

Surgery in Language Learning

Bruce Tesar, John Alderete, Graham Horwood,
Nazarré Merchant, Koichi Nishitani and Alan Prince
Rutgers University

1. Introduction

The architecture of generative phonology brings with it a difficult challenge for any learner: underlying forms must be acquired at the same time as the phonology – the system of rules or constraint-rankings. Yet, each depends on the other, and neither is known in advance. If the learner had prior knowledge of the underlying forms, then the constraint-ranking could be determined by familiar procedures. If the learner knew the ranking, then the range of viable underlying forms would be greatly limited, simplifying the process of finding them. In addition, the learner must identify a phonology which is maximally restrictive, so that distributional restrictions implicit in the data are enforced, rather than being portrayed as accidental gaps in the lexicon. Since it is impossible to explore all possible lexicon-phonology pairings, an effective learner must use an incremental strategy which goes back and forth between hypotheses about the lexicon and hypotheses about the phonology, testing and improving each until a satisfactory match is found. A principal issue in designing any such procedure is deciding what to do when the mapping fails: should the lexicon be changed or should the phonology be changed?

2. The learning problem

* The authors would like to thank audiences at UCLA and Rutgers University for stimulating comments on prior versions of this work. This material is based upon work supported by the National Science Foundation under Grant No. BCS-0083101, and by NIH NRSA Training Grant No. 1-T32-MH-19975-05. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or of the National Institutes of Health. Any errors are the responsibility of the authors.

© 2003 Bruce Tesar, John Alderete, Graham Horwood, Nazarré Merchant, Koichi Nishitani and Alan Prince. *WCCFL 22 Proceedings*, ed. G. Garding and M. Tsujimura, pp. 477-490. Somerville, MA: Cascadilla Press.

2.1. An illustration: Cupeño

The kind of phenomenon considered here is illustrated in (1) with some data from Cupeño. The data are taken from Alderete (2001), but originate with Hill and Nolasquez (1973), Crowhurst (1994), and unpublished field notes of Jane Hill.

- (1) a. /ʔa'yu/ + /-qa'/ → ʔAyuqa
 '(He) wants'
- b. /max/ + /-qa'/ → maxQA
 'giving'
- c. /max/ + /-əm/ → MAXəm
 'Give!'

The generalization of the Cupeño stress pattern is to place stress on an accented root syllable (1a), else an accented suffix syllable (1b), else the left most syllable (1c). The key thing to observe in these examples is that both the morpheme /max/ and the morpheme /-qa'/ alternate: they are stressed in one context, and unstressed in another. Yet, the analysis in (1) requires that /max/ be unaccented underlyingly, and /-qa'/ be accented underlyingly. To arrive at this analysis, the learner must infer the correct mapping giving rise to the root-controlled stress pattern, and infer the assignment of underlying accents to the morphemes.

2.2. Learning main stress

Main stress is an interesting case for the learning of phonological underlying forms. In some languages, main stress is purely predictable, while in others, main stress can be contrastive. Thus, one task facing the learner is to determine whether stress needs to be crucially specified in the language; if stress is predictable, the learner should infer a mapping enforcing the predictable stress placement. Main stress is also culminative; every word will have exactly one main stress, regardless of how many underlying accents there are (zero, one, or more than one). Main stress will appear even in the complete absence of lexical accents. Thus, the learner cannot pursue a strategy as simple as putting an accent on every stressed syllable.

These properties make main stress a good starting point for an investigation of the learning of lexical specification in general. While other features may not necessarily share the culminativity of main stress, it is typically the case that the correct underlying forms for alternating morphemes cannot be determined in isolation. The different environments

in which the morpheme appears must be considered, and that in turn can require simultaneous consideration of the underlying forms of the other morphemes constituting said environments. There is reasonable cause to believe that insight into the learning of main stress will apply to the learning of phonological underlying forms in general.

