Learning ranking information from unspecified overt forms using the join

Nazarré Merchant[1]
Eckerd College, St. Petersburg, FL

1 Introduction

A phonological learner must construct a lexicon that matches observed surface forms and must produce a ranking that maps those constructed lexical items to those observed surface forms. These tasks must be performed simultaneously. To address the simultaneous nature of the learning task, previous researchers have used the tact of determining some ranking information from fully specified forms, and then using that ranking information to narrow subsequent lexical searches, allowing the learner to fully determine underlying forms that previously had multiple possible lexica (Merchant and Tesar 2006, Jarosz 2006, Apoussidou 2007, amongst others). This approach is known to fail given a sufficiently rich linguistic environment, stemming partially from the fact that there are often many possible lexica for a given overt form, never allowing a learner to fully specify some underlying forms, and consequently preventing it from determining pertinent ranking information. The learner needs some way to determine useful ranking information from underlying forms that are not fully specified.

In this paper I propose a learning algorithm in which a learner determines necessary ranking information from unspecified or partially specified words. It does so by using the *join operator* (Merchant 2008), an operator on pairs of ERCs (Prince 2002) that yields an ERC that is entailed by each of the original ERCs. This produced ERC has the property that it entails all ERCs that are jointly entailed by the original two. This property can be exploited to determine precisely those ranking requirements stemming from competing lexical hypotheses.

The algorithm works by first positing all possible lexica for an overt form, testing each for consistency, and for each consistent lexicon producing a set of ERCs using MRCD (similar to Merchant and Tesar 2006). These sets of ERCs are then used to produce a set of *fusional closures*, one for each consistent lexicon. The fusional closure of a set has the property that every ERC entailed by the original set is entailed by exactly one ERC in its fusional closure. These fusional closures are then processed using the join operator. This processing is guaranteed to produce any and all shared ranking information from the original sets.

There are two main ideas presented here. First, an overt form that is highly ambiguous regarding its underlying representation, with each possible underlying

[1] Much thanks go to Alan Prince, Jason Riggle, and Bruce Tesar for fruitful discussions and the to the audience at CLS 47 for thoughtful questions. All errors are retained, as always, by the author.

form giving rise to conflicting ranking requirements, can still yield useful ranking information. For example, in a system with three constraints, a learner may hypothesize two lexica leading to two rankings, say C1 >> C2 >> C3 and C1 >> C3 >> C2. Regardless of which one is correct, in the target language C1 must dominate both C2 and C3. This restriction on possible rankings can be used to restrict subsequent lexical searches leading to a virtuous cycle of lexical determination and ranking extraction.

Second, the logic of the linguistic system plays a crucial role in the learner extracting useful information. Extraction of ranking information shared amongst multiple lexical hypotheses is possible because entailment relationships amongst ERCs give rise to the join operator, a computational tool particularly suited for determining shared information.

This algorithm when applied to contrast pairs (Tesar 2004) and augmented with a lexical specification algorithm can be shown to succeed on linguistic systems on which an algorithm that requires full specification of underlying forms to determine ranking information fails (e.g. Merchant and Tesar 2006a). This provides further evidence that the learner can and must be able to gather information about the target grammar from forms that are not fully specified, and that this algorithm, using the join operator, provides such information.

The paper is organized as follows; section 2 details the learning assumptions, 2.1 discusses the overall structure of the learning algorithm. Section 3 gives an example linguistic system to work in, section 3.1 provides a language to learn, and steps through the lexicon testing for a contrast pair, 3.2 demonstrates that conflicting hypotheses can have useful ranking information. Section 4 defines the join of two ERCs, section 4.1 proves that the join is the lattice theoretic join, section 4.2 shows that inconsistent ERCs can have useful ranking information, section 4.3 details why the pair-wise join is insufficient for learning purposes. Section 5 defines the algorithm for joining sets of ERCs, section 5.1 proves that the algorithm just defined produces all shared ranking information. Section 6 discusses the join algorithm used in a learning algorithm that includes lexical learning and section 7 is the conclusion.

2 Learning assumptions

Throughout I assume that the learner is attempting to learn an optimality theoretic system, and prior to learning the learner has access to the full set of constraints, though no ranking information is known. The learner is also assumed to be morphologically aware, having access to all overt forms (words) of the language. Here morphological awareness entails that morphological segmentation in all overt forms is given to the learner, though the learner does not know what the underlying forms of the morphemes are. The learner also knows the identity of morphemes and their positions in overt forms, so that for a given morpheme the learner knows all of its allomorphs and all of the overt forms in which those allomorphs appear.

