to guarantee that the second stage extracts all shared ranking information. Without this modification, under certain circumstances, the second stage of the algorithm will only extract some ranking information if there is any to be extracted.

The first stage of the algorithm adds to each ERC set its fusional closure. The fusional closure of an ERC set has the property that it entails the same set of ERCs as the original set and for each ERC entailed by the original set there is at least one ERC in the fusional closure of that set that entails this ERC. These properties of the fusional closure are investigated in more depth in later sections.

The second stage of the algorithm works over the set of fusional closures of the original ERC sets. The algorithm constructs a set of ERCs by attending to each constraint in sequence. For each constraint the algorithm determines whether there is an ERC in each fusional closure set that has an L in that constraint. If so, then each ERC set requires that this constraint be dominated by some set of constraints. Having an L in a constraint means that the ERC (and consequently its ERC set) necessarily requires that that constraint be dominated by some set of constraints. Each ERC set having an ERC with an L in a constraint means that *all* of the ERC sets require that constraint to be dominated by some set of constraints (possibly different for each ERC set). The algorithm is designed to extract precisely those constraints that must dominate this given constraint.

So, for the given constraint that has an ERC with an L for that constraint in each fusional closure the algorithm selects one of these ERCs with an L from each fusional closure set (there may be more than one) and determines what these ERCs jointly entail. This is repeated for each combination of ERCs that assign an L to the given constraint. In this way the algorithm determines precisely which constraints must dominate the given constraint. This procedure is then repeated for each of the constraints in the constraint set. An overview of the algorithm is given below.

3. Shared ranking extraction algorithm

```
For a set of ERC sets
    Stage 1: Construct the fusional closure of each of the ERC sets
    Stage 2: For each constraint
        • Determine if there is at least one ERC with an L for that
          constraint in each fusional closure set
        • If so, determine the shared ranking information of the
          collection of ERCs with an L in that constraint
```

Stage 2, the stage that extracts the shared ranking information from the fusional closure sets is discussed in detail in the following sections. Stage 1, the stage the processes the ERC sets ensuring that stage 2 extracts all relevant ranking information is discussed after the discussion of stage 2.

Section 4.3. Ranking information to be extracted across consistent local lexica

For a given contrast pair often multiple local lexica are consistent and these consistent local lexica for the contrast pair entail sets of required ranking restrictions. These ranking restrictions may be contradictory; that is, one consistent local lexicon for a contrast pair may require that constraint A outrank constraint B while another consistent local lexicon may require that constraint B outrank constraint A. Even when two local lexica impose conflicting ranking requirements it is possible that there are certain requirements that are shared by both and these shared requirements may be extracted from the contradictory ranking restrictions. Though, of course, conflicting ranking information is by no means required for information to be extracted from two local lexica.

In the following example two sets of ERCs are given that have contradictory ranking requirements and yet contain shared ranking information.

4. ERC Set 1

	C1	C2	C3
ERC1	W	L	e
ERC2	e	W	L

5. ERC Set 2

	C1	C2	C3
ERC3	W	e	L
ERC4	e	L	W

ERC Set 1 contains two ERCs, the first of which requires that C1 >> C2 and the second requires that C2 >> C3. These two requirements impose the total order on the three constraints of C1 >> C2 >> C3. ERC Set 2 also contains two ERCS, the first requiring C1 >> C3 and the second that C3 >> C2. These two ERCs impose the total order C1 >> C3 >> C2.

These two ERC sets impose conflicting ranking requirements. Specifically the first requires that C2 >> C3 while the second requires that C3 >> C2. These conflicting requirements, while a product of the total orders imposed by both set of ERCs, are directly represented in the ERCs generated from their respective local lexica. In the tableaux below this conflict is shown. ERC 2 from ERC Set 1 fused with ERC 4 from ERC Set 2 is given in the tableaux and clearly shows that these requirements are contradictory.

6. Contradictory ranking requirements

	C1	C2	C3
ERC2	e	W	L
ERC4	e	L	W
2 o 4	e	L	L

While the two local lexica impose conflicting ranking requirements they share certain ranking requirements. Both ERC sets require that C1 dominate both C2 and C3; they only disagree on the relative rankings of C2 and C3. Using the approach described below the learner can extract the shared ranking requirement that C1 dominate both C2 and C3.

Section 4.4. The component-wise join

Ranking information is represented by ERCs and the methods of determining shared ranking information across ERC sets described herein apply to ERCs and ERC sets. But before defining an operator and an algorithm to extract shared ranking information from ERCs an operator is defined on the components of ERCs. This section defines a binary operator over the set {W, e, L}, the constituent components of ERCs. This operator is necessary to determine an operator over ERCs, the join of two ERCs, and ultimately to extract shared ranking information from a set of ERCs.

Define the binary operation $\oplus$ from {W, e, L} x {W, e, L} $\rightarrow$ {W, e, L} by the following operations.

7. $W \oplus X = X \oplus W = W$ (where X is any member of {W, e, L})
8. $X \oplus X = X$
9. $L \oplus e = e \oplus L = e$

So, W is 'dominant', i.e. W join anything is W, the operator is idempotent and L is identity. This operator is both commutative and associative; these facts follow directly from the definition.

If one places the order on the elements of the set {L, e, W}, $L < e < W$, then this operator can be seen as producing the maximal element of any two elements from the set {L, e, W}. That is, the join of two elements of {L, e, W} is the greatest value of the two elements according to the order $L < e < W$. So W join anything is W since W is the maximal element of the order, while L join any element is that other element since L is the least element of the order. The ordering $L < e < W$ places a (quite trivial) lattice structure on the set {L, e, W} and consequently one can define the join of a set of elements of the lattice. The join, j, of a set S of elements of a lattice is that element that has the property that first, every element, s of S is less than or equal to j, and second, that for any element x such that $s \leq x$ for all $s \in S$, it is the case that $j \leq x$. Now, the order given above over the set {L, e, W} defines an admittedly simple lattice structure; it is the case that the join operator, ⊕, gives precisely the lattice-theoretic join of any two elements. It gives exactly the largest element of two elements and no larger according to the order given above. The fact that this is the join of this simple lattice will find an exact parallel when the lattice of ERC entailment is constructed below.

Section 4.5. The join of ERCs

In the section above a binary operator was defined over the set of {W, e, L}. This operator is needed to define a binary operator over ERCs that will lead to a method of determining the shared ranking information over a set of set of ERCs.

Define a binary operator on the set of ERCs as follows: for two ERCs x and y define the join, $x \oplus y$, of the two ERCs as that ERC that has its k^{th} column as the component-wise join of the k^{th} column of x and the k^{th} column of y.[7] This is the composite order. For an ERC x, let x[k] represent the value from the set {W, e, L} that x has in column k. Then the join of two ERCs, x and y, defined as $x \oplus y$, is that ERC where $(x \oplus y)[k] = x[k] \oplus y[k]$. With this definition the operator ⊕ has been overloaded, meaning that the join operator, ⊕, has a component-wise meaning as a binary operator on the set {W, e, L} and

[7] This operator is equivalent to the Lukasiewicz-Kleene-RM3 3-valued 'or', as mentioned in Prince 2002, p51, example 99.

an ERC-wise meaning on the set of ERCs. An example of the join ERC operator is given below.

10. Join of two ERCs

		C1	C2	C3	C4
1	ERC1	W	L	e	L
2	ERC2	e	W	L	L
3	1 ⊕ 2	W	W	e	L

In the table above the join of two ERCs is given. Row 3 shows the join of the ERCs from row 1 and 2; the join of ERC1 and ERC2, ERC1 ⊕ ERC2 is represented in the table with the notation 1 ⊕ 2. In computing the value of (ERC1 ⊕ ERC2)[C1], note that ERC1[C1] is W and ERC2[C1] = e. Hence (ERC1 ⊕ ERC2)[C1] = ERC1[C1] ⊕ ERC2[C2] = W ⊕ e = W. For C3, (ERC1 ⊕ ERC2)[C3] = ERC1[C3] ⊕ ERC2[C3] = e ⊕ L = e. The constraints C2 and C4 following similarly, fully defining the ERC1 ⊕ ERC2. While the symbol ⊕ is overloaded and means two things (component-wise join and ERC-wise join) in these equations it is not being used ambiguously. Each instance of the symbol has at most one interpretation depending on the context.

The ERC join has some of the same properties as the component-wise join which follow directly from its definition using the component-wise join. The ERC join is commutative and associative. These facts follow directly from the definition. Most importantly, for the lattice over the set of ERCs imposed by the component-wise lattice of the join is the entailment lattice-theoretic join. This is proven below.

ERC entailment places a partial order on the set of ERCs. Given two ERCs x and y this order is given by defining $x < y$ if and only if x entails y. This partial order defines a lattice over the set of ERCs and given any two ERCs x and y, $x \oplus y$ is the lattice-theoretic join of x and y over the lattice defined by ERC entailment.

Claim: for two ERCs x and y, $x \oplus y$ is the join of x and y over the lattice defined by ERC entailment.

Proof: Let j be the join of x and y. Then j has the property that $x \rightarrow j$ and $y \rightarrow j$ and if there is an ERC w such that $x \rightarrow w$ and $y \rightarrow w$, then $j \rightarrow w$. First I want to show that $x \rightarrow x \oplus y$ and that $y \rightarrow x \oplus y$. Consider a column k of x, x[k], that contains an L. Then $(x \oplus y)[k]$ contains either an L, e, or W depending on y[k]. If x[k] contains an e, then $(x \oplus y)[k]$ contains either an e or W by the operations of $\oplus$. And finally, if x[k] contains a W, then $(x \oplus y)[k]$ contains only a W by the operations of $\oplus$. But these values of $(x \oplus y)[k]$ are precisely those allowed under the operations of L-retraction and W-extension. Because the resulting ERC derived from L-retraction and W-extension is always implied by the original ERC this shows that $x \rightarrow x \oplus y$. By symmetry, $y \rightarrow x \oplus y$.

Now suppose that w is such that $x \rightarrow w$ and $y \rightarrow w$. Showing that $x \oplus y \rightarrow w$ will prove that $x \oplus y = j$. Now, every ERC entailed by x is a result of some (or no) applications of L-retraction and W-extension operations. So, consider w[k]. Suppose w[k] = L. Then x[k] = L and y[k] = L since w[k] must be a product of L-retraction, W-extensions, or inaction. Hence, $(x \oplus y)[k] = L$. Suppose $w[k] = e$. Then $x[k] = L$ or e and $y[k] = L$ or e and hence $(x \oplus y)[k] = L$ or e. Finally, suppose $w[k] = W$. By definition of what an ERC is, $(x \oplus y)[k] = L$, e, or W. But this means that for each $w[k]$ it is an L-retractions and/or W-extension of $(x \oplus y)[k]$ (or is identical to it). And so $x \oplus y \rightarrow w$. Therefore $x \oplus y$ is the join of x and y over the lattice defined by ERC entailment. Q.E.D.

Initially we defined the join operator, $\oplus$, over the set {W, e, L}. This last section has defined the join operator, $\oplus$, over pairs of ERCs. The symbol, $\oplus$, here is overloaded; it means either joining over {W, e, L} or joining of two ERCs. The joining of two ERCs is not sufficient for our purposes though. We will need an operation to extract ranking information over sets of ERCs that contain more than one element.

Section 4.5.1 Joining can lose information over more than two ERCs

The join operator for two ERCs is a useful computational tool but it can fall short by itself as a tool the learner uses for extracting useful ranking information. This is because the join operator as defined above produces the join for *two* ERCs while local lexica often impose ranking restrictions that are necessarily captured not by one ERC but by a set of ERCs. Joining all of the ERCs from a set of ERC sets will produce the join of each of the individual ERCs and may lose shared ranking information, since ranking information entailed by the set of ERCs created from error-driven learning for a given local lexicon will often be captured by conjunctions of ERCs in that sets. Taking the join of a set of ERCs defined by a local lexicon will lose these relationships and only produce the one ERC that is the join of all the ERCs in the set. If these independent ranking restrictions are shared with ranking restrictions imposed by other local lexica then the ERC join may lose some of this shared ranking information. This is demonstrated in the two local lexica presented at the beginning of this chapter which are repeated below.

11. ERC Set 1

	C1	C2	C3
ERC1	W	L	e
ERC2	e	W	L

12. ERC Set 2

	C1	C2	C3
ERC3	W	e	L
ERC4	e	L	W

ERC Set 1 imposes the total order C1 >> C2 >> C3 while ERC Set 2 imposes the order C1 >> C3 >> C2 and both share the ranking requirement that C1 dominate both C2 and C3. Yet the result of taking the join of all four ERCs is uninformative. It yields an ERC with Ws in each column, a trivially true ERC. This computation is shown in the following tableaux.

13. The join of all four ERCs

	C1	C2	C3
ERC1	W	L	e
ERC2	e	W	L
ERC3	W	e	L
ERC4	e	L	W
$1 \oplus 2 \oplus 3 \oplus 4$	W	W	W

No ranking information is gained by joining all of the ERCs from the two local lexica. ERC2 requires only that C2 dominate C3 while ERC4 requires that C3 dominate C2. When two ERCs have precisely opposite ranking requirements the join will contain no informative ranking information. In general, when these types of opposite ranking requirements do not hold, some information may be gained but it is not guaranteed. In fact in many instances all information about shared ranking requirements will be lost.

Section 4.5.2 Extracting joint ranking information

The algorithm presented here produces a set of ERCs from sets of ERCs. This set is entailed by each of the sets of ERCs. Because each of the sets of ERCs entails the set of ERCs produced by this algorithm this set captures shared ranking information across the ERC sets. The set constructed will always produce some shared ranking information when there is shared ranking information to be obtained. If the ERC sets that the algorithm is applied to have the property that any ERC entailed by the set is entailed by at least one individual member of the set then the algorithm is guaranteed to produce all shared ranking information. This property of individual ERC entailment does not hold for many ERC sets, though crucially it holds for the fusional closure of an ERC set.

For a set of ERC sets the algorithm determines for each constraint whether the sets jointly require that that constraint must be dominated. If so, the algorithm determines which constraints must dominate the given constraint. It does this by taking an ERC with an L in the given constraint from each of the ERC sets and joining these ERCs. Then, this process is repeated for all combinations of ERCs with an L in the given constraint where one ERC is chosen from each of the ERC sets. In this way independence of the ERCs in a given ERC set are disentangled. This process is repeated for each constraint. The resulting set captures all of the shared ranking requirements of the given ERC sets (if the given ERC sets have the fusional closure property described above). A high-level description of the algorithm is given below before the particulars are described.

14. Shared ranking extraction algorithm (stage 2)

- For a set of ERC sets
- For each constraint having an ERC in each of the sets with an L for that given constraint
    - For every combination of one ERC from each ERC set where the ERC has an L in the given constraint
    - Join these ERCs using the join operator, $\oplus$

The next section will precisely describe the steps outlined above.

Section 4.6. Shared ranking extraction algorithm

Turning now to the specifics of the algorithm, let $\{E_i\}$ be a set of ERC sets. This is the set of ERC sets that the algorithm will extract shared ranking information from. Suppose there are n constraints. For each constraint C_k, where $1 \leq k \leq n$, the algorithm determines whether there is at least one ERC in each of the ERC sets $\{E_i\}$ that contains an L in constraint C_k. If, for a given constraint C_k, there is at least one ERC in each ERC set that has an L in constraint C_k the algorithm selects exactly one ERC with an L in constraint C_k

from each ERC set. The algorithm then joins these selected ERCs using the join operator, ⊕. The resultant ERC is ranking information jointly entailed by all of the ERC sets. There may be more than one ERC in any given ERC set that contains an L in constraint C_k and so the algorithm continues by selecting one ERC with an L in constraint C_k from each ERC set and joining them together, doing this until all combinations of possible ERCs have been selected and joined. Those joined together must each have an L in constraint C_k and there must be exactly one from each ERC set. This is necessary to avoid joining two ERCs from the same ERC set with the resultant potential loss of information. This joining procedure is then repeated for each constraint producing a set of ERCs that captures the shared ranking information of the original ERC sets as will be proven later in this chapter.

Section 4.6.1 Algorithm applied

In this section the ranking extraction algorithm is applied to a set of ERC sets. Given below are two sets of ERCs.

15. ERC Set 1

	C1	C2	C3
ERC1	W	L	L
ERC2	e	W	L

16. ERC Set 2

	C1	C2	C3
ERC3	W	e	L

As can be deduced the shared ranking information across these two ERC sets is represented by the ERC <W, e, L>. ERC Set 2 contains only this ERC and therefore obviously entails <W, e, L> while ERC Set 1 contains <W, L, L> and therefore also clearly entails <W, e, L>.