The learning problem investigated here isolates the interdependence of phonological mapping and underlying accents, abstracting away from other nontrivial challenges of learning. We assume that the learner has already solved (correctly) the problem of segmenting words into constituent morphemes, and has constructed paradigms of word forms related by shared morphemes. Each morpheme has exactly one underlying phonological form, and the phonological mapping is analyzed as an Optimality Theoretic system. The learner has access to all of the universal constraints prior to the onset of learning.

3. The PAKA world

We constructed a specific linguistic system of stress grammars, one which allows both grammars with predictable stress and grammars with lexically specified stress. This system allowed us to investigate a formally specified problem. The system has the twin virtues of being simple enough to investigate exhaustively while still capturing the essence of the larger problem of interest.

For reasons that will become apparent shortly, this system will be referred to in this paper as the PAKA system.

3.1. The system: Candidates and constraints

Every language in the PAKA system contains two roots, /pa/ and /ba'/, and two suffixes, /-ka/ and /-ga'/. A total of four words can be formed by root+suffix combinations: paka, paga, baka, and baga. Those four words exist in every language. What differs from language to language is how each of the words is stressed.

Only one feature can vary cross-linguistically in the underlying specification of a morpheme: the presence or absence of an accent. Richness of the base requires the consideration of both accented and unaccented roots and suffixes. In generating languages, we realized the rich base by putting underlying accents on the morphemes with voiced consonants, /ba/ and /-ga/, and by not putting accents on the morphemes with voiceless accents, /pa/ and /-ka/.¹

1. This correlation of voice and accent is a presentational convention, not a linguistic claim of any sort.

The system has four constraints; two are markedness constraints, regulating the alignment of stress with word edges, and two are faithfulness constraints, regulating the realization of accent as surface stress.

- (2) MAINLEFT (ML): stress the leftmost syllable.
 MAINRIGHT (MR): stress the rightmost syllable.
 FAITHACCENT (F): stress an accented syllable.
 FAITHACCENTROOT (FR): stress an accented root syllable.

The four constraints, used with the monosyllabic roots and suffixes, can distinguish five different languages.²

3.2. The typology

The five languages of the system are as follows. In the surface forms, stress is indicated via capitalization. After each ranking, the essential underlying accentuations (if there are any) are listed.

- | | |
|--|--|
| (3) PAka PAga BAka BAga
ML >> {MR, F, FR} | predictable left stress |
| (4) paKA paGA baKA baGA
MR >> {ML, F, FR} | predictable right stress |
| (5) PAka paGA BAka BAga
F >> ML >> {MR, FR} | full accentual contrast, default left
/pa, ba', -ka, -ga'/ |
| (6) paKA paGA BAka baGA
F >> MR >> {ML, FR} | full accentual contrast, default right
/pa, ba', -ka, -ga'/ |
| (7) paKA paGA BAka BAga
FR >> MR >> {ML, F} | contrast in roots only, default right
/pa, ba'/ |

The five languages differ in degree of restrictiveness. The first two languages are the most restrictive, having completely predictable stress and no accentual contrast; with faithfulness completely dominated by markedness, any assignment of underlying accents yields the same surface

2. The constraints are capable of distinguishing eight different languages, but realizing the full eight requires the use of prefixes as well as suffixes, something outside the scope of this paper.

stress pattern. The next two languages are the least restrictive, having full accentual contrast, with general FAITHACCENT top-ranked, so the underlying specifications of all morphemes are relevant. The final language has active FAITHACCENTROOT, so only roots have an accentual contrast; it is of intermediate restrictiveness.

3.3. The learning problem restated

The input to the learner consists of the surface forms, fully morphologically segmented. For the language listed in (5), the input would be:

(8) PA-ka pa-GA BA-ka BA-ga

The correct output of learning would be a ranking and a lexicon:

(9) $F \gg ML \gg \{MR, FR\}$ /pa, ba', -ka, -ga'/

The system has a search space of 384 possible grammars, resulting from a free combination of $4! = 24$ possible rankings, and $2^4 = 16$ possible lexica. For each language, the learner must search this space to find a most restrictive grammar properly reproducing the surface forms.

