CONSTRAINT PROPAGATION IN KIMMO SYSTEMS

G. Edward Barton, Jr.
M.I.T. Artificial Intelligence Laboratory
545 Technology Square
Cambridge, MA 02139

ABSTRACT

Taken abstractly, the two-level (Kimmo) morphological framework allows computationally difficult problems to arise. For example, $N + 1$ small automata are sufficient to encode the Boolean satisfiability problem (SAT) for formulas in N variables. However, the suspicion arises that natural-language problems may have a special structure — not shared with SAT — that is not directly captured in the two-level model. In particular, the natural problems may generally have a modular and local nature that distinguishes them from more "global" SAT problems. By exploiting this structure, it may be possible to solve the natural problems by methods that do not involve combinatorial search.

We have explored this possibility in a preliminary way by applying *constraint propagation* methods to Kimmo generation and recognition. Constraint propagation can succeed when the solution falls into place step-by-step through a chain of limited and local inferences, but it is insufficiently powerful to solve unnaturally hard SAT problems. Limited tests indicate that the constraint-propagation algorithm for Kimmo generation works for English, Turkish, and Warlpiri. When applied to a Kimmo system that encodes SAT problems, the algorithm succeeds on "easy" SAT problems but fails (as desired) on "hard" problems.

INTRODUCTION

A formal computational model of a linguistic process makes explicit a set of assumptions about the nature of the process and the kind of information that it fundamentally involves. At the same time, the formal model will ignore some details and introduce others that are only artifacts of formalization. Thus, whenever the formal model and the actual process seem to differ markedly in properties, a natural assumption is that something has been missed in formalization — though it may be difficult to say exactly what.

When the difference is one of worst-case complexity, with the formal framework allowing problems to arise that are too difficult to be consistent with the received difficulty of actual problems, one suspects that the natural computational task might have significant features that the formalized version does not capture and exploit effectively. This paper introduces a *constraint propagation method* for "two-level" morphology that represents a preliminary attempt to exploit the features of *local information flow* and *linear separability* that we believe are found in natural morphological-analysis problems. Such a local character is not shared by more difficult computational problems such as Boolean satisfiability, though such problems can be encoded in the unrestricted two-level model. Constraint propagation is less powerful than backtracking search, but does not allow possibilities to build up in combinatorial fashion.

TWO-LEVEL MORPHOLOGY

The "two-level" model of morphology developed by Kimmo Koskenniemi is attractive for putting morphological knowledge to use in processing. Two-level rules mediate the relationship between a *lexical string* made up of morphemes from the dictionary and a *surface string* corresponding to the form a word would have in text. Equivalently, the rules correspond to *finite-state transducers* that

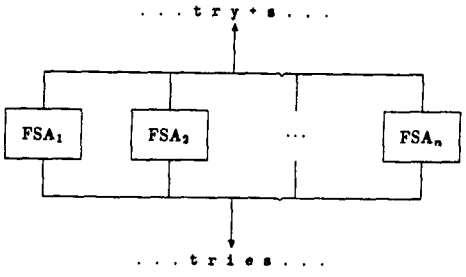

Figure 1: The automaton component of the Kimmo system consists of several two-headed finite-state automata that inspect the lexical/surface correspondence in parallel. The automata move together from left to right. (From Karttunen, 1983:176.)

```
ALPHABET x y z T F - ,

    ANY =
    END
```

```
"x-consistency" 3 3
     x  x  =
     T  F  =
1:   2  3  1
2:   2  0  2
3:   0  3  3

"y-consistency" 3 3
     y  y  =
     T  F  =
1:   2  3  1
2:   2  0  2
3:   0  3  3

"z-consistency" 3 3
     z  z  =
     T  F  =
1:   2  3  1
2:   2  0  2
3:   0  3  3

"satisfaction" 3 4
     =  =  -  ,
     T  F  -  ,
1.   2  1  3  0
2:   2  2  2  1
3.   1  2  0  0

END
```

Figure 2: This is the complete Kimmo generator system for solving SAT problems in the variables x, y, and z. The system includes a consistency automaton for each variable in addition to a satisfaction automaton that does not vary from problem to problem.

can be used in generation and recognition algorithms as implemented in Karttunen's (1983) Kimmo system (and others). As shown in Figure 1, the transducers in the "automaton component" ($\approx$ 20 for Finnish, for instance) all inspect the lexical/surface correspondence at once in order to implement the insertions, deletions, and other spelling changes that may accompany affixation or inflection. Insertions and deletions are handled through *null characters* that are visible only to the automata. A complete Kimmo system also has a "dictionary component" that regulates the sequence of roots and affixes at the lexical level.