The algorithm proceeds by first attending to constraint C1. In neither ERC Set is there an ERC with an L in C1 and hence the algorithm produces no ERCs while attending to C1. The algorithm then moves on to C2. ERC Set 1 has one ERC with an L in C2 but ERC Set 2 does not have an ERC with an L in C2. The algorithm requires that each ERC Set has an ERC with an L in the relevant constraint and here ERC Set 2 does not. So the algorithm produces no ERCs from attending to constraint C2 and so moves on to constraint C3.

Both ERC Set 1 and ERC Set 2 have at least one ERC with an L in constraint C3 and so the algorithm selects one ERC from each of the ERC Set. Selecting ERC 1 and ERC 3 the algorithm joins these two ERCs together.

17. ERC Set 1 Result of joining ERC 1 and ERC3 together

	C1	C2	C3
ERC1	W	L	L
ERC3	W	e	L
$1 \oplus 3$	W	e	L

Now each ERC with an L in C3 must be joined with every other ERC with an L in C3 that is not in the same ERC Set. So ERC 2 from ERC Set 1 has an L in C3 and hence must be joined with ERC 3 from ERC Set 2.

18. Result of joining ERC 2 and ERC3 together

	C1	C2	C3
ERC2	e	W	L
ERC3	W	e	L
$2 \oplus 3$	W	W	L

The joining of ERC 1 with ERC 2 which are both from ERC Set 1 is not necessary because they are from the same local lexicon. It is sufficient to join these ERCs individually with the ERCs from the other ERC set to extract the shared ranking information.

The two ERCs produced, <W, e, L> and <W, W, L> clearly entail the shared ranking information from the two ERC sets which is represented by the ERC <W, e, L>. Here the ERC <W, W, L> is redundant since <W, e, L> entails <W, W, L>. Though the algorithm produced redundant information it also produced information that captured all of the shared ranking information from the two ERC sets.

Section 4.6.2 Algorithm focuses on constraints

The above algorithm is structured around constraints, and information across ERC sets is extracted on a constraint-by-constraint basis. The algorithm focuses on constraints and particularly on Ls in constraints because of the necessity of domination implied by an L in a constraint. If an ERC contains an L in a constraint then that constraint must be dominated by some set of constraints. If each set of ERCs from a collection of sets of ERCs each contain an ERC that has an L in a given constraint then the collection of ERCs (even if containing conflicting information) require that that constraint be dominated by some set of constraints. Because an L in a constraint entails necessary domination the algorithm attends to Ls in constraints.

Since the join of two ERCs is entailed by each ERC (as was shown in section 4.4.1.), the join of the ERCs, one ERC from each ERC set, will be jointly entailed by the ERC sets.

Furthermore, this join is *informative*. That is, first, the join of the collection of ERCs across ERC sets for a given constraint, say C_k, in the algorithm above has an L in at least one constraint precisely because it is the join of a set of ERCs that each have an L in the same constraint, namely C_k. The join of two Ls is always an L and hence by joining only those ERCs in the ERC sets that have an L in constraint C_k the result would have an L in constraint C_k and some number of Ws in the other constraints (here it is necessary that each ERC set contain an ERC with an L in constraint C_k, otherwise the resulting join would not be entailed by all of the ERC sets, only those that contained the L in constraint C_k). This resultant ERC would capture the fact that constraint C_k must be dominated.

While this ERC is entailed by each of the ERC sets and it contains non-trivial information it is still possible that it is not maximally informative. An example of this situation is given below using the two ERC sets from the previous example.

19. ERC Set 1

	C1	C2	C3
ERC1	W	L	L
ERC2	e	W	L

20. ERC Set 2

	C1	C2	C3
ERC3	W	e	L

The first ERC set here requires via ERC1 that constraint C1 dominates C2 and C3 and via ERC2 that constraint C2 dominate C3. These two ranking requirements are logically independent and no single ERC can capture their ranking requirements. The second local ERC set only requires that constraint C1 dominate C3.

Now, attending to the first constraint, constraint C1, it is the case that no ERCs from either ERC set contain an L in the column C1. Moving on to constraint C2, only ERC Set 1 contains an ERC with an L in constraint C2, specifically ERC1. Because ERC Set 2 does not have an ERC with an L in constraint C2 it does not require that C2 be dominated by any constraint. Hence the L in ERC1 in constraint C2 is uninformative. Finally, focusing on constraint C3 both ERC sets contain an ERC with an L in constraint C3. In fact, ERC Set 1 contains two ERCs with an L in column C3. These three ERCs that have L in constraint C3 can be joined together to yield shared ranking information across the two ERC sets. This is done in the tableau below.

21. Joining ERCs containing an L in constraint C3

	C1	C2	C3
ERC 1	W	L	L
ERC 2	e	W	L
ERC 3	e	W	L
1 ⊕ 2 ⊕ 3	W	W	L

Joining the three ERCs yields the information that constraint C3 must be dominated by either C1 or C2. While true for these two ERC sets a stronger statement is also true. Both ERC sets require not just that C1 or C2 dominate C3 but that C1 dominate C3. This information is lost by joining all three ERCs together that contain an L in constraint C3.

The information is lost because there are two ERCs in ERC Set 1 that contain an L in constraint C3 ERC1 <W, L, L> and ERC2 <e, W, L>. The join of these two ERCs together produces <W, W, L> and this joined with ERC3 <e, W, L> produces <W, W, L>. The algorithm avoids this information loss by only selecting one ERC that contains an L in the given constraint from each of the ERC sets. In this way the joining of two ERCs from one ERC set does not happen. With these two ERC sets there are two ERCs that contain an L in constraint C3 in ERC Set 1, ERC1 and ERC2. The algorithm joins ERC1 with ERC3 and then joins ERC2 with ERC3 thereby avoiding the information loss. The result of these joinings is given below.

22. Algorithm applied to ERC Sets 1 and 2

	C1	C2	C3
1 ⊕ 3	W	e	L
2 ⊕ 3	W	W	L

The algorithm produces two ERCs, 1 ⊕ 3 and 2 ⊕ 3, and these two ERCs capture the shared ranking information of the two local lexica. In this case the ERC 1 ⊕ 3 entails the ERC 2 ⊕ 3 and hence the ERC 2 ⊕ 3 adds no information beyond that of 1 ⊕ 3. So the algorithm is not guaranteed to produce maximally concise representations of the shared ranking information, only a set of ERCs that captures the shared ranking information (given the correct structure of the original ERC sets).

Section 4.6.3 Proof of ranking information extraction

The ranking extraction algorithm defined above does indeed extract shared ranking information. The following proves this.

Theorem. For a set of ERC sets $\{A_i\}$, the set of ERCs $J(A_i)$ produced from the above algorithm is entailed by each set of ERCs A_i.

Proof: Let $\{A_i\}$ where i is indexed over a finite set I be a set of ERC sets and let $J(A_i)$ be the result of the above algorithm on $\{A_i\}$. Let $j \in J(A_i)$. I want to show that each ERC set A_i entails j. Now $j \in J(A_i)$ and so it is the result of joining together some set of ERCs from the sets A_i. To be precise, there must be some constraint C such that each A_i contains an ERC with an L in C and j is the result of joining all of these ERCs together. So, for each A_i there is some ERC a_i such that $a_i \rightarrow j$ since j is the result of joining exactly these ERCs together and the individual ERCs of the join entail the join. But this means that $A_i \rightarrow j$ for each A_i. Q.E.D.

The above shows that the result of the algorithm is entailed by each of the constituent ERC sets and hence captures some shared ranking information. Though the algorithm captures some ranking information it is not guaranteed to capture all of the shared ranking information. Only when the ERC sets have the property of individual entailment does the algorithm produce all of the shared ranking information. The next section demonstrates the failure to extract all ranking information and the following section gives a procedure on the ERC sets to guarantee maximal extraction of shared ranking information.

Section 4.6.5 Failure to extract relevant ranking information

Under certain circumstances the algorithm can fail to extract all of the shared ranking information from a set of ERC sets. Although in these situations some ranking information is extracted (when there is shared ranking information to be extracted the algorithm is guaranteed to extract *some* information) the algorithm fails to extract *all* of the shared ranking information. Effectively, transitive ranking information captured by ERCs internal to a given ERC set is missed by the algorithm. If this information were in the ERC sets then no information would be lost. Two examples of this information loss are given below.

Section 4.6.5.1 Information loss example 1

Two ERC sets are presented below.

23. ERC Set 1

	C1	C2	C3
ERC 1	W	L	W
ERC 2	W	e	L

24. ERC Set 2

	C1	C2	C3
ERC 3	W	L	e

The algorithm applied to these ERC sets produces only one ERC. Attending to constraint C1 no ERC is produced since neither ERC set has an ERC with an L in C1. Similarly, no ERC is constructed while attending to C3 since only ERC Set 1 has an ERC with an L in C3 and ERC Set 2 does not. Attending to constraint C2, each ERC set has an ERC with an L in constraint C2. The algorithm joins these two ERCs. The result is given in the tableau below.

25. The join of ERC 1 and 3

	C1	C2	C3
ERC 1	W	L	W
ERC 3	W	L	e
1 ⊕ 3	W	L	W

The result of the entire algorithm on these two ERC Sets then is the single ERC, 1 ⊕ 3, <W, L, W>.

A problem arises because both ERC sets entail the ERC <W, L, e>. First, note that ERC Set 2 clearly entails <W, L, e> since this is the sole ERC constituting ERC Set 2. Also, ERC Set 1 entails <W, L, e>. This is not as obvious. The fusion of two ERCs is entailed by the conjunction of the two and hence the fusion of ERC 1 and ERC 2 is entailed by ERC Set 1. The fusion of ERC 1and ERC2 is <W, L, L> and shown below.

26. The fusion of ERC 1 and 2

	C1	C2	C3
ERC 1	W	L	W
ERC 2	W	e	L
1 o 2	W	L	L

Since the ERC <W, L, L> is entailed by ERC Set 1 it is clear that <W, L, e> is also entailed by ERC Set 1. Hence both ERC Set 1 and 2 entail <W, L, e>. But this ERC is not entailed by the result of the algorithm, the ERC <W, L, W>. Hence the algorithm has failed to determine shared ranking information of these two ERC sets.

This failure of information extraction occurred because transitive ranking information was not present in ERC Set 1. ERC 1 requires that either C1 or C3 dominate C2. ERC 2 requires that C1 dominate C3. If C1 dominates C3 and it is the case that C1 or C3 dominate C2 then necessarily C1 dominates C2. This information is not entailed by any single ERC in the ERC set. The potential to extract the transitive ranking information is lost because ERC 1 does not have an L in constraint C3 and hence is not joined with any ERC from ERC Set 2. Hence the information that is only captured by ERC 1 and ERC 2 is lost because no representative ERC for ERC 1 is constructed in the resultant join set.

Section 4.6.5.2 Information loss example 2

The example presented here demonstrates the same type of information loss but from a more informative set of ERCs. The maximally informative basis (MIB) for a set of ERCs is the set of ERCs that entails all ERCs that that set entails; it is the "most concise (smallest) and most informative (containing the strongest elements, entailmentwise)" basis set of ERC for a given set as described by Brasoveanu and Prince 2005. The algorithm applied to sets of MIBs, even though they are maximally informative, can still fail to extract entailed ranking relations, and it fails for the same reason above – transitive ranking relations were lost for lack of an ERC to join to.

The following two ERC sets are MIBs.

27. ERC Set 1

	C1	C2	C3	C4	C5
ERC 1	W	e	L	e	L
ERC 2	e	W	e	L	W

28. ERC Set 2

	C1	C2	C3	C4	C5
ERC 3	W	W	e	L	e

Applying the algorithm to these two ERC sets produces only one ERC, the ERC produced from joining ERC 2 and ERC 3. This resultant ERC is given below.

29. Result of algorithm on ERC Set 1 and 2

	C1	C2	C3	C4	C5
2 ⊕ 3	W	W	e	L	W

While capturing shared ranking information the algorithm has failed to capture the fact that both ERC set 1 and 2 entail the ERC <W, W, e, L, e>. The second ERC set contains exactly this ERC and therefore clearly entails it. The first ERC set also entails this ERC. Fusing the two ERCs in ERC set 1 produces the ERC <W, W, L, L, L> which clearly entails the ERC <W, W, e, L, e>. Information is lost in this example for the same reasons as in the previous example; ranking information that is implied via transitive ranking relations across ERCs in a single ERC set are lost when one of the ERCs is not joined with any ERCs from the other ERC sets. In the next section a means of overcoming this limitation is given by modifying the ERC sets so the algorithm always extracts all shared ranking information.

Section 4.7 Stage one of the algorithm, processing of ERC sets

The previous discussion was entirely focused on stage 2 of the algorithm. Stage 2 extracts shared ranking information from a collection of ERC sets. This extraction can fail to collect all shared information unless the ERC sets are of the form singular entailment, meaning that for any ERC entailed by the ERC set there is a single ERC in the ERC set that entails this ERC. Stage 1 of the algorithm produces a collection of ERC sets that have precisely this property. Therefore, by executing stage 1 the algorithm ensures that stage 2 captures all ranking information by processing the ERC sets before the extraction of shared ranking information begins. By processing the ERC sets in the way described below the second stage of the algorithm is guaranteed to be maximally informative. All shared ranking information will be determined from the ERC sets.

Given a set of ERCs the initial stage of the algorithm produces the fusional closure of the given ERC set. The fusional closure of the ERC set is the set of ERCs that results from fusing all combinations of ERCs together from the original set and then fusing all combinations of ERCs from the original set and the resulting set and continuing this process until no further new ERCs are obtained. This approach is exactly based on the mathematical meaning of the word “closure”. A set is said to be closed under a binary operation if given any two elements of the set the result of the operation applied to the two elements is still in the set. In this case the operation is fusion and the above construction is the “fusional closure” in the mathematical sense of the original set. It is the smallest set that contains the original set and is closed under the operation of fusion. Any two elements fused together results in an element in the set.

Because the fusion of two ERCs is always entailed by the conjunction of those two ERCs the fusional closure of a set is logically equivalent to the set itself (Prince 2002). Furthermore, this process is guaranteed to terminate since the set of ERCs entailed by a set of ERCs is finite and the process ends when no further new ERCs are created. The fusional closure has one more property that makes it supremely useful for the second stage of the ranking extraction algorithm; for any ERC entailed by the original set there is an ERC in the fusional closure that entails it (Prince 2002, p14, Proposition 2.5).

Section 4.7.1 Example of calculating the fusional closure

Here is presented an example of the fusional closure procedure applied to one ERC set. The set is given below and contains two ERCs, one of which requires that C1 or C2 dominate C3 and the second which requires that C1 dominate C2.

30. ERC Set 1

	C1	C2	C3
ERC 2	W	W	L
ERC 1	W	L	e

The fusional closure procedure starts by taking the fusion of all ERCs pair-wise in the ERC set. In this case there are only two ERCs and hence only one ERC is created from fusing all the ERCs together pair-wise. That ERC is <W, L, L>. Now the algorithm takes this ERC and fuses it pair-wise with all of the ERCs in the original set. In this case the result each time is <W, L, L> and so no new ERCs are created. The algorithm terminates having produced the fusional closure set which consists of the three ERCs below.

31. Fusional closure of ERC Set 1

	C1	C2	C3
ERC 2	W	W	L
ERC 1	W	L	e
1 o 2	W	L	L

This set has the desired property that for any ERC implied by the original set there is one ERC that implies it in the fusional closure. In this fusional closure set given any ERC that is entailed by the original set it is entailed by the ERC <W, L, L> since this ERC entails each of the ERCs in the original set. The other two ERCs in the fusional closure do not provide any information that is not already given by <W, L, L>. In most cases the fusional closure set will contain redundant information. But in many cases this process will not terminate after precisely one round of ERC creation as seen in the example below.

In this example there are three ERCs and four constraints. ERC 1 requires that constraint C1 dominate C2; ERC 2 requires that C1 dominate C3; ERC 3 requires that C1 dominate C4.

32. ERC Set 2

	C1	C2	C3	C4
ERC 1	W	L	e	e
ERC 2	W	e	L	e
ERC 3	W	e	e	L

The algorithm fuses each of these ERCs with each other pair-wise to produce the following three ERCs.

33. Pair-wise fusing of ERCs in ERC Set 2

	C1	C2	C3	C4
1 o 2	W	L	L	e
1 o 3	W	L	e	L
2 o 3	W	e	L	L

These ERCs are then fused with each other and with the original ERCs from the set. This last procedure only produces one new ERC, <W, L, L, L>. Any further fusion of any ERCs from these two steps will not produce any new ERCs. The end result of this procedure then is the seven ERCs above, the three from the original set, the three created from fusing these together, and the final ERC created from the previous six.