4. Overcoming the mutual dependence of mapping and lexicon

4.1. Search strategy: Change the ranking first

The phonological mapping, realized by the constraint ranking, and the underlying forms are mutually dependent. Inferring each seems to require the other; knowing which morphemes to assign accents depends upon what the default alignment of the ranking is, while determining whether to rank a faithfulness constraint high depends upon which morphemes have accents. The learner starts out knowing neither the correct ranking nor the underlying forms, and thus it will need to play each off of the other until an appropriate combination is found.

The mutual dependence raises interesting questions for the learner in searching the space of grammars. Suppose that the learner has a hypothesized grammar in mind, and it fails to match the data in some way. The grammar clearly needs to be changed, but how? Should the learner modify the ranking or the lexicon of underlying forms?

We pursue here the following strategy: when the learner encounters a mismatch between the data and their hypothesized grammar, the learner should first attempt to modify the ranking. If modifying the ranking cannot

resolve the problem, only then will the learner attempt to modify the lexicon.

4.2. Learning restrictive rankings

We propose implementing this strategy in a particular way, one that draws upon much existing work in language learning. The learner will relate the data to possible rankings by accumulating a list of winner-loser pairs, also known as ranking arguments, via the procedure known as Multi-Recursive Constraint Demotion (MRCD) (Tesar 1997b; 2000). Given a list of winner-loser pairs, the learner will use the Biased Constraint Demotion (BCD) procedure to construct the most restrictive ranking consistent with those pairs (Prince and Tesar 1999) (see also Hayes 1999). Modifying the ranking is carried out by adding a new winner-loser pair to the list, and constructing a new ranking.

A winner-loser pair consists of a description taken to be grammatical (the winner), and a description that competes with the winner (the loser). This is conveniently expressed in comparative tableau format (Prince 2000), like in Table 1.

Table 1 A winner-loser pair: paGA is the winner, PAga is the loser.

lexicon	win ~ lose	ML	MR	F	FR
/pa/, /-gaʔ/	paGA ~ PAga	L	W	W	

‘W’ marks constraints preferring the winner over the loser, ‘L’ marks constraints preferring the loser over the winner, and blanks indicate constraints assigning equal numbers of violations. The pair in Table 1 indicates that at least one of MR and F must dominate ML in the ranking.

The learner determines a ranking by accumulating a list of winner-loser pairs, here termed the *support*. Given a support, BCD can be used to find the most restrictive ranking consistent with the support. BCD imposes restrictiveness via a bias towards ranking markedness constraints as high as possible, and faithfulness constraints as low as possible. This can be illustrated using the support in Table 2.

Table 2 A support consisting of three winner-loser pairs.

lexicon	win ~ lose	ML	MR	F	FR
/pa/, /-gaʔ/	paGA ~ PAga	L	W	W	
/baʔ/, /-ka/	BAka ~ baKA	W	L	W	W
	PAka ~ paKA	W	L		

BCD starts by identifying those constraints available to be placed at the top of the constraint hierarchy. A constraint is available if it has no Ls in its

column. In this example, there are two: F and FR. If at least one of the constraints is a markedness constraint, BCD will place only available markedness constraints at the top of the hierarchy. In this instance, both available constraints are faithfulness constraints. BCD then chooses which one to place in the hierarchy based upon which one will make it possible to rank a markedness constraint next. In this case, BCD chooses the constraint F. Because F prefers the winner in the first two pairs, ranking it above all else satisfies the first two pairs, allowing their removal from the list. This means that if F is ranked highest, then ML will be free to be ranked next in the hierarchy; the L in the ML column comes from the first pair, and would be accounted for by the fact that F will dominate ML. If, instead, FR were to be ranked first, it would only account for the second winner-loser pair, and both markedness constraints would still have an L mark unaccounted for. Thus, BCD ranks F at the top of the hierarchy, and removes the first two pairs from the list, leaving the remaining pairs as shown in Table 3 (with the already-ranked constraints removed).