Despite initial appearances to the contrary, the straightforward interpretation of the two-level model in terms of finite-state transducers leads to generation and recognition algorithms that can theoretically do quite a bit of backtracking and search. For illustration we will consider the Kimmo system in Figure 2, which encodes Boolean satisfiability for formulas in three variables x, y, and z. The Kimmo generation algorithm backtracks extensively while determining truth-assignments for formulas according to this system. (See Barton (1986) and references cited therein for further details of the Kimmo system and of the system in Figure 2.)

Taken in the abstract, the two-level model allows computationally difficult situations to arise despite initial appearances to the contrary, so why shouldn't they also turn up in the analysis of natural languages? It may be that they do turn up; indeed, the relevant mathematical reductions are abstractly based on the Kimmo treatment of vowel harmony and other linguistic phenomena. Yet one feels that the artificial systems used in the mathematical reductions are unnatural in some significant way — that similar problems are not likely to turn up in the analysis of Finnish, Turkish, or Warlpiri. If this is so, then the reductions say more about what is thus-far unexpressed in the formal model than about the difficulty of morphological analysis; it would be impossible to crank the difficult problems through the formal machinery, if the machinery could be infused with more knowledge of the special properties of natural language.[1]

MODULAR INFORMATION STRUCTURE

The ability to use particular representations and processing methods is underwritten by what may be called the "information structure" of a task — more abstract than a particular implementation, and concerned with such questions as whether a certain body of information suffices for making certain decisions, given the constraints of the problem. What is it about the information structure of morphological systems that is not captured when they are encoded

[1]The systems under consideration in this paper deal with orthographic representations, which are somewhat remote from the "more natural" linguistic level of phonology and contain both more and less information than phonological representations.

as Kimmo systems? Are there significant locality principles and so forth that hold in natural languages but not in mathematical systems that encode CNF Boolean satisfaction problems (SAT)? Perhaps a better understanding of the information relationships of the natural problem can lead to more specialized processing methods that require less searching, allow more parallelism, run more efficiently, or are more satisfying in some other way.

A lack of *modular information structure* may be one way in which SAT problems are unnatural compared to morphological-analysis problems. Making this idea precise is rather tricky, for the Kimmo systems that encode SAT problems are modular in the sense that they involve various independent Kimmo automata assembled in the usual way. However, the essential notion is that the Boolean satisfaction problem has a more interconnected and "global" character than morphological analysis. The solution to a satisfaction problem generally cannot be deduced piece by piece from local evidence. Instead, the acceptability of each part of the solution may depend on the whole problem. In the worst case, the solution is determined by a complex conspiracy among the problem constraints instead of being composed of independently derivable subparts. There is little alternative to running through the possible cases in a combinatorial way.

In contrast to this picture, in a morphological analysis problem it seems more likely that some pieces of the solution can be read off relatively directly from the input, with other pieces falling into place step-by-step through a chain of limited and local inferences and without the kind of "argument by cases" that search represents. We believe the usual situation is for the various complicating processes to operate in separate domains — defined for instance by separate feature-groups — instead of conspiring closely together.

The idea can be illustrated with a hypothetical language that has no processes affecting consonants but several right-to-left harmony processes affecting different features of vowels. By hypothesis, underlying consonants can be read off directly. The right-to-left harmony processes mean that underlying vowels cannot always be identified when the vowels are first seen. However, since the processes affect different features, uncertainty in one area will not block conclusions in others. For instance, the processing of consonants is not derailed by uncertainty about vowels, so information about underlying consonants can potentially be used to help identify the vowels. In such a scenario, the solution to an analysis problem is constructed more by superposition than by trying out solutions to intertwined constraints.