Section 4.7.2 Fusional closure allows stage 2 to extract all shared ranking information

The procedure defined at the beginning of this chapter for extracting shared ranking information from ERC sets consists of two stages, turning the ERC sets into their fusional closures and then extracting shared ranking information from the fusional closures. This two-stage process is guaranteed to extract all shared ranking information from the original ERC sets. This is proven below. First, because the fusion of any two ERCs is entailed by the conjunction of both ERCs the fusional closure of an ERC set is logically equivalent to the original ERC set. So, to show that the ranking extraction algorithm extracts all shared ranking information from the set of ERCs it suffices to show that for a given set of fusional closures of ERC sets the ranking extraction stage of the algorithm extracts all shared ranking information from a set of fusional closures.

Theorem: For a set {A_i} of fusional closures of ERC sets and any ERC x such that each A_i entails x, then the ranking extraction algorithm produces a set X and this X also entails x.

Proof: Let {A_i} be a set of fusional closures and let X = J(A_i) be the resultant set of ERCs produced by the ranking extraction algorithm. Now let x be an ERC that each A_i entails. I want to show that J(A_i) entails x. Each A_i is a fusional closure and hence for any ERC y that a given A_i entails there is an ERC $w_i \in A_i$ that entails y. So for x, the ERC that each A_i entails, there is an ERC say, $z_i \in A_i$ that entails x. If x is a non-trivial ERC (meaning it has at least one L for one constraint) then each of the z_i must also have an L for the constraint that x has an L in (and possibly other Ls elsewhere), call this constraint C. Now each z_i has an L in C and so the join of all of the z_is is an ERC in J(A_i). Call this ERC j(z_i). But since all of the z_is entail x, j(z_i) must also entail x since it is the lattice-theoretic join of the ERCs under the entailment partial order. Hence any ERC entailed by all of the A_i is also entailed by J(A_i). Q.E.D.

This theorem along with the fact that the fusional closure of a set is logically equivalent to the original set proves that the two-stage algorithm presented in this chapter extracts all shared ranking information from a set of ERC sets.

Section 4.8 Example of both stages of ranking extraction algorithm applied

As shown in a previous section, without the pre-processing of ERC sets to fusional closure sets the ranking extraction algorithm can fail to determine all of the shared ranking information. Here I apply the algorithm to one of the collection of ERC sets on which the algorithm without the pre-processing failed to extract relevant ranking information.

The two ERC sets are given below. The first has two ERCs and the second has one. As demonstrated above, the algorithm as applied to these two ERC sets and not their fusional closures produces one ERC, <W, L, W>. This ERC is entailed by both ERC sets but fails to capture the fact that both ERC sets also entail the stronger ERC <W, L, e>.

34. ERC Set 1

	C1	C2	C3
ERC 1	W	L	W
ERC 2	W	e	L

35. ERC Set 2

	C1	C2	C3
ERC 3	W	L	e

The first stage of the algorithm constructs the fusional closure of both of the ERC sets given. The fusional closure of the first ERC set consists of the two ERCs in the original set plus the ERC <W, L, L>. The fusional closure of the second ERC set is identical to the second ERC set. The fusional closure adds no new ERCs since there is no transitive ranking information to be extracted since the ERC set consists of a single ERC. The fusional closures of the two ERC sets are given below.

36. Fusional closure of ERC Set 1

	C1	C2	C3
ERC 1	W	L	W
ERC 2	W	e	L
1 o 2	W	L	L

37. Fusional closure of ERC Set 2

	C1	C2	C3
ERC 3	W	L	e

The second stage of the algorithm then attends to each of the three constraints in turn. The first constraint produces no ERCs since the two fusional closure sets do not contain an ERC with an L in constraint C1. Attending to constraint C2, the first fusional closure set contains two ERCs that have an L in constraint C2 while the second fusional closure set contains one ERC with an L in constraint C2. Each ERC from the first closure set is joined individually with the ERC from the second fusional closure set. The results of the two joins are given in the tableau below. The final constraint, C3, produces no ERCs because the second fusional closure set contains no ERCs that have an L in C3.

38. Result of second stage on fusional closures

	C1	C2	C3
$1 \oplus 3$	W	L	W
$(1 \text{ o } 2) \oplus 3$	W	L	e

In the previous application of the algorithm only one ERC was produced <W, L, W> and the shared ranking information captured by the ERC <W, L, e> was lost. Here the fusional closure has introduced a new ERC, 1 o 2: <W, L, L>, which directly captures the fact that C1 must dominate C2. This information was only available in the original ERC set via transitive relations across ERCs in the set and was lost when joining with the ERCs from the other ERC sets. The algorithm applied to fusional closures guarantees that this ranking information (and all ranking information) is not lost.

Section 4.9 Ranking extraction algorithm with shared ranking information

As shown above the ranking extraction algorithm is guaranteed to extract all shared ranking information across a set of ERCs. In the following chapter the learning algorithm extracts ranking information in the form of ERCs from phonotactic information. This set of ERCs functions as a global store of ranking information that holds true for all subsequent learning stages. If the shared ranking extraction algorithm is applied to a set of ERCs without taking into account ranking information in the global store that is shared across the sets then it is possible to fail to extract all shared ranking information. In this section I show an example of this and demonstrate a solution that consists of a slight modification to the above algorithm.[8]

Consider two the following two ERC sets.

[8] This issue and its solution was proposed by Bruce Tesar (p.c.).

39. ERC Set 1

	C1	C2	C3
ERC 1	W	L	e

40. ERC Set 2

	C1	C2	C3
ERC 2	W	e	L

Applying the shared ranking extraction algorithm to these two sets will yield no ERCs. Indeed, the fusional closure of ERC Set 1 is simply ERC 1 and the fusional closure of ERC Set 2 is ERC 2. Applying the join algorithm yields no ERCs because there are no constraints that have an L in them across the two fusional closure sets.

Now suppose that it was known (say through phonotactic learning) that there was an ERC that was globally true. That is, there were ranking restrictions that necessarily held of the two ERC sets that weren't represented in the two ERC sets. Suppose in this example that this global store of ranking information was captured by the following ERC.

41. Globally true ranking restriction

	C1	C2	C3
ERC 3	e	W	L

Now ERC 1 and ERC 3 jointly entail the ranking requirement C1 >> C3 and similarly ERC 2 and ERC 3 jointly entail C1 >> C3 (in fact ERC 2 entails C1 >> C3 since that is its entire logical content). Because the shared ranking extraction algorithm only looks at ranking information extractable from individual sets it misses globally entailed transitive ranking requirements.

The solution to this problem is to compute each of the fusional closures with the global ERC store before applying the ranking extraction step to the computed fusional closures. In this example the fusional closure of the first ERC set with the global ranking store is given below.

42. Fusional closure of ERC Set 1 and global store

	C1	C2	C3
ERC 1	W	L	e
ERC 3	e	W	L
1 o 3	W	L	L

In this fusional closure the transitive ranking requirements are extracted and are represented in the ERC 1 o 3.

In the tableau below the fusional closure of ERC Set 2 and the global store is given.

43. Fusional closure of ERC Set 2 and global store

	C1	C2	C3
ERC 2	W	e	L
ERC 3	e	W	L
2 o 3	W	W	L

Now if the shared ranking extraction algorithm is applied to these two fusional closure sets the ranking requirements implied by the global store and the original individual ERC sets is extracted. Specifically, the ERC <W, e, L> is generated along with <e, W, L>, reproducing the information in the global ERC store.

To possibly increase the speed and reduce the creation of redundant information the algorithm could, after creating the fusional closure of each ERC set and the global store, remove from the fusional closure sets the original global store ranking information. This would not decrease the amount of information extracted from the ranking extraction algorithm because, first, the global store ERCs are still stored in the global store and so while not recreated while extracting the shared ranking information they are not lost. Second, all transitive ranking information implied by the global store and each individual ERC set is captured during the creation of the fusional closure set. This transitive ranking information is then used during the ranking extraction algorithm stage and hence guarantees that transitive ranking information will not be lost.

Section 4.10 Moving along

The next chapter shows this algorithm along with the lexical specification algorithm applied to the linguistic system from the previous chapter. The shared ranking information captured from ERC sets generated from applying error-driven learning on the local lexica as presented here allows the learner to set features that would not have been set using only the lexical specification algorithm presented in the second chapter. The extraction of shared ranking information from the consistent local lexica restricts what local lexica are consistent for further contrast pairs.

Chapter 5. Ranking extraction applied

This chapter incorporates the procedure for extracting shared ranking information from consistent local lexica into the CPR algorithm outlined in Chapter 2. Also incorporated into the algorithm is a pre-morphological knowledge stage that derives phonotactic information. These two additions to the algorithm address one of the issues laid out at the end of Chapter 2. For the language presented there, the learning algorithm failed to determine a certain lexical specification that the language required. The overt forms were consistent only with one underlying form for a certain morpheme and by focusing only on what lexical information a contrast pair yielded the learner failed to determine this necessary specifications. By focusing on lexical information *and* ranking information necessitated by the contrast pairs of the language this information can be determined.

The complete algorithm that accomplishes this and the application to the language are given in this chapter. First a discussion of phonotactic learning is given followed by the definition of the entire algorithm. The final sections will apply the algorithm and demonstrate its solution to the previously discussed problem.

Section 5.1. Phonotactic learning

In the algorithm presented in Chapter 2 the learner is assumed to know the morphological decomposition of the overt forms. Phonotactic learning is an attempt to learn ranking information if this assumption is discarded. As before the proposal given here assumes the learner has access to the overt forms of the language and all of the constraints. No knowledge of the morphological decomposition of the overt forms is known. With these assumptions ranking information can be learned using error-driven learning assuming the identity map (Prince & Tesar 2004). Presented here is an algorithm directly following (Prince & Tesar 2004) that extracts ranking information in the form of ERCs. This phonotactic learning algorithm is the first stage of the complete learning algorithm.

Given a set of overt forms with no knowledge of the underlying forms or the morphological decomposition the learner makes the assumption that underlying forms are identical to the overt forms. This provides an input-output mapping which the learner can extract ranking information from. Given these underlying specifications for the overt forms the learner proceeds to apply error-driven learning using MRCD. This application of MRCD produces a set of winner-loser pairs that capture ranking information as ERCs. These ERCs place ranking restrictions on all stages of the learning algorithm. They are used in all further applications of MRCD and restrict which rankings are possible and which lexica are consistent for the target language.

Section 5.2. Phonotactic learning applied

In this section I return to the linguistic system from Chapter 3 and apply phonotactic learning to the language presented there. Recall that the linguistic system has three features, stress, length, and aspiration. Overt forms are bi-morphemic consisting of a mono-syllabic root and mono-syllabic suffix. There are nine constraints in the system, three faithfulness constraints and six markedness constraints. The constraints are repeated from Chapter 3 below. The faithfulness constraints are listed in 1 – 3 and the markedness constraints are listed in 4 – 9.

1.	Id(s)	The stress value must be identical to its input correspondent
2.	Id(l)	The length value must be identical to its input correspondent
3.	Id(a)	The aspiration value must be identical to its input correspondent
4.	ML	Stress must fall on the leftmost syllable
5.	MR	Stress must fall on the rightmost syllable
6.	WSP	If a vowel is long it must be stressed
7.	*V:	Do not have a long vowel
8.	NoAsp	Do not have an aspirated segment
9.	[+s, −l, −a]	Do not have a segment that is stressed, short, and −a

As an example the learner will apply phonotactic learning to a particular language of this linguistic system defined by the constraint hierarchy in 10.

10. WSP, *[+s, –l, –a] >> Id(s) >> ML >> MR >> Id(l) >> NoAsp, *V: >> Id(a)

This defines the mapping of the morphemes given in 11.

11. Mappings of language

/stress, length, asp/	ka /−XX/	sá /+−X/	záa /++X/
pa /−−X/	**p^há**ka	pas**^há**	pa**záa**
baa /−+X/	**báa**ka	bas**^há**	ba**záa**
tá /+−X/	**t^há**ka	**t^há**sa	**t^há**za
dáa /++X/	**dáa**ka	**dáa**sa	**dáa**za

Though this language has four distinct roots and three distinct suffixes and twelve overt forms the learner during the phonotactic stage of learning only attends to the phonotactically distinct overt forms. This language only has four phonotactically distinct overt forms, **r**^{h}**á**sa, **ráa**sa, ras^h**á**, and ra**sáa**. These are the forms that the learner will gain phonotactic ranking information from.

The learner proceeds by positing that the underlying forms of the four overt forms are identical to their overt representation. So the learner has four input-output mappings to apply error-driven learning to. These four forms are given in 12.

12. Phonotactic mappings of the language
    a. /**r**^{h}**á**sa/ → **r**^{h}**á**sa
    b. /**ráa**sa/ → **ráa**sa
    c. /ra**s**^{h}**á**/ → ras^h**á**
    d. /ra**sáa/** → ra**sáa**

MRCD applied to these four input-output pairs produces a set of winner-loser pairs. The ranking restrictions imposed by these winner-loser pairs are given below in maximally concise ERC-form for perspicuity sake. These ERCs are then maintained throughout the remainder of the algorithm and are used in any application of BCD, whether it is checking for consistency or determining a final ranking in the final stage of the algorithm.

13. Ranking restrictions of overt forms **r**^{h}**á**sa, **ráa**sa, ras^h**á**, and ra**sáa**

	WSP	*[+s,–l,–a]	*V:	NoAsp	ML	MR	Id(s)	Id(l)	Id(a)
1					L	L	W	W	
2			L	L				W	W
3					L	L	W		W

The ranking restrictions imposed here require that four markedness constraints be dominated by some collection of faithfulness constraints. The alignment constraint ML and MR must be dominated by either Id(s) or by both Id(l) and Id(a) as shown in ERC 1 and 3. The markedness constraints against the features length and aspiration must be dominated by Id(l) or Id(a). The two markedness constraints WSP and *[+s,–l,–a] do not have any ranking restrictions imposed by phonotactic learning even though these two constraints are not violated in any form in the language and, in fact, are undominated in the target ranking. The necessary domination of these two constraints is not captured by the phonotactic learning approach here precisely because no overt form violates them. Because no overt form violates these constraints no sub-optimal candidate will be produced from the error-driven learning stage that violates these constraints and hence the learner will not learn that these forms are necessarily undominated.

As shown above, phonotactic learning produces a set of ERCs that the learner can use to restrict the ranking during all further stages of the learning algorithm. After phonotactic learning the learner is assumed to gain morphological awareness, knowing what the morphemes of the language are and the morphological decomposition of the overt forms. The ranking restrictions imposed by phonotactic learning hold for the target language even after morphological awareness is gained and are maintained throughout the remainder of the learning algorithm.

Section 5.3. Complete learning algorithm

In this section a new learning algorithm, the full and complete CPR: Contrast Pair and Ranking information algorithm, based directly on the algorithm presented in Chapter 2 is described. The complete CPR algorithm presented here is precisely the algorithm of Chapter 2 with two components added, phonotactic learning and ranking information extraction. There are now four stages to CPR, phonotactic learning, initial lexicon construction, contrast pair analysis (which incorporates the ranking information extraction), and final lexical assignment. The four stages are given below.

14. Overview of the complete CPR learning algorithm
    - Stage 1: Phonotactic learning
        - Overt forms are assumed to have underlying forms identical to overt forms
        - Error-driven learning using MRCD is applied to input-output pairs
        - ERCs from error-driven learning are added to a newly created set of ERCs that contain all of the known ranking information about the language
            - This global ranking information set is used in all further BCD applications throughout the algorithm's remaining stages
    - Stage 2: Initial lexical assignment
        - Underlying values of features are set for those features of morphemes that have the same value in every environment
    - Stage 3: Contrast pair processing
        - Step 1: Error-driven learning is applied to those overt forms whose constituent morphemes are fully specified, producing ERCs that are added to the global ranking information set
        - Step 2: The contrast pair with the least number of unspecified features is selected
        - Step 3: Local lexica for the selected contrast pair are tested for consistency using error-driven learning with MRCD
            - Features with the same value across all consistent local lexica are set in the lexicon
            - Shared ranking information is extracted from the consistent local lexica and added to the global ranking information set
        - Steps 1-3 are repeated until no further changes occur in the lexicon
    - Stage 4: Final lexical assignment and ranking determination
        - Remaining unset features are given a default value

- Error-driven learning with BCD is applied to all overt forms producing a final ranking

The first stage of the algorithm is the phonotactic learning presented at the beginning of this chapter. It assumes only that the learner knows what the overt forms of the language are and what the full constraint set is. Phonotactic learning produces a set of ERCs that represent the necessary ranking conditions imposed by assuming the identity map on the morphologically unanalyzed overt forms. These ranking restrictions, captured as ERCs, are then maintained in a global ranking information set throughout the remainder of the algorithm and are used whenever BCD is used to determine rankings or consistency of lexica. In this way the phonotactic restrictions are maintained throughout the learning process and are captured in the final ranking the learner produces.