Table 3 Remaining winner-loser pairs after ranking F

lexicon	win ~ lose	ML	MR	FR
	PAka ~ paKA	W	L	

BCD now repeats the procedure on the remaining pairs. Two constraints are now available for ranking: ML and FR. Because ML is a markedness constraint, while FR is a faithfulness constraint, ML is placed into the ranking and FR is not (yet). At this point, BCD has constructed the partial constraint hierarchy shown in (10).

(10) F \gg ML

ML prefers the winner in the remaining pair, so it may be removed from the list. At this point, both remaining constraints are available to be ranked. The bias towards markedness above faithfulness mandates that the remaining markedness constraint, MR, be placed into the ranking before FR. Thus, BCD ultimately constructs the hierarchy in (11).

(11) F \gg ML \gg MR \gg FR

4.3. Detecting inconsistency

The stated strategy crucially relies on the learner being able to determine when no amount of ranking modification will reconcile the learner with the data, given the learner's current hypothesized lexicon. That determination is made as a side effect of the use of BCD to construct a

ranking. When no ranking exists that is consistent with all of the data, the learner will end up with a list of winner-loser pairs that is *inconsistent*. BCD, in its effort to construct a ranking consistent with a winner-loser pair list, rapidly detects when such a list is inconsistent. This *inconsistency detection* (Tesar 1997a; 2000) occurs during BCD when unranked constraints remain, but none of the constraints are available to be placed into the ranking. Such a situation is shown in Table 4.

Table 4 An inconsistent set of winner-loser pairs

lexicon	win ~ lose	ML	MR
/pa/, /-ga/	paGA ~ PAga	L	W
/-ka/	PAka ~ paKA	W	L

Observe that every constraint in the tableau has at least one L in its column. Effectively, every constraint must be dominated by some other constraint, so no constraint is available to be placed into the ranking without violating the requirements of one of the pairs. If such a situation arises during the execution of BCD, the learner halts ranking construction, knowing that no constraint hierarchy will satisfy all of the winner-loser pairs.

Inconsistency detection is the cue to the learner that the lexicon must be modified. The basic idea of modifying the lexicon in response to indications of inconsistency during ranking learning was proposed by Kager (1999). We propose the same basic idea in a different algorithmic setting, one which allows us to determine definitively whether or not the data are inconsistent with the hypothesized lexicon, rather than depending upon indirect diagnostics. To fully implement the algorithm, we supply a particular method for modifying the lexicon in response to inconsistency.

The strategy of modifying the lexicon in response to inconsistency raises a complication. The winner-loser pairs depend upon the particular hypothesized lexicon in use; modifying the lexicon can invalidate a winner-loser pair. Rather than discard the winner-loser pairs in question, we propose that the learner update them in response to lexicon modification, via a process known as *surgery*. Using surgery to preserve the winner-loser pairs not only prevents the learner from discarding valid information, but allows the learner to test proposed modifications to the lexicon on the spot.

5. Modifying the lexicon

The algorithm will be illustrated using the data from (5).

5.1. Paradigm analysis and the initial lexicon

The learner starts by analyzing the paradigm of data. For each morpheme, the learner examines all surface realizations, and checks to see

how they differ. In the present situation, a morpheme can only differ with respect to whether or not it is stressed, so the learner classifies the morphemes into three types: always stressed, always unstressed, and alternating. In the illustration, the learner classifies the morphemes as shown in (12).

(12) Stressed: {ba} Unstressed: {-ka} Alternating: {pa, -ga}

This classification provides the basis for both the initial lexicon and the subsequent search space of lexica to be considered. The learner proceeds on the following assumptions: it is safe to put an accent on an always stressed morpheme, and it is safe to leave an always unstressed morpheme unaccented. Thus, the learner puts accents on always stressed morphemes in the initial lexicon, and subsequent learning does not change the underlying forms of consistent morphemes (always stressed or always unstressed).