The learning task presented to the learner then is two-fold: the learner must produce a total order on the constraints of the system along with assigning underlying forms to the given morphemes. This two-fold task is constrained by the overt forms of the language: the learner must assign underlying forms to the morphemes and produce a ranking that maps those assigned underlying forms to the given overt forms.

2.1 Learning ranking information from forms not fully specified

The majority of the following discussion focuses on producing ranking information from overt forms that are not fully specified. The learner, when approached with the learning task of producing both a ranking and a lexicon needs a way to gain information about one or the other. As stated above, previous work assumes that a learner must have fully specified overt forms to determine ranking information, but as will be shown, a learner can produce ranking information from overt forms that are not fully specified.

The algorithm for producing ranking information from forms not fully specified presented here begins with a learning selecting a *contrast pair* (Alderete et al. 2005, Tesar 2004), a pair of overt forms that differ in one morpheme, for consideration. The algorithm then produces a *local lexicon* (Merchant & Tesar 2006b) for that selected contrast pair, a local lexicon being the set of all possible lexica for the morphemes in the contrast pair. Each lexicon from the local lexicon is then tested for consistency using inconsistency detection (Tesar 1997). Inconsistency detection here uses MRCD (Tesar 1997) to produce a set of ERCs (Prince 2002), one set for each consistent lexicon. These sets of ERCs are then processed using the *join algorithm* (Merchant 2008), an algorithm that produces a set of ERCs that captures exactly the shared ranking information amongst the sets of ERCs produced from MRCD.

Effectively, what this algorithm does is consider all mappings from inputs to overt forms, determining which are consistent, that is, which lexica the learner need be concerned with. For each of these consistent lexica there is an associated ERC set, representing ranking requirements that need to hold if that particular lexicon is the correct lexicon. Of course, the learner does not know which one of the lexica is correct, and so does not know which of the ranking requirements are true. The join algorithm extracts from these sets of lexica the shared ranking requirements. The shared ranking requirements are true of the target language, allowing the learner to determine ranking information from overt forms that are not fully specified.

3 A linguistic system to work in

In understanding how the algorithm works it is useful to see it in an actual linguistic system. The system under consideration consists of overt forms consisting of monosyllable roots and suffixes. There is one binary feature in the system, accent, and three constraints. The constraints of the system are listed below and are main left (ML), main right (MR), and faithfulness to accent (F(A)).

1. Constraints of the system
    a. ML main stress must fall on the leftmost syllable
    b. MR main stress must fall on the rightmost syllable
    c. F(A) Faithfulness to underlying accent

As overt forms are formed from monosyllable roots and suffixes, the system consists of four unique morphemes: *rá, ra, sá, sa*, where initial *r-* indicates a root and an initial *s-* indicates a suffix. These four morphemes combine to form, for the linguistic system, two possible overt forms *rása* and *rasá*. Of course, the ranking of the constraints will determine whether both overt forms possible are realized.

3.1 Determining possible lexica for overt forms

For a given language the learner is attempting to learn, the algorithm first selects a contrast pair and produces all possible underlying forms for the morphemes in the contrast pair. To see this in action, consider the language defined by the ranking F(A) >> ML >> MR and the given contrast pair $rá_1sa_1$ and $ra_1sá_2$.

2. Language and contrast pair under consideration
    a. F(A) >> ML >> MR
    b. $rá_1sa_1$ and $ra_1sá_2$

Now, the learner knows neither the ranking nor the underlying forms of the three morphemes in the contrast. Because there are three morphemes in this contrast pair, and an underlying form in this language consists of either a plus accent featural value or a minus accent featural value, there are eight (=2^3) possible lexica. The algorithm constructs all eight and tests each for consistency.

For this contrast pair, first consider the underlying lexicon that has all features set to minus accent, given in 3 below. The algorithm uses this lexicon, using MRCD, to test for consistency. Looking to the comparative tableau in 4 below it is readily apparent that this lexicon is not a viable lexical hypothesis for these two contrast pairs.