The second stage of the algorithm is the initial lexical construction stage as described in the previous version of the algorithm in section 2.1.1. Nothing is changed. The learner is aware of the morphological decomposition of the overt forms of the language and determines which features of the morphemes do not vary across environments. Those features that have the same value in every environment are then set to that value in the underlying lexicon.

The third stage of the algorithm is nearly identical to the previously presented contrast pair processing stage. What is added is that shared ranking information is extracted from the consistent local lexica for a contrast pair. This stage consists of three steps.

First those overt forms that are fully specified have error-driven learning using MRCD applied to them. This error-driven learning uses the ERCs that were generated during the phonotactic learning stage. Error-driven learning on these forms determines if they are optimal. If they are not, meaning the ranking restrictions from the phonotactic stage are insufficient to ensure optimality of the fully-specified forms, then error-driven learning will produce new ERCs representing novel ranking information required by the fully specified overt forms. The ERCs from phonotactic learning and those generated from the fully specified overt forms are now part of the global ranking requirement ERC set. They are used in all further applications of BCD.

The second step of the contrast pair processing stage (stage 3) consists of selecting a contrast pair. The contrast pair with the least number of unset features is chosen to be used in the third step, potentially reducing the number of local lexica tested for consistency.

For the selected contrast pair, the third step constructs all local lexica for the pair and tests each for consistency. Testing for consistency entails applying error-driven learning using MRCD. Here the ranking restrictions imposed by the fully specified overt forms and from phonotactic learning are enforced during the RCD stage. This, for some contrast pairs, reduces the number of local lexica that are consistent allowing the learner to extract more lexical and ranking information from the local lexica than otherwise would determined. For the consistent local lexica the learner does two things: determine lexical information and ranking information necessarily entailed by the consistent local lexica. Features that have the same value across all of the consistent local lexica are set in the lexicon. This is accomplished in exactly the same manner as described in Chapter 2. Determining whether a local lexicon is consistent or not generates a set of ERCs that the learner can extract shared ranking information from. For the consistent local lexica the learner extracts the shared ranking information the ERCs generated from testing for consistency entail using the algorithm presented in Chapter 4. This shared ranking information captured in the form of ERCs is then added to the set of global ranking requirements containing the ERCs generated from phonotactic learning and from the processing of the fully specified overt forms. This combined set of ERCs then is retained for each further application of BCD.

These three steps of the contrast analysis stage, error-driven learning on the fully specified forms, selection of the contrast pair with the least number of unset features, and extraction of lexical and ranking information from the consistent local lexica, are then repeated until no further lexical information can be obtained from any contrast pair.

The final stage of the algorithm produces a complete lexicon and ranking by assigning a default feature value to each of the features that remain unset and then applying BCD to the now fully specified overt forms of the language. The contrast analysis stage is not guaranteed to set all features (and in many cases will not) and hence some mechanism for setting feature values is needed. Often, though not always, the features that remain unset are fully predictable from other features in the overt form and so any value will suffice. A default value is assumed for each feature to accomplish this. Throughout the application of the algorithm ranking restrictions are generated in the form of ERCs. While potentially quite restrictive they alone do not necessarily impose a ranking that will generate the given overt forms. The ERCs from the global ranking requirements set used with BCD will do so. This assumes that no features were incorrectly set during any of the feature setting stages. Given the assumptions about the linguistic system no features can be incorrectly set during the initial lexical assignment and contrast pair processing stage. Features could be set incorrectly during the final lexical assignment stage.

It is important to know what type of information is generated and where that information is generated and how that information is used. In CPR, information about the grammar, that is ranking information, is generated solely as ERCs. These ERCs are generated in three places: during phonotactic learning, during error-driven learning on the fully specified overt forms, and from ranking information shared across consistent local lexica of a contrast pair. The ranking information is then used whenever BCD is used; so it is used, again, in three places, during error-driven learning on the fully specified overt forms, during the checking of consistency of the local lexica, and during the determination of a final ranking.

Lexical information is generated in three places: during the initial lexical assignment of non-alternating features, from the uniform values of consistent local lexica, and from the final lexical assignment of features to their default values. Specifying the features of morphemes restricts the potential lexical space and restricts the potential rankings of the grammar. In this way specification of underlying features impacts the extraction of ranking information by reducing the number of rankings that could produce the given forms by reducing the number of local lexica for a contrast pair. So the specification of the lexicon can produce ranking information during the processing of the fully specified overt forms and through the extraction of ranking information from the consistent local lexica. Specification of features also restricts which local lexica are potentially consistent. This allows lexical information gained from one contrast pair to increase the lexical information gained from another.

In the next section this interrelatedness of the lexical specification and ranking restrictions is demonstrated by the application of the algorithm to the problematic language presented in Chapter 3.

Section 5.4.1. Application of complete CPR algorithm

In this section the complete CPR algorithm described above is applied to the language given in Chapter 3 that presented a problem for the first version of the algorithm (given in Chapter 2). This language is the same one that phonotactic learning was applied to at the beginning of this chapter. The two components added to the original algorithm, phonotactic learning and the extraction of shared ranking information from the ranking restrictions imposed by the consistent local lexica, contribute enough information for the learner to determine the necessary feature specifications of the given language. The original version of the algorithm did not accomplish this. In the application of this augmented algorithm we will see that the newly acquired ranking information will capture this necessary entailment by reducing the number of consistent local lexica. Information from each stage of the algorithm except the final stage is used to determine the correct feature specification; phonotactic learning restricts possible rankings, the initial lexical construction reduces the number of consistent local lexica, ranking information from contrast pairs are used with other contrast pairs and exclude potential local lexica allowing the learner to set necessary feature values.

Section 5.4.2. The language repeated

The language the learner is attempting to learn is the same as given at the beginning of this chapter. The ranking and mappings of the underlying forms are repeated below.

15. WSP, *[+s, –l, –a] >> Id(s) >> ML >> MR >> Id(l) >> {NoAsp, *V:} >> Id(a)

16. Mappings of language

/stress, length, asp/	ka /−XX/	sá /+−X/	záa /++X/
pa /−−X/	**p^há**ka	pa**s^há**	pa**záa**
baa /−+X/	**báa**ka	ba**s^há**	ba**záa**
tá /+−X/	**t^há**ka	**t^há**sa	**t^há**za
dáa /++X/	**dáa**ka	**dáa**sa	**dáa**za

This language has four distinct roots and three distinct suffixes. Recall that there are three features in this linguistic system, stress, length, and aspiration. In this language the feature aspiration is completely predictable from the featural values of stress and length. Non-stressed syllables always surface as short and –a because WSP outranks Id(l) ensuring no long unstressed vowels surface and because NoAsp outranks Id(a). Stressed syllables surface as either long or +a but never both. The constraint *[+s, –l, –a] is undominated and so stressed syllables must be either long or +a. And stressed syllables are never both long and +a for the same kind of reason unstressed syllables are never long or +a. The constraint *[+s, –l, –a] only requires that one of the features long or aspirated surface as + under stress. Which value is chosen is determined by underlying length since Id(l) dominates Id(a) along with *V: and NoAsp. Because of this aspiration is completely predictable from the underlying specification of length and stress.

Length and stress must be specified in all underlying forms except for the suffix ka. Because the language has ML dominating MR and length only is contrastive under stress, length is not contrastive under the non-stressed suffix ka. Any underlying specification of length and a for ka will result in the same alternations under the given ranking.

The feature of most concern for this discussion is the length of baa. In the target language baa is necessarily long. As demonstrated in Chapter 3, if baa is short there is no ranking that is consistent with all of the forms of the language. The learner must determine that baa must be set and must be set to long. But in the application of the un-augmented version of the algorithm the learner failed to do just that. By checking for consistency of local lexica and using only ranking information from fully specified overt forms in every contrast pair containing baa the learner found two consistent local lexica, one with baa long and one with baa short. The learner failed to determine that baa short is not consistent with all of the overt forms. As shown in the next few sections the new algorithm will detect this. Phonotactic learning and ranking information from the consistent local lexica along with the previously gained information from the fully specified overt forms and the contrast pairs will provide enough information for the learner to determine that r_2 must be specified and specified long.

Section 5.4.3. The algorithm applied: Stage 1, Phonotactic learning

The first stage of the algorithm applies the phonotactic learning algorithm described at the beginning of this chapter. Phonotactic learning assumes that the phonotactically distinct overt forms have underlying forms identical to their overt forms. As described in the application of phonotactic learning above this language has four phonotactically distinct overt forms, **r^{h}á**sa, **ráa**sa, ras**^{h}á**, and ra**sáa**. The process of phonotactic learning for this language will not be described in detail here since this is the same language phonotactic learning is applied to in the above discussion of phonotactic learning.

The results of phonotactic learning are ranking restrictions. The ranking restrictions obtained from this stage of the algorithm are given below in 17.

17. Ranking restrictions of overt forms **r^{h}á**sa, **ráa**sa, ras**^{h}á**, and ra**sáa**

	WSP	*[+s,–l,–a]	*V:	NoAsp	ML	MR	Id(s)	Id(l)	Id(a)
1					L	L	W	W	
2			L	L				W	W
3					L	L	W		W

These ERCs capture the total ranking restrictions extracted during the phonotactic learning stage on the target language and are maintained throughout the algorithm in the global ranking requirement set of ERCs.

Section 5.4.4. The algorithm applied: Stage 2, Initial lexical construction

The second stage of the algorithm constructs an initial partial lexicon. Those features that do not alternate on a given morpheme are set in the lexicon to their surface value. The overt forms of the language and their necessary underlying specifications are given below in 18.

18. Mappings of language

/stress, length, asp/	ka /−XX/	sá /+−X/	záa /++X/
pa /−−X/	**p^{h}á**ka	pas^h**á**	pa**záa**
baa /−+X/	**báa**ka	bas^h**á**	ba**záa**
tá /+−X/	**t^{h}á**ka	**t^{h}á**sa	**t^{h}á**za
dáa /++X/	**dáa**ka	**dáa**sa	**dáa**za

Here again there are four roots and three suffixes. An 'X' in the underlying specification in 18 signifies the necessary underlying specification. So, for example, baa must be underlying specified as unstressed and long. The underlying specification for the a value of baa is immaterial; the target ranking of the language will produce the same surface forms for the given suffixes whether baa is underlyingly +a or −a. As discussed above, none of the underlying values of a for any of the morphemes need be set to either value; the target ranking will produce the correct output for either.

The algorithm at this stage attends to those features that do not alternate for a given morpheme and sets those values in the lexicon. So, for example, the length of pa never varies; it is always short. So the algorithm sets the feature length for pa to short. Notice it is not necessary that all features of a given morpheme not alternate for the algorithm to set the feature value. The feature values of stress and aspiration vary for pa but because the length of pa does not it is set in the lexicon as short. This process is carried out for all of the features that do not alternate for all of the morphemes. The results of this process are given below.

19. Lexicon after initial lexical construction stage

	/s l a/
pa	/?–?/
baa	/??–/
tá	/+–+/
dáa	/++–/

	/s l a/
ka	/–––/
sá	/?–?/
záa	/??–/

The '?' means that that feature has not been set yet in the lexicon. So for baa the a value is set to – while the stress and length remain unset. Notice the difference in specified underlying values and necessary underlying values. Each of the morphemes need not have a particular value for aspiration and yet five of the seven morphemes have the aspiration value specified. This comes about because the a value is predictable from the values of stress and length and even though either a value is possible for any of the morphemes the stress and length value cause the a value not to alternate on the surface. In these forms the algorithm sets the non-alternating values of a in the lexicon. It in no way has determined that a is predictable, only that some forms have non-alternating a values. The fact that a is predictable is a property of the complete ranking and is obscured from the learner at this time.

At the end of this stage the learner has set some of the features of the lexicon, though not all, and has determined some ranking restrictions imposed by the phonotactics. The learner needs more information about both the lexicon and the ranking to learn this language.

Section 5.4.5. The algorithm applied: Stage 3, Contrast pair processing

The learner now proceeds to the contrast pair processing stage. This stage is composed of three steps that are repeated until no further changes occur in the lexicon. The first step is error-driven learning applied to the fully specified overt forms. At the end of the second stage there were three morphemes that were fully specified, tá, dáa, and ka, and consequently two overt forms that are fully specified, **t^{h}á**ka and **dáa**ka. The algorithm begins by attempting to parse /táka/ using the hierarchy produced by BCD on the ERCs from phonotactic learning. This hierarchy is given below.

20. Initial hierarchy before parsing /táka/

{WSP, [+s,–l,–a]} >> Id(s) >> {ML, MR} >> Id(l) >> {*V:, NoAsp} >> Id(a)

This hierarchy is one of several that is consistent with the ERCs from phonotactic learning. In the construction of this hierarchy BCD has a choice of which faithfulness constraint to rank highest; any of the three faithfulness constraints will free two markedness constraints for ranking. Id(s) frees ML and MR while Id(l) and Id(a) free *V: and NoAsp. An arbitrary choice must be made. In the application of this algorithm I decided to select a hierarchy that is as close to the target hierarchy as possible when BCD is presented with an arbitrary ranking decision. Here this hierarchy is chosen since the target language ranks Id(s) above Id(l) and Id(a), Id(s) is chosen. This ranking information about the target language is unavailable to the learner and in a complete investigation of every path this algorithm could take each decision point would be investigated. This choice of ranking by the analyst and not the learner (which is only made when there is an arbitrary choice to be made) is done with the intuition that parsing during error-driven learning with a hierarchy that respects more of the target ranking's restrictions will be *less* informative than parsing with a hierarchy that is more dissimilar from the target ranking. This intuition comes from the fact that correct parsing does not produce a winner-loser pair and consequently does not provide the learner with new ranking information. Incorrect parses are informative while correct parses are not.

Using the hierarchy derived from BCD on the ERCs from phonotactic learning on the two forms /táka/ and /dáaka/ is uninformative. That is, the hierarchy in 23 used to parse the underlying form of /táka/ produces the correct output of **t^{h}á**ka. Similarly the hierarchy in 23 used to parse the underlying form of /dáaka/ produces the correct output of **dáa**ka. So, parsing of the fully specified overt forms produces no winner-loser pairs and consequently no ranking information as expected since these same mappings were parsed during the phonotactic learning stage.

The learner now selects the contrast pair with the least number of unset features (though, of course, non-zero number of unset features). At this stage of the learning algorithm there are 19 contrast pairs and of those 19 four have exactly two unset features. All others have more than two unset features. The four contrast pairs are listed below.

21. Four contrast pairs with two unset features

/paka/ ~ /dáaka/	/baaka/ ~ /taka/	/tásá/ ~ /dáasá/	/tázáa/ ~ /dáazáa/
p^{h}**á**ka ~ **dáa**ka	**báa**ka ~ **t**^{h}**á**ka	**t**^{h}**á**sa ~ **dáa**sa	**t**^{h}**á**za ~ **dáa**za

The algorithm randomly selects one of these four minimally unset contrast pairs, say paka ~ dáaka. Of the three morphemes in this contrast pair pa is the only morpheme with unset features, stress and aspiration. These two unset features give rise to four local lexica for the contrast pair, one lexicon for each of the combinations of stress and aspiration for pa. The length value of pa has already been set in the lexicon as minus and all features of ka and dáa are set. The suffix ka is /s: –, l: –, a: –/ and the root dáa is /s: +, l: +, a: –/. These two morphemes underlying values do not vary across the local lexica, only the values for stress and aspiration of pa. The four local lexica are given below.

22. Local lexica for contrast pair paka ~ dáaka

	pa stress	pa aspiration
LL1	−	−
LL2	−	+
LL3	+	−
LL4	+	+

As it turns out, MRCD applied to local lexicon one and four yields rankings that produce the target overt forms and hence the algorithm does not set any underlying features while processing this contrast pair. In fact, none of the initial four contrast pairs yield lexical information at this stage of processing. The next pair that does yield lexical information during the processing of contrast pairs has four unset features and hence $2^4 = 16$ local lexica.

As discussed in chapter 3 the algorithm (without the use of ranking information extracted during the processing of contrast pairs) sets some, but not all, of the underlying features of the morphemes of the language during the contrast pair processing stage. Furthermore, it fails to set crucial features of the language, features that must be set to a specific value for the language to be correctly learned. Repeated below is the lexicon that was constructed by the initial lexical construction stage.