This reduces the search space of lexica; the learner is now only actively considering which alternating morphemes to accent. The learner's approach to the alternating morphemes is to initially accent none of them, and then respond to inconsistency by adding accents to alternating morphemes.

(13) Initial Lexicon: {pa ba' -ka -ga}

5.2. Searching lexical space

The learner constructs, as part of the paradigm analysis, a list of alternating morphemes. When the learner detects inconsistency and concludes that the lexicon must be changed, that change will consist of putting an accent on one of the alternating morphemes. The learner first tries putting an accent on the first morpheme in the list.³ If the added accent resolves the inconsistency, then it is kept as a permanent change to the lexicon. In this algorithm, once an added accent is kept, it will never be revoked by learning at a later time. If the added accent does not resolve the inconsistency, then the accent is retracted, and the learner next tries adding an accent to the next alternating morpheme on the list. This continues until either an added accent resolves the inconsistency, or the end of the list of alternating morphemes is reached. In the latter condition, the algorithm would simply declare failure of learning.

3. In the simulations discussed below, the learner ordered the alternating morphemes in the list according to the order in which they first appeared in the data. This is, in turn, dependent upon the order in which the data are presented. As discussed below, the simulations tried every possible ordering of each data set to ensure that success was not crucially dependent upon data order.

Note that if an alternating morpheme already has an accent on it (added and kept at an earlier point during learning), then accenting that morpheme will have no effect. The learner contends with this simply and directly: before adding an accent, it checks to ensure that the morpheme does not already have one.

5.3. Lexicon modification and surgery

The learner may modify the lexicon simply by altering an underlying form within it: here, by adding an accent to the underlying form for a morpheme. However, such a change has significant consequences for the list of winner-loser pairs being maintained by the learner. A winner-loser pair includes a pair of structural descriptions, and structural descriptions include the underlying forms for the morphemes in the input. These underlying forms are referenced by faithfulness constraints. Thus, altering the underlying form for a morpheme can change how some faithfulness constraints evaluate any candidate that contains the altered morpheme. For example, consider the winner-loser pair in Table 5.

Table 5 Winner-Loser pair before adding an accent to /-ga/

lexicon	win~lose	ML	MR	F	FR
/pa/, /-ga/	paGA ~ PAga	L	W		

Suppose the underlying form for the morpheme /-ga/ is altered by adding an accent, so that the new underlying form is /-ga'/. If we re-evaluate the candidates with respect to the new lexicon, we get the following very different winner-loser pair.

Table 6 The evaluation of constraint F changes as a result of the added accent.

lexicon	win~lose	ML	MR	F	FR
/pa/, /-ga'/'	paGA ~ PAga	L	W	W	

The question that arises is: what should the learner do with the list of winner-loser pairs once the lexicon is modified? It would be very simple to just throw away any pairs making reference to the altered morpheme, but that throws away comparisons (valuable ones, as we will see). We propose, instead, that whenever an underlying form for a morpheme is altered, each winner-loser pair making reference to that morpheme is immediately “adjusted” so that it matches the new underlying form. This process of adjustment is here labeled *surgery*.

Conveniently, surgery can be performed without any special surgery mechanism. All that is required is to reparse the output forms using the new

lexicon. Reparsing an output constructs a full structural description using the new underlying forms for the morphemes of the output, and recalculates the constraint violations of the description. By reparsing both the winner and the loser of any pair including a morpheme whose underlying form has changed, the learner “surgically alters” the pair so as to update it to the changed lexicon.

6. The benefits of surgery

Altering existing winner-loser pairs whenever the lexicon is modified allows the learner to retain the pairs. This is quite beneficial to a learner trying to determine how to modify the lexicon.