A SAT problem can have either a local or global information structure; not all SAT problems are difficult. The unique satisfying assignment for the formula $(\overline{y} \vee z)\&(x \vee y)\&\overline{x}$ is forced piece by piece; the conjunct $\overline{x}$ forces x to be false, so y must be true, so finally z must be true. In contrast, it is harder to see that the formula

$$(x \vee y \vee z)\&(\overline{x} \vee \overline{z})\&(\overline{x} \vee z)\&(\overline{y} \vee \overline{z})\&(\overline{y} \vee z)\&(\overline{z} \vee y)$$

is unsatisfiable. The problem is not just increased length; a different method of argument is required. Conclusions about the difficult formula are not forced step by step as with the easy formula. Instead, the lack of "local information channels" seems to force an argument by cases.

A search procedure of the sort used in the Kimmo system embodies few assumptions about possible modularity in natural-language phonology. Instead, the implicit assumption is that any part of an analysis may depend on anything to its left. For example, consider the treatment of a right-to-left long-distance harmony process, which makes it impossible to determine the interpretation of a vowel when it is first encountered in a left-to-right scan. Faced with such a vowel, the current Kimmo system will choose an arbitrary possible interpretation and arrange for eventual rejection if the required right context never shows up. In the event of rejection, the system will carry out chronological backtracking until it eventually backs up to the erroneous choice point. Another choice will then be made, but the entire analysis to the right of the choice point will be recomputed — thus revealing the implicit assumption of possible dependence.

By making few assumptions, such a search procedure is able to succeed even in the difficult case of SAT problems. On the other hand, if modularity, local constraint, and limited information flow are more typical than difficult global problems, it is appropriate to explore methods that might reduce search by exploiting this aspect of information structure.

We have begun exploring such methods in a preliminary and approximate way by implementing a modular, non-searching *constraint-propagation algorithm* (see Winston (1984) and other sources) for Kimmo generation and recognition. The deductive capabilities of the algorithm are limited and local, reflecting the belief that morphological analyses can generally be determined piece by piece through local processes. The automata are largely decoupled from each other, reflecting an expectation that phonological constraints generally will not conspire together in complicated ways.

The algorithm will succeed when a solution can be built up, piece by superimposed piece, by individual automata — but by design, in more difficult cases the constraints of the automata will be enforced only in an approximate way, with some nonsolutions accepted (as is usual

with this kind of algorithm). In general, the guiding assumption is that morphological analysis problems actually have the kind of modular and superpositional information structure that will allow constraint propagation to succeed, so that the complexity of a high-powered algorithm is not needed. (Such a modular structure seems consonant with the picture suggested by autosegmental phonology, in which various separate tiers flesh out the skeletal slots of a central core of CV timing slots; see Halle (1985) and references cited there.)

SUMMARIZING COMBINATIONS OF POSSIBILITIES

The constraint-propagation algorithm differs from the Kimmo algorithms in its treatment of nondeterminism. In terms of Figure 1, nondeterminism cannot arise if both the lexical surface strings have already been determined. This is true because a Kimmo automaton lists only one next state for a given lexical/surface pair. However, in the more common tasks of generation and recognition, only one of the two strings is given. The generation task that will be the focus here uses the automata to find the surface string (*e.g.* `tries`) that corresponds to a lexical string (*e.g.* `try+s`) that is supplied as input.

As the Kimmo automata progress through the input, they step over one lexical/surface pair at a time. Some lexical characters will uniquely determine a lexical/surface pair; in generation from `try+s` the first two pairs must be `t/t` and `r/r`. But at various points, more than one lexical/surface pair will be admissible given the evidence so far. If `y/y` and `y/i` are both possible, the Kimmo search machinery tries both pairs in subcomputations that have nothing to do with each other. The choice points can potentially build on each other to define a search space that is exponential in the number of independent choice points. This is true regardless of whether the search is carried out depth-first or breadth-first.[2]

For example, return to the artificial Kimmo system that decides Boolean satisfiability for formulas in variables x, y, and z (Figure 2). When the initial y of the formula `yz,x-y-z,-x,-y` is seen, there is nothing to decide between the pairs `y/T` and `y/F`. If the system chooses `y/T` first, the choice will be remembered by the y-consistency automaton, which will enter state 2. Alternatively, if the possibility `y/F` is explored first, the y-consistency automaton will enter state 3. After `yz,x...` has been seen, the x-, y-, and z-consistency automata may be in any of the following state-combinations:

$$\begin{array}{ll} \langle 3,3,2\rangle & \langle 2,3,2\rangle \\ \langle 3,2,3\rangle & \langle 2,2,3\rangle \\ \langle 3,2,2\rangle & \langle 2,2,2\rangle \end{array}$$

(The combinations $\langle 3,3,3\rangle$ and $\langle 2,3,3\rangle$ are not reachable because the disjunction yz that will have been processed rules out both y and z being false, but on a slightly different problem those combinations would be reachable as well.) The search mechanism will consider these possible combinations individually.