23. Lexicon after processing of contrast pairs without using shared ranking information

	/s l c/
pa	/−−?/
baa	/−?−/
tá	/+−+/
dáa	/++−/

	/s l c/
ka	/−−−/
sá	/+−?/
záa	/?+−/

Section 5.4.6.1. Stage 3, Contrast pair processing, ranking information extraction

The previous section summarized the processing of the contrast pairs that occurs when no ranking information is gathered from the pairs during the processing stage. This section will show that first, ranking information can be extracted from the ERCs generated while checking for consistency of local lexica, and second, that this ranking information when used with a different contrast pair will allow the learner to set a feature that would not have been set by only checking for uniform feature values across consistent lexica. This will demonstrate how lexical information gives rise to ranking information and ranking information, in turn, gives rise to further lexical information.

The learning algorithm, without any ranking information from consistent local lexica, will set nearly all of the features of the language in question as is discussed in Chapter 3. The resultant lexicon after the processing of contrast pairs (but before the setting of default values) when using the original version of the algorithm is given below.

24. Lexicon after processing of contrast pairs without using shared ranking information

	/s l c/
pa	/−−?/
baa	/−?−/
tá	/+−+/
dáa	/++−/

	/s l c/
ka	/−−−/
sá	/+−?/
záa	/?+−/

Recall that two of these unset features must have specific values. The length of baa must be long and zaa must be stressed. The algorithm presented here with the two new sources of information is able to determine that baa must be long precisely because it is able to capture ranking information imposed by one contrast pair and restrict which are the consistent local lexica for another contrast pair. The next section demonstrates the extraction of the shared ranking information from the relevant contrast pair and is followed by a section showing this ranking information being used to set the value of the length of baa.

Section 5.4.6.2. Ranking information extraction

The contrast pair pasá ~ dáasá (which surfaces as pa**s**^{h}**á** ~ **dáa**sa) is the contrast pair that produces ranking information that allows the feature stress to be set in baa. Here I step through the processing of this contrast pair and show the ERCs that are extracted from the consistent local lexica.

This contrast pair has, at this point in the processing, two unset features, the a values for pa and for sá. The known underlying specifications for the three morphemes are in 29.

25. Fixed lexical specifications for contrast pair

	/s l a/
pa	/−−?/
dáa	/++−/

	/s l a/
sá	/+−?/

The two unknown feature values give rise to four local lexica for the contrast pair defined below.

26. Local lexica for contrast pair

	pa a	sá a
LL1	−	−
LL2	−	+
LL3	+	−
LL4	+	+

The first step of the algorithm's processing of this contrast is the determination of consistency of the local lexica. As will be shown, all four local lexica are consistent and consequently no lexical information will be extracted from this contrast pair. But in the determination of consistency ERCs will be generated and ranking information will be gained.

Local lexicon 1 for pas^{h}**á** ~ **dáa**sa

The consistency check for the first local lexicon (that lexicon with the a values of pa and sá set to **–**a) starts with the hierarchy gained from phonotactic learning repeated below.

27. Phonotactic learning hierarchy

{WSP, *[+s, **–**l, **–**a]} >> Id(s) >> {ML, MR} >> Id(c) >> {*V:, NoAsp} >> Id(l)

Error-driven learning proceeds on the first overt form of the contrast pair, pa**s^{h}á**. The hierarchy above does not produce the correct output of pa**s^{h}á** but rather the output pa**sáa**. This creates a winner-loser pair given below.

28. Winner-loser pair generated from first pass of error-driven learning

	/pasá/	WSP	*[+s,-l,-a]	*V:	NoAsp	ML	MR	Id(s)	Id(l)	Id(a)
1	pa**s^{h}á** ~ pa**sáa**			W	L				W	L

BCD applied to this ERC along with the global ranking requirements produce another hierarchy that is used to reparse /pasá/. That hierarchy, given below, successfully parses the input /pasá/ producing the output pa**s**^{h}**á**.

29. Ranking after first winner-loser pair generated

{WSP,*[+s,–l,–a]}>>Id(s) >> {ML, MR} >> Id(l) >> *V:, NoAsp >> Id(a)

The first overt form now has successfully been parsed. The algorithm then attempts to parse the second overt form with the same hierarchy as was used for the successful parsing of the first overt form. This hierarchy incorrectly parses the input /dáasá/ producing the output da**s**^{h}**á**. This incorrect output along with the correct output of **dáa**sa is used to create another winner-loser pair. This winner-loser pair is given below.

30. Winner-loser pair generated from parsing the second overt form

	/dáasá/	WSP	*[+s,-l,-a]	*V:	No Asp	ML	MR	Id(s)	Id(l)	Id(a)
1	**dáa**sa ~ da**s**^{h}**á**			L	W	W	L		W	W

BCD is then applied to this ERC along with the ERC generated from the parsing of the first overt form and the ERCs from phonotactic learning to produce the following hierarchy.

31. Hierarchy generated from BCD

{WSP,*[+s,–l,–a]}>>Id(s) >>ML >>MR >> Id(l) >> *V:, NoAsp >> Id(a)

This hierarchy then successfully parses both the input /dáasá/ and the input /pasá/ producing the correct outputs for the two overt forms of the contrast pair demonstrating that this local lexicon is consistent. The hierarchy above with these feature settings produces the correct outputs. While checking for consistency of this local lexicon two ERCs were created. These are the ranking restrictions imposed by this local lexicon. These ERCs are then kept until the checking for consistency of all of the local lexica has been completed.

Local lexicon 2 for pas$^{\mathbf{h}}$**á** ~ **dáa**sa

The second local lexicon has the underlying value for a for pa set to –a and the underlying value for a for sá set to +a. This local lexicon is also consistent. Error-driven learning applied to the two overt forms of the contrast pair produces two winner-loser pairs. The resulting ERCs and the hierarchy that successfully parses the two overt forms are given below. The step-by-step description of the algorithm processing these two forms is omitted for this local lexicon and the remaining two also.

32. Resulting ERCs from error-driven learning on local lexicon two

	/dáasá/	WSP	*[+s,-l,-a]	*V:	NoAsp	ML	MR	Id(s)	Id(l)	Id(a)
1	**dáa**sa ~ da**s$^{\mathbf{h}}$á**			L	W	W	L		W	L
2	**dáa**sa ~ **dáas**^{h}a				W					L

33. Resulting ranking that successfully parses the two overt forms
{WSP,*[+s,–l,–a]}>>Id(s) >>ML >>MR >> Id(l) >> *V:, NoAsp >> Id(a)

Local lexicon 3 for pas^h**á** ~ **dáa**sa

The third local lexicon is also consistent. The underlying specifications for a for pa and sá are +a and –a respectively. Error-driven learning on the two overt forms produces two winner-loser pairs, one for each of the two overt forms in contra-distinction to the previous local lexicon where one of the two overt forms was the driving force behind the creation of both winner-loser pairs.

34. Resulting ERCs from error-driven learning on local lexicon three

		WSP	*[+s,-l,-a]	*V:	NoAsp	ML	MR	Id(s)	Id(l)	Id(a)
1	pas^h**á** ~ p^{h}as^h**á**				W					L
2	**dáa**sa ~ das^h**á**			L	W	W	L		W	W

35. Resulting ranking that successfully parses the two overt forms
{WSP,*[+s,–l,–a]}>>Id(s) >>ML >>MR >> Id(l) >> *V:, NoAsp >> Id(a)

Local lexicon 4 for pas^h**á** ~ **dáa**sa

Finally, the local lexicon with both values of a set to +a is also consistent. Error-driven learning produces two winner-loser pairs given below.

36. Resulting ERCs from error-driven learning on local lexicon four

		WSP	*[+s,-l,-a]	*V:	NoAsp	ML	MR	Id(s)	Id(l)	Id(a)
1	pas^h**á** ~ p^{h}as^h**á**				W					L
2	**dáa**sa ~ das^h**á**			L	W	W	L		W	L

37. Resulting ranking that successfully parses the two overt forms
{WSP,*[+s,–l,–a]}>>Id(s) >>ML >>MR >> Id(l) >> *V:, NoAsp >> Id(a)

Ranking extraction from the four local lexica

After determining that all four local lexica are consistent the algorithm inspects whether any lexical information can be extracted from the consistent local lexica. In this case the answer is no. Since all four local lexica are consistent no feature value has a uniform value across consistent local lexica and hence no lexical information is gained. Even though no lexical information is available to the learner ranking information is. Checking for consistency of the local lexica produced four sets of winner-loser pairs that the learner now turns to to extract shared ranking information. The four sets of ERCs created during consistency checking are given below. These are the sets that the ranking extraction algorithm is applied to.

38. ERC set 1 from local lexicon 1

Set 1	WSP	*[+s,-l,-a]	*V:	NoAsp	ML	MR	Id(s)	Id(l)	Id(a)
1.1			W	L				W	L
1.2			L	W	W	L		W	W

39. ERC set 2 from local lexicon 2

Set 2	WSP	*[+s,-l,-a]	*V:	NoAsp	ML	MR	Id(s)	Id(l)	Id(a)
2.1			L	W	W	L		W	L
2.2				W					L

40. ERC set 3 from local lexicon 3

Set 3	WSP	*[+s,-l,-a]	*V:	NoAsp	ML	MR	Id(s)	Id(l)	Id(a)
3.1				W					L
3.2			L	W	W	L		W	W

41. ERC set 4 from local lexicon 4

Set 4	WSP	*[+s,-l,-a]	*V:	NoAsp	ML	MR	Id(s)	Id(l)	Id(a)
4.1				W					L
4.2			L	W	W	L		W	L

The first step of the ranking extraction algorithm is to compute the fusional closure of each of the four ERC sets. The result of this step is given below.

42. Fusional closure of Set 1

	WSP	*[+s,–l,–a]	*V:	*+a	ML	MR	Id(s)	Id(l)	Id(a)
1.1			W	L				W	L
1.2			L	W	W	L		W	W
1.1 o 1.2			L	L	W	L		W	L

43. Fusional closure of Set 2

Set 2	WSP	*[+s,–l,–a]	*V:	*+a	ML	MR	Id(s)	Id(l)	Id(a)
2.1			L	W	W	L		W	L
2.2				W					L

44. Fusional closure of Set 3

Set 3	WSP	*[+s,–l,–a]	*V:	*+a	ML	MR	Id(s)	Id(l)	Id(a)
3.1				W					L
3.2			L	W	W	L		W	W
3.1 o 3.2			L	W	W	L		W	L

45. Fusional closure of Set 4

Set 4	WSP	*[+s,–l,–a]	*V:	*+a	ML	MR	Id(s)	Id(l)	Id(a)
4.1				W					L
4.2			L	W	W	L		W	L

Notice that in the fusional closure of two of the four ERC sets, namely Set 2 and Set 4, no new ERCs were introduced in calculating the fusional closure. In these cases the fusion provides no transitive ranking relations implied by the conjunction of the two ERCs. Set 2 and Set 4 were already fusionally closed.

The ranking extraction algorithm now has the four sets from which it will attempt to extract ranking information. It attends to each of the constraints in turn, checking to see if there are ERCs in each of the sets that have an L in that constraint. For these four sets there are only three constraints that have ERCs with an L for that constraint in each of the sets, *V:, MR, and Id(a). The results of joining the selected ERCs for each of the constraints are given below. The constraints WSP and *[+s,–l,–a] are omitted from the tableaux because no ERCs contain any ranking information about these two constraints.[9]

46. Domination information for *V:

	*V:	*+a	ML	MR	Id(s)	Id(l)	Id(a)
1.2 ⊕ 2.1 ⊕ 3.2 ⊕ 4.2	L	W	W	L		W	W
1.2 ⊕ 2.1 ⊕ (3.1 o 3.2) ⊕ 4.2	L	W	W	L		W	W
(1.1 o 1.2) ⊕ 2.1 ⊕ 3.2 ⊕ 4.2	L	W	W	L		W	W
(1.1 o 1.2) ⊕ 2.1 ⊕ (3.1 o 3.2) ⊕ 4.2	L	W	W	L		W	L

47. Domination information for MR

	*V:	*+a	ML	MR	Id(s)	Id(l)	Id(a)
1.2 ⊕ 2.1 ⊕ 3.2 ⊕ 4.2	L	W	W	L		W	W
1.2 ⊕ 2.1 ⊕ (3.1 o 3.2) ⊕ 4.2	L	W	W	L		W	W
(1.1 o 1.2) ⊕ 2.1 ⊕ 3.2 ⊕ 4.2	L	W	W	L		W	W
(1.1 o 1.2) ⊕ 2.1 ⊕ (3.1 o 3.2) ⊕ 4.2	L	W	W	L		W	L

48. Domination information for Id(a)

	*V:	*+a	ML	MR	Id(s)	Id(l)	Id(a)
1.1 ⊕ 2.1 ⊕ 3.1 ⊕ 4.1	W	W	W			W	L
1.1 ⊕ 2.1 ⊕ 3.1 ⊕ 4.2	W	W	W			W	L
1.1 ⊕ 2.1 ⊕ (3.1 o 3.2) ⊕ 4.1	W	W	W			W	L
1.1 ⊕ 2.1 ⊕ (3.1 o 3.2) ⊕ 4.2	W	W	W			W	L
1.1 ⊕ 2.2 ⊕ 3.1 ⊕ 4.1	W	W				W	L
1.1 ⊕ 2.2 ⊕ 3.1 ⊕ 4.2	W	W	W			W	L
1.1 ⊕ 2.2 ⊕ (3.1 o 3.2) ⊕ 4.1	W	W	W			W	L
1.1 ⊕ 2.2 ⊕ (3.1 o 3.2) ⊕ 4.2	W	W	W			W	L
(1.1 o 1.2) ⊕ 2.1 ⊕ 3.1 ⊕ 4.1		W	W			W	L
(1.1 o 1.2) ⊕ 2.1 ⊕ 3.1 ⊕ 4.2		W	W			W	L
(1.1 o 1.2) ⊕ 2.1 ⊕ (3.1 o 3.2) ⊕ 4.1		W	W			W	L
(1.1 o 1.2) ⊕ 2.1 ⊕ (3.1 o 3.2) ⊕ 4.2	L	W	W	L		W	L
(1.1 o 1.2) ⊕ 2.2 ⊕ 3.1 ⊕ 4.1		W	W			W	L
(1.1 o 1.2) ⊕ 2.2 ⊕ 3.1 ⊕ 4.2		W	W			W	L
(1.1 o 1.2) ⊕ 2.2 ⊕ (3.1 o 3.2) ⊕ 4.1		W	W			W	L
(1.1 o 1.2) ⊕ 2.2 ⊕ (3.1 o 3.2) ⊕ 4.2		W	W			W	L

[9] This is because WSP and *[+s, -l, -a] are never violated on the surface and the generated rankings from BCD place them at the top of the hierarchy ensuring that sub-optimal candidates produced will not violated WSP and *[+s, -l, -a].

There is clearly quite a bit of redundant information here. In fact, just focusing on the ERCs generated from the constraint *V: and MR (those ERCs in 50 and 51 above) it is the case that these two sets of ERCs are identical. This is because the ERCs across all four of the ERC sets that have an L in *V: are exactly the same as the ERCs from each of the ERC sets that have an L in MR. Consequently, the algorithm applied to these identical ERCs will produce an identical set of shared ranking information. While each of these ERCS are entailed by each of the ERC sets there is no need to maintain identical ERCs from contrast pair to contrast pair. Simply by discarding identical ERCs and those ERCs that are entailed by a single other ERC we reduce the number of ERCs the learner adds to their list from 24 to 2. These two ERCs are given below.

49. Ranking entailments from the local lexica

	*V:	*+a	ML	MR	Id(s)	Id(l)	Id(a)
(1.1 o 1.2) ⊕ 2.1 ⊕ (3.1 o 3.2) ⊕ 4.2	L	W	W	L		W	L
1.1 ⊕ 2.2 ⊕ 3.1 ⊕ 4.1	W	W				W	L

As shown in the next section the crucial ranking requirement imposed by these two ERCs is the domination of Id(a). The first ERC in 53 requires that Id(a) be dominated by one of NoAsp, ML, or Id(l). This is the key ranking requirement that will provide enough information to set the length feature of sá in the next contrast pair. Id(a) must be dominated in each of these local lexica because of the shared morpheme sá in the contrast pair. In the local lexicon where sá is +a we see that sá in the overt form dáasá, which surfaces as **dáa**sa, must surface as –a requiring that NoAsp dominate Id(a). In the local lexica where sá is set to –a the constraint Id(a) still must be dominated. Because *[+s,–l,–a] is never violated in the language and sá receives stress in the overt form pasá (as pa**s^{h}á**) the morpheme must surface as either long or +a. Since it surfaces as +a and it is underlyingly short and –a one of the two constraints Id(l) or *V: must dominate Id(a) to force the stressed morpheme to surface as +a and not long. So the differing ranking requirements across consistent local lexica all require that Id(a) be dominated by one of the three constraints Id(l), *V:, or NoAsp.