We will illustrate the usefulness of surgery by applying the algorithm to the following data in the following order: BAKa paGA PAka BAgA. Paradigm analysis reveals to the learner that there is one always-stressed morpheme, /ba/, and two alternating morphemes, /pa/ and /-ga/. Thus, the learner starts out with an accent only on /ba/, and an ordered list of alternating morphemes, {/pa/, /-ga/}. Applying MRCD with BCD results in the list of three winner-loser pairs shown in Table 7.

Table 7 Inconsistency indicates the lexicon must be changed.

lexicon	win ~ lose	ML	MR	F	FR
/ba ² /, /-ka/	BAka ~ baKA	W	L	W	W
/pa/, /-ga/	paGA ~ PAga	L	W		
	PAka ~ paKA	W	L		

This set of pairs is inconsistent, as can be readily seen from comparing the second and third pairs, which make opposite requirements of the relative ranking of ML and MR. This signals to the learner that the lexicon must be changed. As described above, the learner will modify things by adding an accent to an alternating morpheme: but which one?

The learner selects /pa/, the first alternating morpheme on the list. It adds an accent, and performs surgery on the winner-loser pairs containing /pa/ (the learner reparses them). The new list of winner-loser pairs is shown in Table 8.

Table 8 After accenting /pa/, the list is still inconsistent.

lexicon	win ~ lose	ML	MR	F	FR
/ba ² /, /-ka/	BAka ~ baKA	W	L	W	W
/pa ² /, /-ga/	paGA ~ PAga	L	W	L	L
	PAka ~ paKA	W	L	W	W

Unfortunately, the modification of the lexicon did not resolve the inconsistency; the second and third winner-loser pairs still directly clash. Fortunately, the learner can use this fact to conclude that adding an accent to /pa/ was probably a mistake; an accent should have been added to a different alternating morpheme instead. This insight can only be achieved if the learner retains and updates the winner-loser pairs via surgery. If the learner had simply discarded all winner-loser pairs containing /pa/, it would not be possible at this point to determine whether the modification of the lexicon addressed the inconsistency or not.

Since the accent on /pa/ failed to resolve the inconsistency, the learner retracts that accent, and moves on to the next alternating morpheme in the list, /-ga/. It places an accent on /-ga/, and performs surgery again, resulting in the list of winner-loser pairs shown in Table 9.

Table 9 The accent on /-ga/ resolves the inconsistency.

lexicon	win ~ lose	ML	MR	F	FR
/baʔ/, /-ka/	BAka ~ baKA	W	L	W	W
/pa/, /-gaʔ/	paGA ~ PAga	L	W	W	
	PAka ~ paKA	W	L		

The new accent resolves the inconsistency, as all three winner-loser pairs can now be explained by the ranking in (14).

(14) F \gg ML \gg MR \gg FR

The learner has now successfully learned the language, having arrived at both the correct ranking and the correct lexicon.

7. Evaluating with PAKA: Simulation results

We verified that the algorithm works for the five PAKA languages via exhaustive simulations. We ran the learning algorithm on the data for each of the five languages, for each of the 24 possible data orders.⁴ The algorithm learned a correct grammar in each case.

How much effort is required by the learner to reach the correct grammar? We measured the performance of the learner in terms of the number of distinct grammars constructed by the learner on the way to

4. The learner receives the full paradigm of data at once, so data order is under the control of the learner.

reaching the correct one.⁵ A new grammar is constructed whenever the learner adds a winner-loser pair or modifies the lexicon. The system has 384 possible grammars. If the learner has to construct and evaluate a significant proportion of the possible grammars, then the approach is unlikely to scale up well to more complex systems.

As it happens, the initial lexicon constructed by the lexicon via paradigm analysis is a correct lexicon for three of the five languages (languages (3), (4), and (7)). For those languages, no inconsistency was ever reached, and thus no surgery was ever necessary. The maximum number of grammars constructed by the learner for any data order for any of those three languages was 2. The efficiency of the learner in these cases may be attributed to the efficiency of biased constraint demotion in finding a restrictive constraint ranking, given accurate underlying forms.