3

Lexical hypothesis: accent values		
ra_1	sa_1	sa_2
–	–	–

4

	F(A)	ML	MR
$rá_1sa_1$~*rasá*	e	W	L
$ra_1sá_2$~*rása*	e	L	W

The first form, *rá$_1$sa$_1$~rasá*, requires ML over MR, while the second, requires MR over ML. Clearly both rankings cannot obtain and the lexical hypothesis under consideration is rejected by the learner as a viable lexicon. As analysts one can quickly see why any hypothesis in which the underlying lexical values of the morphemes are the same, as here, is doomed to failure: the contrast pair *contrasts* on something and that contrast must be because some faithfulness constraint requires that some differing underlying value manifest itself faithfully, an impossibility if all the underlying featural values are the same.

This tactic of testing for consistency is now repeated for a different lexical hypothesis, one in which the underlying forms contrast on accent giving hope to producing a viable language. In 5 below, suffix 2, *sa$_2$*, is underlyingly accented, while root 1 and suffix 1 remain unaccented. The output of MRCD is given in 6, producing a viable language in which F(A) outranks ML which outranks MR.

5

Lexical hypothesis: accent values		
ra$_1$	*sa*$_1$	*sa*$_2$
–	–	+

6

	F(A)	ML	MR
rá$_1$sa$_1$~rasá	e	W	L
ra$_1$sá$_2$~rása	W	L	W

This process of producing a lexicon and determining if it is consistent can be repeated for all eight lexica of the three morphemes. The outcome of testing for consistency for each of the eight lexica is given in the table below.

7

	Accent Values			
	ra$_1$	*sa*$_1$	*sa*$_2$	Consistent?
	–	–	–	no
A	–	–	+	yes
	–	+	–	no
	–	+	+	no
	+	–	–	no
B	+	–	+	yes
	+	+	–	no
	+	+	+	no

Only two of the eight possible lexica are consistent, language A and language B, the two non-greyed-out languages above. Having determined this, the learning can conclude that any ranking information that is true of *both* language A and

language B is true of the target language, regardless of which set of featural values is correct.

3.2 Existence of ranking information across sets

Having determined that there are two languages that are correct it is instructive to look to the two languages to see if there are indeed shared ranking information between them. As it turns out, there is. Repeated below is the tableau produced above for the lexical hypothesis A and the ERCs that produced the it along with the tableau produced for lexical hypothesis B.

8

Hyp(A)	F(A)	ML	MR
ERC 1	e	W	L
ERC 2	W	L	W

9

Hyp(B)	F(A)	ML	MR
ERC 3	W	W	L
ERC 4	e	L	W

As can be readily gleaned, hypothesis A requires the ranking F(A) >> ML >> MR while hypothesis B requires F(A) >> ML >> MR. Notice that in both rankings F(A) ranks above both ML and MR. Regardless of which lexical hypothesis is ultimately correct (determined by other information in a more complicated example), the learner can safely conclude that in the target language F(A) >> ML & MR. Of course, the learner needs a mechanism for determining this shared ranking information.

The output of the MRCD algorithm is a set of ERCs, available to the learner. It is over these sets of ERCs that the learner can compute the shared ranking information across lexical hypotheses. To do so, one needs additional computational machinery, defined in the next section.

4 The join operator

The following section will produce an algorithm that extracts shared ranking information from sets of ERCs. To do so, first I define an ordering on the set {W, e, L} by the order W > e > L. Then, define the *join* operation, denoted by ⊕, on this set to be the lattice theoretic join; that is, ⊕ is the function from {W, e, L}x{W, e, L} into {W, e, L} that produces the maximal element of the pair according to the just defined order on {W, e, L}.

10 (a) $x \oplus W = W \oplus x = W$

(b) $L \oplus e = e \oplus L = e$

(c) $x \oplus x = x$

In 10 above (a) demonstrates that W join anything is W, that is, W is 'dominant'. L join e is e, as shown in (b), and (c) shows that the operation is idempotent. From 10, on can conclude that L is the identity and, of course, the operation is commutative.

This defines a join on too small of a set for the purposes of learning; needed is an operator on ERCs, not just elements of {W, e, L}. Using the above definition of the join on the set {W, e, L} an operation on pairs of ERCs can be defined, which will also be called the *join*.

The *join*, denoted ⊕, on a pair of ERCs is produced via the component-wise join on the entries in the pair of ERCs. So, for a pair of ERCs, x and y, the column k of $x \oplus y$ is the entry in the k^{th} column of x joined via the join operator on {W, e, L} with the k^{th} column of y. This is demonstrated with two ERCs in the tableau below.