Section 5.4.6.3 Using the ranking information

This ranking information extracted from the previous contrast pair will prove crucial to determining the underlying specification for length for the morpheme baa. Having extracted the ranking information from the previous contrast pair the algorithm processes the contrast pair consisting of the of the overt forms **báa**ka and ba**s^{h}á**. This contrast pair consists of three morphemes, baa, ka, and sá. Of these morphemes baa and sá have unset feature values, baa's length and the a value for sá. The third morpheme ka has all of its features set. The set features for these three morphemes are given below followed by the four local lexica defined by the two unset feature values of baa and sá.

50. Currently set lexical specifications for contrast pair **báa**ka ~ ba**s^{h}á**

	/s l a/
baa	/–?–/

	/s l a/
ka	/–––/
sá	/+–?/

51. Local lexica for contrast pair **báa**ka ~ ba**s^{h}á**

	baa length	sá a
LL1	−	−
LL2	−	+
LL3	+	−
LL4	+	+

To show that the algorithm sets the length of baa it suffices to show that the local lexica that have baa's length short are inconsistent. While it suffices to show that there does not exist a mapping that can produce both of the overt forms for local lexica 1 and 2 it is the case that given the current ranking restrictions there is no mapping that can produce the overt form associated with /baaka/ given that baa's length is short regardless of the output of the other overt form /baasá/. Below it is shown that baa cannot be short; no available ranking will produce the output **báa**ka from /baaka/ when baa is short.

Because it suffices to look at only the form **báa**ka it is acceptable to ignore the feature values for the morpheme sá, specifically its aspiration feature. The overt form containing sá plays no role in setting of the length feature of **báa**ka. Now sá's aspiration feature is the only distinguishing feature between the local lexica 1 and 2, those lexica that have the length of baa short. Therefore below when examining **báa**ka there is no need to distinguish local lexica 1 and 2.

At this stage in the algorithm the learner has determined minimally the ranking restrictions imposed by phonotactic learning and the restrictions from the previous contrast pair. These are repeated below, the first three ERCs being the phonotactic restrictions and the fourth and fifth the ERCs imposed from the previous contrast pair; also shown is the hierarchy they generate.

52. Ranking entailments from the local lexica and phonotactics

	WSP	*[+s,–l,–a]	*V:	*+a	ML	MR	Id(s)	Id(l)	Id(a)
1					L	L	W	W	
2			L	L				W	W
3					L	L	W		W
4			L	W	W	L		W	L
5			W	W				W	L

53. Hierarchy generated from these ERCs
{WSP, [+s,–l,–a]} >> Id(s) >> ML >> MR >> Id(l) >> {*V:, NoAsp} >> Id(a)

This hierarchy is then used to parse the input /baaka/ (which has the underlying value of short for baa). The output for this input with this hierarchy is the incorrect output of **b^{h}á**ka. Because baa is underlyingly short and Id(l) dominates NoAsp and Id(a) the correct output of **báa**ka cannot surface. There is no long feature to be faithful to and hence the root stays short and is unfaithful to its underlying a feature. This creates a winner-loser pair given below.

54. ERC created from incorrect parsing of /baaka/

		WSP	*[+s,-l,-a]	*V:	*+a	ML	MR	Id(s)	Id(l)	Id(a)
6	**báa**ka ~ **b**^{h}**á**ka			L	W				L	W

This ERC is inconsistent with the ranking restrictions imposed by ERCs generated previously in the algorithm. The ranking information that leads to inconsistency comes from two sources, phonotactic learning and the previously encountered contrast pair. From the ERCs listed in 56 the second ERC (which is from phonotactic learning) and the fifth ERC (which is from the previous contrast pair) are precisely the ranking restrictions which lead to inconsistency. The fusion of these two ERCs with ERC from 58 show that they are, in fact, inconsistent, as is demonstrated below.

55. The inconsistency of ERCs 2, 5 and 6

	WSP	*[+s,–l,–a]	*V:	*+a	ML	MR	Id(s)	Id(l)	Id(a)
6			L	W				L	W
2			L	L				W	W
5			W	W				W	L
2 o 5 o 6			L	L				L	L

The fusion of all three of these ERCs is a member of L^{+}. All entries consist either or e or L, with at least one L, and hence the ranking requirements imposed by these ERCs is inconsistent. Because this inconsistency stemmed from only parsing the overt form /baaka/ and not both of the overt forms of the contrast pair the underlying value of the feature for a for the morpheme sá, the other morpheme in this contrast pair, is immaterial. Hence every local lexica that has the underlying value of length for baa set to short will be inconsistent and because of this the algorithm will set the underlying value of length for baa to long. The local lexica that are consistent for this contrast pair must have the underlying value of length for baa be long.

The ranking restrictions that lead to the setting of this feature come from two places, the phonotactic learning phase and the extraction of the shared ranking information from the previous contrast pair. Both sources of ranking information are necessary to set this feature. Indeed, inspecting the fusion of the three ERCs in 59, it is the case that removal of any one of these three ERCs will result in a consistent ranking. Remove any of the three ERCs and the feature will not be set. The fusion of 2 and 5 result in the ranking requirement Id(l) >> {*V:, NoAsp, Id(a)}. ERC 2 is from the phonotactics and ERC 5 is from ranking extraction. Both stages contribute necessary information. The phonotactics contribute the requirement that the markedness constraints *V: and NoAsp be dominated by either of Id(l) or Id(a). This is because both long and +a morphemes are realized on the surface and hence one or both of these faithfulness constraints must dominate the featural markedness constraints. The shared ranking information from the contrast pairs requires that Id(a) be dominated by one of *V:, NoAsp or Id(l) as is discussed above. Identity to an underlying a value cannot cause a morpheme to surface +a, only length and stress can. Specifically, only a short and stressed morpheme (or short and unstressed root) can cause this. This information is gathered from the contrast pair pa**s^{h}á** ~ **dáa**sa. The alternation of sá, surfacing +a and –a forces this domination of Id(a). Finally, the overt form **báa**ka is in conflict with these requirements when the underlying value of length for baa is short. When baa is underlyingly short the fact that it surfaces long cannot be accounted for because of underlying length; the a value of baa must explain why it surfaces as long. The a value of baa is –a, but for this value to be driving the lengthening of baa under stress either Id(a) must dominate {Id(l) and *V:} or NoAsp must dominate {Id(l) and *V:}. But these requirements are in conflict with the fusion of 2 and 5. Effectively, the phonotactics and ranking requirements imposed by the previous contrast pair do not allow the a value of a morpheme to determine the length of a morpheme under stress. Together they require that a be fully predictable from a morpheme's underlying length and stress values. This imposition of the necessary cause of lengthening is what is missing from the previous version of the algorithm that lacked any early phonotactic learning stage. In the previous version of the algorithm from Chapter 2 in these contrast pairs there were two explanations for why a morpheme under stress surfaced long. Either the morpheme was long and was being faithful to underlying length or the morpheme was underlyingly –a and was being faithful to underlying a value. The second option has been removed with the further ranking information obtained by this version of the algorithm.

Section 5.5 Summary of algorithm and sources of information

The algorithm presented here captures both ranking and lexical information about the target language. The initial stage of the learning algorithm determines ranking information from the phonotactics of the language. The second stage of the algorithm determines lexical information from non-alternating features of the overt forms. These two stages are not interrelated. The phonotactic learning provides no useful information for the initial lexical construction stage and the lexical information is not needed for the phonotactic learning stage. It is the third stage of the algorithm where these two types of information, lexical and ranking, directly interact.

For a given contrast pair, the initial lexical information allows the learner to first reduce its search space significantly by obviating the need to consider all possible underlying forms. Initially, only combinations of those features that alternate need be considered. As more features are set in other contrast pairs fewer and fewer local lexica may be considered. The ranking information from the phonotactics and the fully specified overt forms also plays a role here. Ranking information may reduce the number of consistent local lexica increasing the likelihood that the learner will be able to both determine lexical information and extract shared ranking information from the consistent local lexica. The lexical information and ranking information obtained can then impact the next contrast pair again by possibly decreasing the number of consistent local lexica and therefore determining lexical information and ranking information.

Chapter 6. Palauan and implications about the lexicon

The algorithm given in the previous chapter has consequences regarding the underlying representation of lexical items; it can learn languages in linguistic theories that have languages that contain lexical items whose underlying representation does not match any of their surface representations. In this chapter I discuss the implications for the structure of the lexicon and investigate the natural language Palauan which is as an example of a language that has morphemes whose underlying representations do not match any of their surface allomorphs. Palauan provides evidence that learning theories must be able to learn languages that contain lexical items that do not match any of their surface allomorphs. Also presented is a linguistic system that contains a language that has the non-surface matching underlying forms and the learning algorithm presenting in this dissertation is applied to this language successfully learning it.

Section 6.1. Implications for the lexicon

The learning algorithms presented in Chapter 2 and Chapter 5 both rely on consistency checking of local lexica to set underlying values. During the processing of contrast pairs, underlying values are set if and only if they have the same value across consistent local lexica. There are no other requirements imposed on the setting of features during the processing of contrast pairs. This allows for a given morpheme to have its underlying representation not match any of its surface allomorphs. This could come about if, during the processing of contrast pairs, each feature of the underlying form was set by matching the feature of a different allomorph in such a way that the underlying form did not match any of the allomorphs in the language. In this chapter an example of such a situation is given.

Besides showing that formal linguistic systems are capable of having languages that have underlying forms that do not match any surface allomorphs and that these languages are learnable below I present empirical evidence that natural languages also demonstrate this type of lexical behavior. Palauan, as shown below, has precisely these types of underlying forms.

Section 6.2. Palauan

As presented in Flora 1974 and discussed in Schane 1974, Palauan presents evidence that underlying forms may differ from all of their surface allomorphs. This analysis arises because of the nature of stress placement in Palauan and the interaction of stress and vowel quality. In Palauan the position of stress is predictable; if a word contains a suffix the word receives final stress, otherwise the word receives penultimate stress. This is demonstrated in the forms below.

1. Palauan stress placement

mə-téʔəb		kərə-mám
PRESENT MIDDLE-pull out		question-our
‘pull out’		‘our question’

The present middle marker is the prefix ‘mə-’ and with the verb ‘teʔəb’ the form receives penultimate stress. But in the possessive construction ‘kərə-mám’ with the suffix ‘-mam’ the overt form receives final stress. This pattern of final stress for words that contain a suffix and penultimate stress otherwise is robust across the forms of the language.

What causes the necessity of underlying forms that differ from all of their overt forms in Palauan is not stress placement per se but what happens to stressed and unstressed vowels and the changing placement of stress that is dependent on the existence or non-existence of suffixes. Under stress vowels may be one of five different forms: a, i, e, o, u. Non-stressed vowels all reduce to ə. This is seen in the words given in 1 above and those below.

2. Stressed vowels: o, e, i, a, u, and unstressed ə

dəŋób-l	kér	kərí-k	kərə-mám	bədú-k
cover opening-FUTURE(CONS)	question	question-my	question-our	rock-my
‘cover opening’	‘question’	‘my question’	‘our question’	‘my rock’

In 2 is shown the surface manifestation of each of the six vowels with the five a, i, o, e, u only surfacing under stress and schwa only surfacing unstressed. As argued in Schane 1974, faithfulness to underlying vowel quality under stress and reduction to schwa otherwise necessitates an underlying form for a morpheme that differs from each of its surface allomorphs. Focusing on the root for the verb ‘cover opening’ we can see precisely this need.

3. Root ‘cover opening’ with three affixes

Present Middle	Future Participle (conservative)	Future Participle (innovative)	Gloss
mə-dáŋəb	dəŋób-l	dəŋəb-áll	‘cover opening’

In the present middle form of the verb the root receives the present middle prefix ‘mə-’. Because the word has no suffixes it receives penultimate stress and the root itself surfaces as [dáŋəb] with the first vowel surfacing as [a] and the second vowel reducing to schwa. In the future participle (conservative) form the root receives the suffix ‘-l’ and because the word contains a suffix it has final stress. In this case the root surfaces as [dəŋób] and the initial vowel of the root reduces to schwa while the second vowel of the root surfaces faithfully as [o]. Finally, if the suffix itself can bear stress as in the case with the future participle (innovative) suffix ‘-all’, then both of the vowels of the root will reduce to schwa as shown in the word ‘dəŋəb-áll’.

So, for the root ‘cover opening’ there are three allomorphs, [dáŋəb], [dəŋób], and [dəŋəb]. Vowel quality under stress is not predictable based on stress placement as alluded to above and shown below, hence under stress the overt forms must be being faithful to the underlying vowel quality of the root. Therefore the underlying root for ‘cover opening’ is /daŋob/. Note that this form does not match any of the three surface allomorphs of the root and yet for the vowels to be faithful to the underlying specification of vowel quality under stress this must be the underlying form.

In the forms below are shown that each of the five vowels may surface in the positions that may receive stress and hence vowel quality is lexically specified.

4. Stress position does not predict vowel quality

Verbs

Present Middle	Future Participle (conservative)	Future Participle (innovative)	Gloss
mə-dáŋəb	dəŋób-l	dəŋəb-áll	‘cover opening’
mə-téʔəb	təʔíb-l	təʔəb-áll	‘pull out’

Nouns

Unpossessed	‘My –‘	‘Our –‘	Gloss
kér	kərí-k	kərə-mám	‘question’
bád	bədú-k	bədə-mám	‘rock’

The four forms presented above also all have the property that their underlying forms differ from each of their surface allomorphs; the phenomenon is not limited to the forms presented at length above. Following similar reasoning to the analysis above the four roots’ underlying forms are given below.

5. Underlying forms for the four roots above

'cover opening'	daŋob
'pull out'	teʔib
'question'	keri
'rock'	badu

Below will be given an optimality theoretic analysis of the phenomenon of interest. It suffices at this point to conclude that natural languages exhibit underlying forms that differ from each of their surface allomorphs. In the next section I present a linguistic system based on the Palauan language that contains a language that exhibits this property and show that the algorithm presented in Chapter 5 will learn this language and assign underlying forms to roots that do not match any surface allomorphs.

Section 6.3. A linguistic system that contains a pseudo-Palauan language

The linguistic system presented here is in many ways similar to the linguistic systems presented in the previous chapters: insertion and deletion are banned, there is a quite restricted typology of vowel behaviors, and morphological structure is severely restricted.

Morphological structure

Overt forms in this linguistic system are composed of either a free-standing root or a root and a suffix. There are no prefixes in this linguistic system. Roots and suffixes themselves are restricted in their composition. Roots are required to be bi-syllabic always. Suffixes may be mono-syllabic or bi-syllabic. The linguistic system does not allow insertion or deletion, so overt forms of the language have two, three, or four syllables. Non-affixed roots always surface as two syllables because underlyingly roots are always bi-syllabic. A root with a mono-syllabic suffix will surface with three syllables and a root with a bi-syllabic suffix will have four syllables. A word may be composed of precisely one root and one suffix or only one root. Multiple suffixes affixed to a root are not allowed.

Features

Syllables of roots and suffixes are specified for stress and vowel quality only. Stress is a binary feature; a syllable may be underlyingly stressed or not. Because syllables are specified for stress (and not morphemes) a root, which is always bi-syllabic, may have each of its syllables either specified for stress or not independent of the other syllable allowing for a four-way distinction between roots with regards to stress. A root may have either both syllables unstressed or both stressed or the first unstressed and the second stressed or the first stressed and the second unstressed. Stress itself is cumulative; it may appear on only one syllable in the output.

Syllables are also specified for vowel quality. There are three vowels in this system: /i/, /u/, and /ə/. A syllable may be specified for precisely one of these vowels. In effect, this is a three-way feature system for vowel quality. The results presented below regarding underlying morphemes not matching any of their surface allomorphs do not rely on this three-way system. A similar result can be constructed using two binary features for vowel quality that yields four distinct vowels but for expository sake a three-way feature system is used. Two of the vowels, /i/ and /u/ are labeled peripheral vowels, while the schwa is labeled a central vowel. This distinction between types of vowels will be used in the construction of some of the constraints of the system.

Constraints

The linguistic system has eight constraints, six markedness constraints and two faithfulness constraints.[10] The markedness constraints are given below.