For the other two languages, the learner always had to modify the lexicon on the way to finding a correct grammar. However, the maximum number of grammars constructed by the learner for any data order for either of those languages was 6. This is far short of the 384 possible grammars. The efficiency of the learner in these cases may be attributed to the speed with which BCD detects inconsistency, combined with the retention of winner-loser pairs (permitted by surgery), allowing the learner to evaluate different lexical hypotheses on the spot.

8. Conclusions and discussion

The learner described here combines several computational elements in a particular way, and they work together to solve the problem. Paradigm analysis allows a significant reduction in the search space being actively combed by the learner, to only the possible assignment of accents to alternating morphemes. BCD quickly searches the space of possible rankings to either find the most restrictive consistent one or conclude that the winner-loser pairs are inconsistent, indicating that the lexicon must be changed. Surgery allows the learner to retain winner-loser pairs in the face of an altered lexicon, which makes it possible for the learner to test changes to the lexicon on the spot.

While we expect all of these mechanisms to be included in the solution to more complex lexical learning problems, we don't expect the current algorithm to succeed unaltered. The PAKA problem was selected as the simplest system we could construct that contained the key elements of mutual dependence between ranking and lexicon. As such, it is a good place to start, but almost certainly it abstracts away from some of the challenges

5. The initial ranking constructed by the learner, generated from an empty list of winner-loser pairs, is not a possible final grammar and is not included in the count.

of more complex phonological learning problems. In the PAKA system, paradigm analysis makes it possible to resolve inconsistency by the addition of a single accent, regardless of data order. Larger systems involving more alternating morphemes may not share that property, presenting the opportunity to uncover further lexical search strategies with benefits that cannot be seen in a system as simple as the one investigated here. It may be necessary for the learner to contemplate adding more than one accent at once, or to consider retracting an accent added at an earlier point during learning. Such modifications might add computational complexity to the search. Alternately, more complex systems might have further structure within the paradigm of data, such that having the learner choose a particular order of the data would have advantages.

References

- Alderete, John. 2001. Root-controlled accent in Cupeño. *Natural Language and Linguistic Theory* 19:455-502.
- Crowhurst, Megan. 1994. Foot extrametricality and template mapping in Cupeño. *Natural Language and Linguistic Theory* 12:177-202.
- Hayes, Bruce. 1999. Phonological acquisition in Optimality Theory: The early stages. Ms., UCLA. ROA-327.
- Hill, Jane, and Nolasquez, Rosinda. 1973. *Mulu'wetam: The First People*. Banning: Malki Museum Press.
- Kager, Rene. 1999. *Optimality Theory*. Cambridge: Cambridge University Press.
- Prince, Alan. 2000. Comparative tableaux. Ms., Rutgers University, New Brunswick. ROA-376.
- Prince, Alan, and Tesar, Bruce. 1999. Learning phonotactic distributions. Technical Report RuCCS-TR-54, Rutgers Center for Cognitive Science, Rutgers University, New Brunswick. ROA-353.
- Tesar, Bruce. 1997a. Using the mutual inconsistency of structural descriptions to overcome ambiguity in language learning. In *Proceedings of the North East Linguistic Society 28*, eds. Pius N. Tamanji and Kiyomi Kusumoto, 469-483. Amherst, MA: GLSA, University of Massachusetts.
- Tesar, Bruce. 1997b. Multi-Recursive Constraint Demotion. Ms., Rutgers University, New Brunswick. ROA-197.
- Tesar, Bruce. 2000. Using inconsistency detection to overcome structural ambiguity in language learning. Technical Report RuCCS-TR-58, Rutgers Center for Cognitive Science, Rutgers University, New Brunswick, NJ. ROA-426.