Thus, the Kimmo machinery applied to a k-variable SAT problem explores a search space whose elements are k-tuples of truth-values for the variables, represented in the form of k-tuples of automaton states. If there are $k = 3$ variables, the search space distinguishes among $\langle T,T,T\rangle$, $\langle T,T,F\rangle$, and so forth — among 2^k elements in general. Roughly speaking, the Kimmo machinery considers the elements of the search space one at a time, and in the worst case it will enumerate all the elements.

Instead of considering the tuples in this space individually, the constraint-propagation algorithm summarizes whole sets of tuples in slightly imprecise form. For example, the above set of state-combinations would be summarized by the single vector

$$\langle \{2,3\},\{2,3\},\{2,3\}\rangle$$

representing the truth-assignment possibilities

$$\langle x \in \{T,F\}, y \in \{T,F\}, z \in \{T,F\}\rangle.$$

The summary is less precise than the full set of state-tuples about the global constraints among the automata; here, the summary does not indicate that the state-combinations $\langle 3,3,3\rangle$ and $\langle 2,3,3\rangle$ are excluded. The constraint-propagation algorithm never enumerates the set of possibilities covered by its summary, but works with the summary itself.

The imprecision that arises from listing the possible states of each automaton instead of listing the possible combinations of states represents a *decoupling* of the automata. In addition to helping avoid combinatorial blowup, this decoupling allows the state-possibilities for different automata to be adjusted individually. We do not expect that the corresponding imprecision will matter for natural language; instead, we expect that the decoupled automata will individually determine unique states for themselves, a situation in which the summary is precise.[3] For instance,

[2] See Karttunen (1983:184) on the difference in search order between Karttunen's Kimmo algorithms and the equivalent procedures originally presented by Koskenniemi.

[3] Obviously, this can be true in a recognition problem only if the input is morphologically unambiguous, in which case it can still fail to hold if the constraint-propagation method is insufficiently powerful to

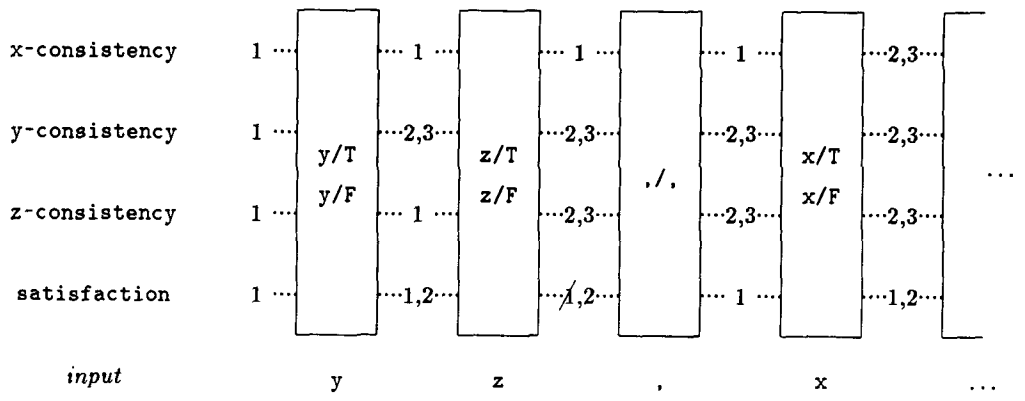

Figure 3: The constraint-propagation algorithm produces this representation when processing the first few characters of the formula yz,x-y-z,-x,-y using the automata from Figure 2. At this point no truth-values have been definitely determined.

in the case of generation involving right-to-left vowel harmony, only the vowel harmony automaton should exhibit nondeterminism, which should be resolved upon processing of the necessary right context. The imprecision also will not matter if two constraints are so independent that their solutions can be freely combined, since the summary will not lose any information in that case.