6. *V(peripheral)/unstress	Do not have a peripheral vowel be unstressed
7. *V(central)/stress	Do not have a central vowel stressed
8. *u	Do not have the vowel u
9. NonFinal	Do not place stress on the final syllable
10. MainLeft	Place stress on the leftmost syllable
11. MainRight	Place stress on the rightmost syllable

The first two markedness constraints, *V(peripheral)/unstress and *V(central)/stress, state preferences for the absence or presence of stress in conjunction with a particular type of vowel. *V(peripheral)/unstress says do not have a peripheral vowel that is

[10] This system is based on an analysis of Palauan suggested by Alan Prince (p.c.). His proposal included *V(central)/unstress, *V(peripheral)/stress, NonFinal, and Max and Dep. This system has been modified to remove insertion and deletion.

unstressed, i.e. [i] and [u] must be stressed. *V(central)/stress says do not have a central vowel stressed, i.e. do not stress [ə]. *u simply says do not have the vowel [u].

MainRight and MainLeft say place stress on the rightmost and leftmost syllable respectively. Both of these constraints are evaluated gradiently. In all of the previous linguistic systems presented in this dissertation gradient evaluation is identical to non-gradient evaluation because overt forms are restricted to bi-syllabicity. In this system overt forms may have two, three, or four syllables and gradient evaluation is needed. NonFinal penalizes overt forms that have stress on the last syllable.

This linguistic system also has two faithfulness constraints. These are given below.

12. Ident(Stress)	The stress value must be identical to its input correspondent
13. Ident(Vowel quality)	The vowel quality must be identical to its input correspondent

Ident(Stress) says that the stress value on a syllable in the output form must be identical to its value in its underlying form. Ident(Vowel quality) requires that the vowel in the output have the same value as the vowel in its corresponding input syllable.

Section 6.3.1 Pseudo-Palauan

In this section a language is presented from the above linguistic system that exhibits some of the behaviors of Palauan and importantly exhibits the same behavior with respect to some of the morphemes' underlying forms; specifically, one of the morphemes of this language must have an underlying form that is different from any of its surface allomorphs.

The language is defined by the ranking below.

14. Ranking of pseudo-Palauan
{*V(p)/u, *V(c)/s, NonFinal} >> MR >> {Id(v), Id(s), ML} >> *u

In this language NonFinal is undominated. This fact ensures that all forms of this language do not have final stress. NonFinal immediately dominates MainRight which dominates Ident(Vowel quality), Ident(Stress), and MainLeft. Because MainRight dominates all of the other constraints that could have an influence on stress placement and NonFinal dominates MainRight stress in this language is always on the penultimate syllable of the word.

The constraints *V(peripheral)/unstressed and *V(central)/stressed are also undominated. Looking to *V(central)/stressed first, having this constraint undominated means that the vowel [ə] will never be stressed in this language. Having the constraint *V(peripheral)/unstressed undominated means that [i] and [u] will never surface unstressed in any overt form. So in any given overt form the stressed syllable will always be either [i] or [u] and all other syllables will [ə]. This fact together with the fact that stress will always appear on the penultimate syllable ensures that alternations will occur in the language. A root with no suffixes will have initial stress (because it is bi-syllabic) with the first vowel either a [i] or [u] and the second syllable schwa. If the underlying specification of the first syllable is either [i] or [u] the surface form will be faithful to that underlying specification. If the first vowel is underlyingly schwa it still will receive stress and it will surface as [i]. Even though *u is ranked at the bottom of the hierarchy it determines what the vowel will be under stress when the underlying vowel is a schwa. The overt form cannot be faithful to the schwa and hence the default vowel under stress is [i]. Regardless of the second syllable's underlying vowel value it will surface as schwa.

That same root with a bi-syllabic suffix will have both of its vowels surface as schwa. Stress will be placed on the suffix, which contains the penultimate syllable, and the two unstressed syllables of the root will necessarily surfaces as schwa. The underlying specification of the vowels of the roots is immaterial when stress does not fall on the root.

The distinct morphemes of this language and the mappings the ranking produces are given below.

15. Mappings of morphemes in this language

/v1, s1, v2, s2 /[11]	∅	kə /X,X/	gigə /¬u,X,X,X/	susə/u,X,X,X/
pəpə /¬u,X, ¬u,X /	**pí**pə	pə**pí**kə	pəpə**gí**gə	pəpə**sú**sə
bəbu /¬u,X,u,X /	**bí**bə	bə**bú**kə	bəbə**gí**gə	bəbə**sú**sə
dudə /u,X,¬u,X/	**dú**də	də**dí**kə	dədə**gí**gə	dədə**sú**sə
tutu /u,X,u,X/	**tú**tə	tə**tú**kə	tətə**gí**gə	tətə**sú**sə

[11] In /v1, s1, v2, s2/ v1 and s1 represents the vowel quality and stress for the first syllable respectively and v2 and s2 represent the vowel quality and stress for the second syllable respectively.

This language has four distinct roots and three distinct suffixes and 16 distinct overt forms that arise from the combination of the four roots with the three suffixes along with the four roots sans suffixes. The consonants in the morpheme's names represent morphemic identity and do not represent any linguistic property besides morphemic identity. For the bi-syllabic morphemes there are four features that need to be specified: the vowel quality and stress value of the first syllable and the vowel quality and stress value of the second syllable. For the mono-syllabic suffix only the vowel quality and stress value of its single syllable need be specified. The four values of the bi-syllabic morphemes are represented in the order /v1, s1, v2, s2/ where v1 is the vowel of the first syllable, s1 the stress value of the first syllable, v2 the vowel of the second, and s2 the stress of the second. An 'X' means the morpheme can have any value for that feature; the ranking will produce the same overt forms regardless of that feature's underlying specification. For vowels a '¬u' means that the underlying value for that vowel must be either schwa or /i/; it may not be /u/. So, for example, looking at the root 'dudə', it has the underlying specification of /u,X,¬u,X/. This means that the first vowel must be underlyingly a /u/ and that it may be either underlyingly stressed or unstressed. The second vowel of the root 'dudə' must not be /u/; it must be either /i/ or schwa. The second vowel may be either stressed or unstressed as signified by the second 'X'.

Notice in this language that all of the morphemes do not require any particular value of stress for any of their syllables. Any given syllable may be either stressed or unstressed. This is to be expected in a language that has fully predictable stress. This language has penultimate stress and consequently any underlying value of stress for any given syllable is immaterial to stress placement. Though other languages in this linguistic system do allow non-predictable stress and so the learner must determine that this language has predictable stress.

Section 6.3.1.1 Root with underlying form not matching any surface allomorphs

This language has a morpheme that has an underlying form that does not match any of its surface allomorphs. Consider the root morpheme ‘tutu’. In the four morphological environments of this language (the empty environment having no suffix, in the environment ‘-kə’, ‘-gigə’, and ‘-susə’) the root surfaces in three distinct ways, [**tú**tə], [tə**tú**], and [tətə]. The four words that this morpheme appears in are given again below.

16. Overt forms containing the root ‘tutu’

    tútə
    tə**tú**kə
    tətə**gí**gə
    tətə**sú**sə

The underlying form for ‘tutu’ though must be /u, X, u, X/. Both of its vowels must be /u/. Under stress a vowel must surface as either [i] or [u]. Crucially, in this language a stressed vowel will surface as a [u] if and only if it is underlyingly /u/. This is because, first, Ident(vowel quality) outranks *u. So an underlying /u/ that receives stress will maintain its vowel quality and surface as [u]. Secondly, an underlying schwa or /i/ will, under stress, surface as [i]. This obtains because of two facts. *V(central)/stress is undominated and hence a stressed vowel that is underlyingly schwa or /i/ will not surface as schwa it will surface as [i]. It surfaces as [i] and not [u] because of the existence of the constraint *u and the lack of existence of a constraint *i, one that disprefers the vowel [i] on the surface. So, because a vowel that surfaces as [u] must be underlyingly /u/ in this language and because in the overt form **tú**tə the first vowel surfaces as [u], it therefore must be underlyingly /u/. Also, because in the overt form tə**tú**kə the second syllable of the root surfaces as [u] it must also be underlyingly /u/. Hence the root 'tutu' must have an underlying specification of /u, X, u, X/. Regardless of the value of stress for the syllables in 'tutu' the underlying form cannot match any of its surface allomorphs since none of the overt forms have both of the vowels of 'tutu' surface as [u].

In the next section I apply the algorithm to this language and show that this language is learnable using the learning algorithm given in the fifth chapter.

Section 6.3.2 Learning algorithm applied to pseudo-Palauan

The ranking and mappings of pseudo-Palauan are repeated below.

17. Ranking of pseudo-Palauan
{*V(p)/u, *V(c)/s, NonFinal} >> MR >> {Id(v), Id(s), ML} >> *u

18. Mappings of morphemes in this language

/v1, s1, v2, s2 /	∅	kə /X,X/	gigə /¬u,X,X,X/	susə/u,X,X,X/
pəpə /¬u,X, ¬u,X /	**pí**pə	pə**pí**kə	pəpə**gí**gə	pəpə**sú**sə
bəbu /¬u,X,u,X /	**bí**bə	bə**bú**kə	bəbə**gí**gə	bəbə**sú**sə
dudə /u,X,¬u,X/	**dú**də	də**dí**kə	dədə**gí**gə	dədə**sú**sə
tutu /u,X,u,X/	**tú**tə	tə**tú**kə	tətə**gí**gə	tətə**sú**sə

Stage 1: Initial lexical assignment to non-alternating features

The first stage of the learning algorithm assigns underlying values to those features that do not alternate. The result of this stage of learning is given in the table below.

19. Results of feature assignment to non-alternating features

	/vs,vs/
pəpə	/??,??/
bəbu	/??,??/
dudə	/??,??/
tutu	/??,??/

	/vs,vs/
kə	/ə−/
gigə	/i+,ə−/
susə/	/u+,ə−/

After the first learning stage the four roots remain entirely unset for any feature values. Each syllable of each root receives stress and hence its vowel quality changes from either [i] or [u] to schwa. The suffixes on the other hand all are fully specified. The mono-syllabic root never receives stress and hence always surfaces as an unstressed schwa. The final syllables of the two bi-syllabic suffixes also never receive stress and always surface as an unstressed schwa. The initial syllables of the two bi-syllabic suffixes always receive stress (being the penultimate syllable in any form they appear in) and hence are set to underlyingly stressed. Also, their vowel quality does not change and hence is set underlyingly in the initial stage of the learning algorithm.

Stage 2: Processing of contrast pairs

The second stage of the learning algorithm, which processes the contrast pairs, correctly sets the vowel quality of all of the vowels that underlyingly must be /u/. All other feature values remain unset after this stage. The results of this stage are given below.

20. Underlying values after the processing of contrast pairs

	/vs,vs/
pəpə	/??,??/
bəbu	/??,u?/
dudə	/u?,??/
tutu	/u?,u?/

	/vs,vs/
kə	/ə–/
gigə	/i+,ə–/
susə/	/u+,ə–/

As discussed in section 6.3.1.1 overt forms that contain a surface [u] must underlyingly have the vowel quality /u/. No other vowel quality will give rise to a surface [u] for any ranking in this linguistic system. So, for any contrast pair that contains an overt form that contains a [u], the vowel quality feature for that syllable will be set to /u/ because all local lexica having any value other than /u/ will be inconsistent. Hence all vowel quality features that must be /u/ will be set during the contrast pair processing stage.

This last statement that the underlying vowel quality /u/ will be set in the contrast pair processing stage for all overt forms needs one addendum. First, it is true that any contrast pair containing a form containing a [u] will set that vowel quality correctly to /u/, but to set all overt forms containing a [u] each overt form must appear in a contrast pair, else the contrast pair processing stage will simply not consider the underlying specifications of that overt form. Consider the two morphemes ‘dudə’ and ‘tutu’. For the initial vowel quality features to be set to /u/ in the morphemes ‘dudə’ and ‘tutu’ the contrast processing stage must evaluate pairs of overt forms that have the property that each overt form of the pair consists of a single morpheme. This is because contrast in the initial syllable for the forms **dú**də and **tú**tə (the two mono-morphemic overt forms that contain the morphemes ‘dudə’ and ‘tutu’) only exists in mono-morphemic overt forms. All overt forms of this language that contain more than one morpheme do not contrast in the initial syllable. The contrast pair processing stage must contain contrast pairs such as **pí**pə and **tú**tə, overt forms that contrast in the initial syllable.

This requirement that contrast pairs contain mono-morphemic forms does not contradict the previous definition of contrast pairs. Contrast pairs are defined as pairs of overt forms that differ on the surface and share a morphological environment. Appearing in isolation is a morphological environment and hence contrast pairs need not share a morpheme.

Stage 3: Final lexical assignment and ranking determination

The final stage of the learning algorithm assigns feature values to those features that have not been set in any of the previous stages of the learning algorithm and determines a ranking from all of the overt forms of the language.

I assume that the default value for vowel quality is schwa and the default value of stress is unstressed. This leads to the following final lexicon.

21. Final lexicon

	/vs,vs/
pəpə	/ə–,ə–/
bəbu	/ə–,u–/
dudə	/u–,ə–/
tutu	/u–,u–/

	/vs,vs/
kə	/ə–/
gigə	/i+,ə–/
susə/	/u+,ə–/

This is a correct lexicon for this language. Because the underlying value of stress does not affect any outputs of the grammar any underlying value of stress may be assigned to any of the overt forms of the language. The vowel quality feature is a different matter though. Those vowels that surface under stress as [i] must be underlyingly either /i/ or /ə/. If they were underlyingly /u/ then they would surface incorrectly under stress as [u]. The assumption that the default value of vowel quality is schwa is not necessary for a correct lexical hypothesis; the value /i/ would also work. Crucially, a default value of /u/ must not be chosen else the underlying specifications of the roots pəpə, bəbu, and dudə would be set incorrectly, but with the selection of schwa the correct lexicon is obtained.

During learning there is enough information for the learner to determine that the underlying value for, say, pəpə must not be /u/ because in all local lexica this underlying value leads to inconsistency. This information though is not retained (or even noted across local lexica) and hence with a non-judicious choice of default underlying vowel quality an incorrect lexicon could be chosen in the final stage of learning. Here this type of prohibited lexical choice is not attended to in the learning algorithm; required lexical selection is.

Error-driven learning on the overt forms of the language then proceeds and produces the correct ranking given below.

22. Ranking of pseudo-Palauan
{*V(p)/u, *V(c)/s, NonFinal} >> MR >> {Id(v), Id(s), ML} >> *u

So the application of the algorithm to this language has correctly learned the underlying forms and the ranking.

Section 6.4. Conclusions regarding the lexicon

It is not necessary to restrict the potential underlying forms of a morpheme to one of its surface allomorphs. As shown in this chapter it is possible that a linguistic system not just allows but requires that an underlying form be different from all of its surface allomorphs. In the pseudo-Palauan language there is a morpheme ‘tutu’ that has three surface allomorphs, [**tú**tə], [tə**tú**], and [tətə], none of which match the underlying form of the morpheme which must have both the vowel quality features for its two vowels set to /u/. This language (and the linguistic system which generates it) is based on the natural language Palauan. Palauan itself provides conclusive evidence that languages like this exist. Finally, I have shown that a language that has these types of underlying forms is learnable. There is no need to exclude these types of languages from linguistic theorizing based on empirical evidence, theoretical plausibility, or learnability considerations.

Chapter 7. Comparison and conclusions

This final chapter discusses an approach to lexical learning presented in Tesar 2004 and compares it with the approach presented here and discusses some of the implications and future directions the research presented here leads to.

Section 7.1 Contrast Analysis

In Tesar 2004 an algorithm using contrast to determine underlying values of features is presented. Tesar shows that given certain assumptions about a linguistic system that that system will have the Faithful Contrastive Feature property (FCF) and that this property can be used to determine underlying forms. The FCF says that "if two morphemes contrast phonologically, they must differ underlyingly in at least one *contrastive feature*, and that feature's values must be *faithfully* preserved in the outputs of the morphemes in the contrasting environment." [Emphasis in original, pp. 4.] So, given two morphemes that contrast on some number of features in a given environment, one of those features that the two morphemes contrast on must be mapped faithfully in both of the morphemes. Knowing which feature the two morphemes must map faithfully allows the learner to set the underlying value of the feature in both morphemes. This is the key to the learning algorithm presented in Tesar 2004.

Sufficient conditions for having the FCF are precisely those assumptions that are made about the linguistic systems presented in this dissertation bar one. For a linguistic system to have the FCF features of the system must be binary. The linguistic systems presented in the first five chapters have this property; all features are binary. The last chapter presents a linguistic system that has a three-valued system. The vowel quality feature has three values, /i/, /u/, and /ə/.[12] Besides the linguistic system presented in the previous chapter, all of the linguistic systems presented throughout this dissertation share the same assumptions as those presented in Tesar 2004 and hence also have the FCF property.