11

	C1	C2	C3
x	W	L	e
y	L	L	W
$x \oplus y$	W	L	W

As can be seen, the join of x and y, $x \oplus y$, has its entries determined by the components of the individual ERCs x and y so that the k^{th} column of $x \oplus y$, $(x \oplus y)[k] = x[k] \oplus y[k]$. Here it's important to note that the operator ⊕ is overloaded in the sense that it either means the join of two elements from the set {W, e, L} or the join of two ERCs. The context in which it appears will unambiguously determine which version of the join operator is being used.

4.1 Properties of the join

The join operator on pairs of ERCs has a very nice property that will be useful for learning purposes. Specifically, the join of two ERCs, which is an ERC, contains precisely the shared ranking information of the two original ERCs. That is, for ERCs x and y, $x \oplus y$ is that ERC which entails all ERCs that both x and y entail. Furthermore, $x \oplus y$ is entailed by both x and y. These two facts, proved below, combine to yield the conclusion that no other information about shared ranking between the two original ERCs can be gained either with ERCs or other representations. That is, if one knows that either ERC x is true or ERC y is true, one can conclude that ERC $x \oplus y$ is true. Furthermore, *no other information is available from x and y*.

Before laying out the proof, it's useful to note that this join, ⊕, is equivalent to the lattice theoretic join created by ERC entailment. ERC entailment imposes an order on ERCs via $x > y$ if y entails x. This order immediately produces a lattice with its concomitant operation, the join. This join is equivalent to the join just defined. That is, the join on the lattice created by the induced order from ERC

entailment is the same as the join, ⊕, on two ERCs. This fact is proved in the claim below.

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

Proof: Let j be the lattice theoretic join of *x* and *y*. Then j has the property that $x \rightarrow$ j and $y \rightarrow$ j and if there is an ERC *w* such that $x \rightarrow w$ and $y \rightarrow w$, then j $\rightarrow w$.

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

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

4.2 Information from inconsistency

As stated above, the join produces all and only the shared ranking information from two ERCs, even if those two ERCs are inconsistent. That is, if two ERCs impose contradictory ranking requirements it is still possible that there is useful information to be gained by taking the join of the two ERCs. To see this, suppose that either ERC *x* or ERC *y* is true, and the learner does not know which holds (but does know that *one* of the two is true, where *x* and *y* are as defined below.

	C1	C2	C3
x	W	L	L
y	L	W	L
*x*o*y*	L	L	L
x⊕*y*	W	W	L

Quick inspection, facilitated by the fusion operator, shows that *x* and *y* are inconsistent. ERC *x* requires that C1 dominate both C2 and C3, while ERC *y* requires that C2 dominate C1 and C3. Clearly they both can't hold as shown in *x*o*y*. Even though they are contradictory, the learner can determine, using the join, that regardless of which is true, *x*⊕*y* is true, and consequently that C3 must be dominated.

4.3 The pair-wise join is insufficient for learning purposes

Even though the pair-wise join produces a useful ERC that contains the shared ranking information of the two joinands it is insufficient in a more general learning environment because a lexical hypothesis will often produce more than one ERC. The learner, given a number of lexica, will produce a number of *sets* of ERCs, one of which is produced from the correct lexicon. Because of this, the learner needs a means of producing the shared ranking information across sets of ERCs, not only from pairs.

A naïve approach to producing shared ranking information from sets of ERCs is simply to take the join of all the ERCs in all of the sets. This approach is doomed to failure, as it will produce a single ERC, which, in most circumstances, will be underpowered representationally. As will be shown in the next section, a particular set of ERCs, produced using the join, will do the job where a single ERC will fail. Before showing this though, it is instructive to see an example in which joining all the ERCs fails to yield the desired information.

Returning to the two consistent lexical hypotheses produced in the first example and their corresponding ERCs, reproduced below, the join of all four ERCs is given in 15. As shown, the join is uninformative, consisting of all Ws, Ws that were contributed by each of the two ERCs in each of the hypotheses. Joining all four ERCs erases the necessary domination information since each constraint has a W in it in *some* ERC. What is needed is some way to ensure that the Ws, being dominant under the join, do not obscure the fact that, in this case, there are Ls in ML and in MR in both of the ERC sets, Ls that need to be maintained.

