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

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 $\approx 1.3 \times 10^{14}$ 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	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

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^{h}á	/+–+/
dáa	/++–/

	/s l a/
ka	/–––/
sá	/?–?/
záa	/??–/

Three of the seven morphemes are now completely specified. The roots t^{h}á 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^{h}á**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	/–?–/
t^{h}á	/+–+/
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, t^{h}á**za**>**
baa /**–**, ?, –/	Stress is set to – in <ba**záa, t^{h}á**za**>**
sá /**+**, –, ?/	Stress is set to + in <**báa**ka**,** bas^{h}**á>**
záa /?, **+**, –/	Length is set to + in < **p^{h}á**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^{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^{h}á**ka

21. Incorrect mapping of pa**záa** with záa set to /–+–/
    /pazáa/ → **p^{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	pas^{h}**á**	pa**záa**	par^{h}**á**	pa**láa**	pax^{h}**á**
baa /–+X/	**báa**ka	bas^{h}**á**	ba**záa**	bar^{h}**á**	ba**láa**	**báa**xa
tá /+–+/	**t^{h}á**ka	**t^{h}á**sa	**t^{h}á**za	tar^{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**	fas^{h}**á**	fa**záa**	far^{h}**á**	fa**láa**	fa**x^{h}á**
cá /+––/	**c^{h}á**ka	cas^{h}**á**	**c^{h}á**za	car^{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^{h}á**ra, ra**r^{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	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

28. Mappings of morphemes in L2

/s, l, a/	ka /–––/	sá /+––/	zaa /–+X/		rá /+–+/	láa /++X/	xa /––+/
pa /––+/	**p^{h}á**ka	pas^{h}**á**	pa**záa**		par^{h}**á**	pa**láa**	pa**x^{h}á**
baa /–+X/	**báa**ka	bas^{h}**á**	ba**záa**		bar^{h}**á**	ba**láa**	**báa**xa
tá /+–+/	**t^{h}á**ka	**t^{h}á**sa	**t^{h}á**za		tar^{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**	fas^{h}**á**	fa**záa**		far^{h}**á**	fa**láa**	fax^{h}**á**
cá /+––/	**c^{h}á**ka	cas^{h}**á**	**c^{h}á**za		car^{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.