The Contrast Analysis (CA) procedure (presented in Tesar 2004) is an algorithm that uses the FCF to determine underlying forms. Because the linguistic systems presented here have the FCF property it is possible to compare the learning algorithm of this dissertation with the CA procedure with respect to the setting of underlying forms (the CA procedure does not attempt to determine a ranking for a language).[13] The CA only sets feature values when "only one surface-differing feature could possibly be faithfully mapped" [pp. 15]. So, if there are two features that could be possibly faithfully mapped the algorithm will not set either of those features even if both of those features must necessarily be faithfully mapped. Because the CA only sets a feature when a feature is necessarily faithfully mapped this implies that the algorithm presented in this dissertation will always set every feature that the CA will set. If a feature is necessarily faithfully mapped for a morpheme in an overt form, then an underlying value for that morpheme

[12] Though it has a three-valued feature this linguistic system could be modified so that the vowel quality is in fact a binary feature with, say, the two binary features of high and back.
[13] It is also possible to compare this algorithm with the CA for linguistic systems that do not have the FCF. Simply, the CA is not guaranteed to set features correctly in these systems while the algorithm presented here will (excepting the final default assignment of features).

will be inconsistent. That is, no ranking will map that underlying value to the correct output in that overt form because that feature must be faithfully mapped. What this means for the contrast-pair processing algorithm presented here is that all local lexica that do not have the underlying value for this morpheme that matches the overt form, that is, that do not map the feature faithfully, will be inconsistent. Hence this algorithm will set the feature correctly and therefore every feature that the CA procedure will set the contrast-pair processing algorithm will also set.

The contrast-pair processing algorithm in some languages will also set more features than the CA procedure. The language presented in Tesar 2004 on pp18, which the CA procedure is applied to, has the same overt forms and morphological decomposition as one of the languages from the linguistic system presented in Chapter 2. This language is given by the ranking and mappings of forms given below.

1. Language L1
WSP >> Ident(Stress) >> MainLeft >> MainRight >> Ident(Length) >> *V:

2. Mappings of morphemes in L1

/stress, length/	ka /–X/	sá /+–/	záa /++/
pa /––/	**pá**ka	pa**sá**	pa**záa**
baa /–+/	**báa**ka	ba**sá**	ba**záa**
tá /+–/	**tá**ka	**tá**sa	**tá**za
dáa /++/	**dáa**ka	**dáa**sa	**dáa**za

This is the language that the algorithm is demonstrated on in Chapter 2. As shown there, the contrast-pair processing algorithm successfully determines the underlying forms of the language and determines a ranking that produces the correct overt forms. As shown in Tesar 2004, the CA procedure fails to set one of the underlying features, specifically the stress of the suffix záa.

The CA procedure fails to set the stress of záa because in each contrast pair that contains the morpheme and which requires the faithful mapping of stress for záa there already exists a feature that could account for the contrasting value of stress, namely length. Because záa is underlyingly long (and the CA procedure has set the length value) there will always be a potentially explanatory (from the perspective of the CA procedure) faithfully mapped feature. The contrast-pair processing algorithm uses further information about the linguistic system to determine that in fact the záa must also be underlyingly stressed. The algorithm uses information about the constraints, that is, grammatical information that the CA procedure does not use. Because the contrast-pair processing algorithm determines that the alignment constraint MainLeft outranks Ident(Length) it determines that underlying length is not enough to draw stress onto the záa morpheme in the overt forms pa**záa** and ba**záa** and this relative ranking of MainLeft and Ident(Length) cause each local lexicon with záa underlyingly unstressed to be inconsistent for this contrast pair. So grammatical knowledge (ranking information) implied by other forms of the language allows the contrast-pair processing algorithm to extract more information about the underlying forms than does the CA procedure.

Section 7.2 Sources of information and further direction

Learning a natural language is a search of the linguistic space. For any reasonably complex and realistic linguistic system exhaustive searching, which is of course guaranteed to succeed, is bound to end in search times that are unreasonable. Because of this, in this dissertation I have argued that contrast pairs are the correct focus for learning. They provide a balance of tractable search and information content. The algorithm proposed, CPR, by focusing on consistent local lexica of a contrast pair can gather both ranking information and lexical information allowing the learner to bootstrap its way to knowledge about the target language.

It is the fact that contrast pairs yield both ranking information and lexical information that make them such a promising source of information for the learner. Contrast is crucial to this information yield. Because the pairs of overt forms that constitute a contrast pair necessarily contrast on some feature it is guaranteed that some lexical information accounts for the surface contrast. Underlying lexical differences are the only means of explaining surface contrast. Looking to consistent local lexica and finding similarities between them allows the learner to extract some of the lexical requirements imposed by the contrasting surface features. Contrasting pairs both contain lexical information and are tractable enough to allow the learner to extract that information.

The learner does not necessarily extract all lexical information available from a contrast pair though, only those lexical requirements imposed by the current knowledge of the learner. Further information gained from other contrast pairs may then lead the learner, later in learning, to extract from that very same contrast pair further lexical information. This interaction of information from different contrast pairs can allow the learner to quickly hone in on the correct lexicon without resorting to exhaustive search.

Contrast pairs can also yield information about the ranking. Focusing on the consistent local lexica of a contrast pair provides the learner with potential input-output mappings. Knowing that a set of underlying forms could be the correct underlying forms for a contrast pair puts the learner in a position to determine ranking information about the grammar as a whole. It is the extraction of shared ranking information across the consistent local lexica that gives the learner information about the grammar.

These two types of information, ranking and lexical, are intertwined. Knowledge of the lexicon restricts what the potential rankings are and knowledge of the ranking of a language restricts what the potential lexica are. Incremental information gain about the lexicon extracted while investigating a contrast pair can lead to incremental information gain about the ranking while investigating a different contrast pair, and similarly information about the ranking can lead to information about the lexicon. In this way these two sources of information feed each other allowing the learner to avoid testing each lexicon and each ranking together for consistency. The learner can “bootstrap” itself from no knowledge of the ranking or lexicon to complete knowledge of the ranking and the lexicon.

Section 7.3 Further issues

As discussed, this approach does not solve all problems regarding learning a lexicon and ranking concomitantly. The processing of contrast pairs procedure is not guaranteed to set all necessary features and determine all necessary ranking requirements and may during the final stage of learning incorrectly set a feature. The approach laid out here also does not fully explore the issue of determining subset relationships between languages and may determine a superset language is being learned when a subset language is the correct target language. Knowing what aspects of the learning algorithm contribute to these issues may allow future research to shed light on solutions to them.

As mentioned, specification of default feature values in the final stage of the learning algorithm may lead the learner to incorrectly set those features that have not been set by the contrast-pair processing stage. This potential was demonstrated in the pseudo-Palauan language presented in chapter 6. Even though default values may be incorrect the staged learning presented here allows the learner to localize any errors made during learning to those features set in the final stage. Knowing that any lexical or ranking information gained from the previous stages of learning must be correct allows the learner to restrict the focus of correcting any feature-setting mistakes to the final stage.

Furthermore, if an error is made in the final stage it is discoverable. Error-driven learning on the fully-specified forms will fail if any of the features assigned default values are incorrect. The ranking information gained from the algorithm cannot be the source of the error, only the final lexical assignment. Localization and discoverability of error allow this approach to be maintained even if the final stage of learning is insufficient for the learner to correctly determine the target language.

Another issue addressed only briefly in this dissertation is the problem of determining subset relationships between languages. As shown in chapter 3 there can exist languages that are consistent with the overt forms of the language that have different ranking and different underlying forms. In a situation like that presented the method demonstrated here may not produce the correct language. This is because the determination of lexical information and ranking information from contrast pairs only takes correct steps. Ranking information is only accepted if the learner can be sure that it is true of the target language. Similarly, excepting the final stage of learning, the learner only accepts lexical information if the learner is guaranteed that the information is correct. An approach such as this will not be able to adjudicate between two languages that are both consistent with the overt forms presented to the learner. Further techniques and heuristics are needed. One such technique was outlined in chapter 3. While not fully developed such an approach may lead to insights into the nature of determining subset relationships between languages.

Section 7.4 Further direction

The approach to learning presented in the dissertation focuses on extracting ranking information and lexical information from morphologically segmented overt forms. While showing that efficient extraction of information is possible given the focus on the correct unit, it leads to further questions about the nature of learning. Assumed is that the learner knows the morphological decomposition of the overt forms of the language. If this assumption is dropped, how does the learning of the morphemes of the language interact with learning of the underlying forms and of the grammar? How does knowledge and learning of the grammar affect morpheme acquisition? What role does phonotactic learning play here? There is every reason to believe that the techniques and insights developed here could lead to further understanding of the interrelated problem of learning the underlying forms of a language, the grammar of a language, and the morphological decomposition of overt forms.

The interrelated nature of determining ranking and lexical information seems to pose an insurmountable challenge to the learner. And yet by focusing on contrast pairs and iteratively extracting ranking information and lexical information from these contrast pairs the learner is able to determine, if not all, then nearly all information that is possibly gained from the overt forms of the language. This type of iterative, incremental approach of intertwined learning yields much useful information to the learner.

References

Albright, Adam. 2002. The identification of bases in morphological paradigms. PhD. dissertation, UCLA.

Alderete, John, Brasoveanu, Adrian, Merchant, Nazarré, Prince, Alan, and Tesar, Bruce. 2005. Contrast analysis aids the learning of phonological underlying forms. In *Proceedings of the Twenty-Fourth West Coast Conference on Formal Linguistics*, ed. by John Alderete, Chung-hye Han, and Alexei Kochetov, 34-42. Cascadilla Press.

Angluin, Dana. 1980. Inductive Inference of Formal Languages from Positive Data. *Information and Control* 45:117-135.

Apoussidou, Diana. 2006. On-line learning of underlying forms. Ms. University of Amsterdam.

Apoussidou, Diana. 2007. The Learnability of Metrical Phonology. Doctoral dissertation, University of Amsterdam.

Baker, C. Lee. 1979. Syntactic theory and the projection problem. *LinguisticInquiry* 10:533-581.

Bermudez-Otero, Ricardo. 2003. The acquisition of phonological opacity. In *Proceedings of the Stockholm Workshop on Variation within Optimality Theory*, Eds. J. Spenader, A. Eriksson, & O. Dahl. Stockholm: Department of Linguistics, Stockholm University, 25-36.

Boersma, Paul and Bruce Hayes. 2001. Empirical tests of the Gradual Learning Algorithm, *Linguistic Inquiry* 32:45-86.

Brasoveanu, Adrian. 2004. Stress/Length Interaction: Alternations In A System With WSP. Ms. Rutgers University.

Brasoveanu, Adrian and Alan Prince. 2005. Ranking and Necessity. Part I. Ms., Linguistics Dept., Rutgers University. ROA-794.

Gold, E. M. 1967. Language Identification in the Limit. *Information and Control*, 16:447-474.

Hammond, Michael. 2005. Gradience, phonotactics, and the lexicon in English phonology. To appear in *International Journal of English Studies 4* (2004), pp. 1-24. ROA-736.

Hayes, Bruce. 2004. Phonological acquisition in Optimality Theory: The early stages. In Constraints in Phonological Acquisition, ed. by René Kager, Joe Pater, and Wim Zonneveld, 158-203. Cambridge: Cambridge University Press.

Jarosz, Gaja. 2006. Rich Lexicons and Restrictive Grammars – Maximum Likelihood Learning in Optimality Theory. Ph.D. dissertation. Johns Hopkins University.

McCarthy, John. 2005. Taking a Free-Ride in Morphophonemic Learning. Catalan Journal of Linguistics 4. ROA-946.

McCarthy, John and Alan Prince. 1995. Faithfulness and Reduplicative Identity. In *University of Massachusetts Occasional Papers 18: Papers in Optimality Theory*, ed. By Jill Beckman, Laura Walsh Dickey, and Suzanne Urbancyzk, 249-384. Amherst, MA: GLSA, University of Massachusetts.

McCarthy, John and Prince, Alan. 1993. Generalized alignment. In Yearbook of Morphology, eds. Geert Booij and Jaap Van Marle, 79-154. Dordrecht: Kluwer.

Merchant, Nazarré and Bruce Tesar. 2006a. Learning underlying forms by searching restricted lexical subspaces. To appear in *Proceedings of the 40th Conference of the Chicago Linguistics Society.*

Merchant, Nazarré and Bruce Tesar. 2006b. Using Local Lexica to Learn Ranking Information and Underlying Feature Values. Ms., Linguistics Dept., Rutgers University.

Rosenthall, Sam. 1994. Vowel/glide alternation in a theory of constraint interaction. PhD. dissertation, University of Massachusetts, Amherst.

Pater, Joe. 2007. Morpheme-Specific Phonology: Constraint Indexation and Inconsistency Resolution. To appear in *Phonological Argumentation: Essays on Evidence and Motivation*, ed. Steven Pinker. London: Equinox. ROA-1291.

Prince, Alan. 1990. Quantitative consequences of rhythmic organization. *Chicago Linguistic Society*, 26.2, 355-98.

Prince, Alan. 2000. Comparative Tableaux. Ms., Linguistics Dept., Rutgers University. ROA-376

Prince, Alan. 2002. Entailed Ranking arguments. Ms., Linguistics Dept., Rutgers University. ROA-500.

Prince, Alan and Paul Smolensky. 1993. Optimality Theory: Constraint Interaction in Generative Grammar. Ms., Linguistics Dept., Rutgers University. ROA-537.

Prince, Alan and Bruce Tesar. 1999. Learning Phonotactic Distributions. Ms., Linguistics Dept., Rutgers University. ROA-353.

Prince, Alan, and Tesar, Bruce. 2004. Learning phonotactic distributions. In *Constraints in Phonological Acquisition*, ed. by René Kager, Joe Pater, and Wim Zonneveld, 245-291. Cambridge: Cambridge University Press.

Samek-Lodovici, Vieri and Alan Prince. 1999. Optima. ROA-363.

Tesar, Bruce. 1995. Computational Optimality Theory. Doctoral dissertation, Department of Computer Science, University of Colorado, Boulder. Roa-90.

Tesar, Bruce. 1997. Using the mutual inconsistency of structural descriptions to overcome ambiguity in language learning. In *Proceedings of the North East Linguistic Society 28*, ed. by Pius N. Tamanji and Kiyomi Kusumoto, 469-483. Amherst, MA: GLSA, University of Massachusetts.

Tesar, Bruce. 1997. Multi-Recursive Constraint Demotion. ROA-197.

Tesar, Bruce. 2004. Contrast analysis in phonological learning. Ms., Linguistics Dept., Rutgers University. ROA-695.

Tesar, Bruce, 2006. Learning from Paradigmatic Information. To appear in *Proceedings of the North East Linguistic Society 36*. Ms. Rutgers University. ROA-795.

Tesar, Bruce, Alderete, John, Horwood, Graham, Merchant, Nazarré, Nishitani, Koichi, and Prince, Alan. 2003. Surgery in language learning. In *Proceedings of the Twenty-Second West Coast Conference on Formal Linguistics*, ed. by G. Garding and M. Tsujimura, 477-490. Somerville, MA: Cascadilla Press. ROA-619.

Wexler, K. and Cullicover, P. 1980. *Formal Principles of Language Acquisition*, MIT Press, Cambridge MA.

Curriculum Vita

Nazarré Nathaniel Merchant

Education

May 2008	Ph.D. Linguistics, Rutgers University
June 1999	M.S. Computer Science, University of Oregon
June 1997	M.S. Mathematics, University of Oregon
May 1995	B.S. Mathematics and Computer Science, High Honors, Eckerd College

Employment

2007 – Current	Actuarial Associate, Travelers Insurance Co.
2005, Summer	Actuarial Intern, PricewaterhouseCoopers
2004 – 2005	Rutgers University Teaching Fellow
2002 – 2004	Rutgers University Graduate Research Assistant
2000 – 2001	Senior Java Developer, ECWise, Inc.
1999 – 2000	Software Developer, HK Systems.
1995 – 1999	University of Oregon Graduate Teaching Fellow

Publications

Merchant, Nazarré and Bruce Tesar. 2006a. Learning underlying forms by searching restricted lexical subspaces. To appear in *Proceedings of the 40th Conference of the Chicago Linguistics Society.*

Merchant, Nazarré and Bruce Tesar. 2006b. Using Local Lexica to Learn Ranking Information and Underlying Feature Values. Ms., Linguistics Dept., Rutgers University.

Merchant, Nazarré, Tesar, Bruce [first author], Alderete, John, Horwood, Graham, Nishitani, Koichi, and Prince, Alan. 2003. Surgery in language learning. In *Proceedings of the Twenty-Second West Coast Conference on Formal Linguistics*, ed. by G. Garding and M. Tsujimura, 477-490. Somerville, MA: Cascadilla Press. ROA-619.