13

Hyp(A)	F(A)	ML	MR
ERC 1	e	W	L
ERC 2	W	L	W

14

Hyp(B)	F(A)	ML	MR
ERC 3	W	W	L
ERC 4	e	L	W

15

	F(A)	ML	MR
1⊕2⊕3⊕4	W	W	W
Goal ERC	W	L	L

Above, the Ls are highlighted in yellow, and as can be seen in the goal ERC in 15, the algorithm that produces the goal ERC must capture the fact that ML and MR both must be dominated in F(A). That is, it must ensure that the Ls are not lost by joining with non-L members.

5 Joining sets of ERCs

The algorithm for producing a set of ERCs from the sets of ERCs produced from lexical hypotheses is shown below in pseudocode.

16. Algorithm for joining sets of ERCs
    For the sets of ERCs produced from consistent lexical hypotheses
    - For each constraint in the constraint set
        - Check to see if there are ERCs with an L in given constraint in each of the ERC sets
            - Select one of these ERCs from each of the constraint sets
            - Join the selected ERCs
        - Do this joining for every combination of ERCs across the constraint sets

The algorithm works by focusing on necessary domination. A constraint C that has an L in an ERC in each of the ERC sets must be dominated by some set of constraints. Joining all of these ERCs across the ERC sets will leave an L in that constraint, ensuring that in the produced set there is an ERC that captures the fact that that constraint must be dominated.

The algorithm also never extracts incorrect ranking information. The target lexicon is always a consistent lexicon and because the join only produces ERCs that are entailed by the individual joinands, the produced set will be entailed by the target lexicon. Furthermore, the join of a set of ERCs using the above algorithm will always produce exactly the shared ranking information across the ERC sets, guaranteeing that no information is lost.[2]

Now, stepping through the algorithm in an example, we return to the example above with the two ERC sets stemming from Hypothesis(A) and Hypothesis(B), the algorithm proceeds through each of the constraints, starting with F(A). Reproduced below are the two ERC sets, and as can be seen, neither ERC set contains an ERC that has an L in F(A).

After having determined that there are no ERCs to join from F(A), the algorithm moves on to ML and determines that ERC 2 and ERC 4 both contain an

---

[2] Under the condition that the ERC sets are fusional closures. See Merchant 2008 or Merchant (to appear) for extensive discussion of the fusional closure and its relation to the joining of ERC sets, along with an algorithm for producing the fusional closure. The key property that fusional closures bring is that every ERC entailed by an ERC set is entailed by a single ERC in that ERC set.

L, highlighted in yellow in the tableaux. Having selected these two ERCs, the algorithms joins them using the pair-wise join operator producing ERC2⊕ERC4, which is the ERC <W,L,W>, shown below.

Finally, the algorithm moves to constraint MR, recognizing that ERC1 and ERC3 have an L in MR, here marked in the tableaux below in orange, and joins those two ERCs, producing ERC1⊕ERC3, <W,L,W>.

17

Hyp(A)	F(A)	ML	MR
ERC 1	e	W	L
ERC 2	W	L	W

18

Hyp(B)	F(A)	ML	MR
ERC 3	W	W	L
ERC 4	e	L	W

19

	F(A)	ML	MR
2⊕4	W	L	W
1⊕3	W	W	L
(*2⊕4*)o(*1⊕3*)	W	L	L

At this point the algorithm is done, having produced two ERCs that are exactly the shared ranking information of the two original ERC sets. Recall that the goal ERC was <W, L, L>, representing that F(A) must dominate both ML and MR (though is silent on ML and MRs respective ranking). This can be seen more clearly by fusing the two, as shown immediately below the two joined ERCs in 19, producing an equivalent ERC to the two joined ERCs. This fused ERC is <W, L, L>, exactly the ranking information one can determine from the original two ERC sets.

5.1 Proof that joining sets of ERCs produces all shared ranking information

Having shown that this algorithm works in this specific case, I now turn to showing that it works in the general case, for any sets of ERCs produced from any consistent set of lexica. I will show this in two parts: first, by demonstrating that the set of ERCs produced by the join algorithm from a given set of ERCs is entailed by each of the original ERC sets. And second, by showing that any entailed ERC of the original ERC sets is entailed by the join of the ERC sets. These two facts together demonstrate that the join algorithm is maximally informative, producing all and only those ranking requirements imposed by the original ERC sets.

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

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

Theorem: for a set $\{A_i\}$ of fusional closures[3] of ERC sets and any ERC x such that each A_i entails x, then the set of ERCs $J(A_i)$ produced from the join algorithm on these fusionally closed sets also entails x.

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

6 Ranking and lexical learning algorithm applied

The join algorithm presented here produces ranking information from consistent lexica, tested using MRCD. This ranking information production algorithm can be augmented with a lexical learning component that uses the ranking restrictions produced from the join algorithm to restrict which lexica are consistent (as in Merchant 2008). This can lead to a virtuous cycle of lexical learning and ranking determination. The join algorithm can produce ranking restrictions which feed into the lexical learning component restricting which lexica are consistent. Having fewer consistent lexica then can produce more

[3] One can replace this requirement with the equivalent one that each ERC set is such that any ERC entailed by the ERC set is entailed by a single ERC in the ERC set. See footnote 1 for further comments.

restrictive lexica and a more informative join. This back and forth can then be used by the learner to home in on the correct language quickly.

In fact, this expanded algorithm of lexical setting and ranking determination was applied to more complex systems, first to a system with two features and six constraints, four of which were markedness constraints and two faithfulness constraints. The learning algorithm learned all the languages of the system, though this system was too simple to require the join algorithm – setting of underlying forms by checking consistency of lexica was sufficient to learn all the languages. In this system, checking for consistency and using the fully specified overt forms provided enough information to produce a ranking for each language. The algorithm was also efficient in that it only considered eight lexica for any given contrast pair, and so there was no combinatorial explosion of checking all possible underlying forms for either contrast pairs or the entire lexicon at once.

Going to a more complex system than the one above did yield a set of languages that the join algorithm provided crucial information for. The system had three features, nine constraints, six of which were markedness and three faithfulness constraints. In this system, learning ranking information from fully specified forms was not sufficient to learn all of the languages. For some, the join algorithm provided crucial information that allowed the learner to determine both the lexica and a correct ranking consistent with the overt forms of the language. The algorithm learned all languages of this system using the augmented learning algorithm that includes the join except for one language on which it failed because a subset problem related issue.

7 Conclusion

Knowledge of the ranking and lexicon can be determined from not-fully specified forms and from not-fully specified mappings using the join algorithm presented above. In previous research, ranking production algorithms assumed that underlying forms must be fully specified to produce accurate information about the target ranking. As shown, this assumption is false, one can fruitfully consider multiple hypotheses for a given overt form, and, as long as one of the considered hypotheses is correct, the learner can gain useful information about the target grammar, information that is useful for, of course, learning the ranking, but also for learning underlying forms by providing further constraints on lexical hypotheses that would not otherwise have been available.

Besides being applicable to learning underlying forms, this approach can be applied to a wider range of learning phenomena, phenomena in which a learner has multiple competing hypotheses, only one of which is correct. For example, this could be applied to learning parses of overt forms that have multiple consistent parses (Jason Riggle (p.c.) has undertaken such work), or to noisy data.

Potential mappings can yield information about the ranking, and potential lexica can yield information about the target lexicon. As discussed, there is a close relationship between possible lexica and possible mappings for a language: the information gained from potential mappings from selected overt forms can

place restrictions on other overt forms increasing likelihood of setting features, and the newly set features then restrict possible mappings increasingly the likelihood of gaining ranking information, and it is, of course, through the structure of the grammar, using ERCs and the join, that this information is gained.

References

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

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

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

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

Merchant, N. 2008. Discovering Underlying Forms: Contrast Pairs and Ranking. Doctoral dissertation, Department of Linguistics, Rutgers University. ROA 964.

Merchant, N. & B. Tesar. 2006a. Using local lexica to learn ranking information and underlying feature values. In *Proceedings of the 4th North American Phonology Conference.*

Merchant, N. & B. Tesar. 2006b. Learning underlying forms by searching restricted lexical subspaces. In *Proceedings of the 40th Conference of the Chicago Linguistics Society.*

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

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

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

Prince, A., & B. Tesar. 2004. Learning phonotactic distributions. In *Constraints in Phonological Acquisition*, ed. by R. Kager, J. Pater, and W. Zonneveld, 245-291. Cambridge: Cambridge University Press.

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

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

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

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