CONSTRAINT PROPAGATION

Like the Kimmo machinery, the constraint-propagation machinery is concerned with the states of the automata at intercharacter positions. But when nondeterminism makes more than one state-combination possible at some position, the constraint-propagation method summarizes the possibilities and continues instead of trying a single guess. The result is a two-dimensional multi-valued tableau containing one row for each automaton and one column for each intercharacter position in the input.[4] Figure 3 shows the first few columns that are produced in generating from the SAT formula yz,x-y-z,-x,-y. The initial y can be interpreted as either y/T or y/F, and consequently the y-consistency automaton can end up in either state 2 or state 3. Similarly, depending on which pair is chosen, the satisfaction automaton can end up in either state 1 (no true value seen) or state 2 (a true value seen).

In addition to the states of the automata, the tableau contains a *pair set* for each character, initialized to contain all feasible lexical/surface pairs (*cf.* Gajek *et al.*, 1983) that match the input character. As Figure 3 suggests, the pair set is common to all the automata; each pair in the pair set must be acceptable to every automaton. If one automaton has concluded that there cannot be a surface g at the current position, it makes no sense to let another automaton assume there might be one. The automata are therefore not completely decoupled, and effects may propagate to other automata when one automaton eliminates a pair from consideration. Such propagation will occur only if more than one automaton distinguishes among the possible pairs at a given position. For example, an automaton concerned solely with consonants would be unaffected by new information about the identity of a vowel.

Waltz's line-labelling procedure, the best-known early example of a constraint-propagation procedure (*cf.* Winston, 1984), proceeds from an underconstrained initial labelling by eliminating impossible junction labels. A label is impossible if it is incompatible with every possible label at some adjacent junction. The constraint-propagation procedure for Kimmo systems proceeds in much the same way.

rule out invalid possibilities. Note that many cases of morphological ambiguity involve *bracketing* (*e.g.* un[loadable]/[unload]able) rather than the identity of lexical characters. Though the matter is not discussed here, we propose to handle bracketing ambiguity and lexical-string ambiguity by different mechanisms. In addition, for discussions of morphological ambiguity, it becomes very important whether the input representation is phonetic or non-phonetically orthographic.

[4] An extra column is needed at each position where a null might be inserted.

A possible state of an automaton can be eliminated in four ways:

- The only possible predecessor of the state (given the pair set) is ruled out in the previous state set.
- The only possible successor of the state (given the pair set) is ruled out in the next state set.
- Every pair that allows a transition out of the state is eliminated at the rightward character position.
- Every pair that allows a transition into the state is eliminated at the leftward character position.

Similarly, a pair is ruled out whenever any automaton becomes unable to traverse it given the possible starting and ending states for the transition. (There are special rules for the first and last character position. Null characters also require special treatment, which will not be described here.)

The configuration shown in Figure 3 is in need of constraint propagation according to these rules. State 1 of the satisfaction automaton does not accept the comma/comma pair, so state 1 is eliminated from the possible states $\{1,2\}$ of the satisfaction automaton after z. State 1 has therefore been shown as cancelled. However, the elimination of state 1 causes no further effects at this point.

The current implementation simplifies the checking of the elimination conditions by associating sets of *triples* with character positions. Each triple ⟨old state, pair, new state⟩ is a complete description of one transition of a particular automaton. The left, right, and center projections of each triple set must agree with the state sets to the left and right and with the pair set for the position, respectively. Figure 4 shows two of the triple-sets associated with the z-position in Figure 3.

The nondeterminism of Figure 3 is finally resolved when the trivial clauses at the end of the formula `yz,x-y-z,-x,-y` are processed. After x in the clause `-x` all of the consistency automata are noncommittal, *i.e.* can be in either state 2 or state 3. The satisfaction automaton was in state 3 before the x because of the minus sign and it can use either of the triples $\langle 3,x/T,1\rangle$ or $\langle 3,x/F,2\rangle$. However, on the next step it is discovered that only state 2 will allow it to traverse the comma that follows the x. The triple $\langle 3,x/T,1\rangle$ is eliminated and the pair x/T goes with it. The elimination of x/T is propagated to the x-consistency automaton, which loses the triple $\langle 2,x/T,2\rangle$ and can no longer support state 2 in the left and right state sets. The loss of state 2, in turn, propagates leftward on the x-satisfaction line back to the initial occurrence of x. The possibility x/T is eliminated everywhere it occurs along the way. Finally, processing resumes at the right edge.

In similar fashion, the trivial clause `-y` eliminates the possibility y/T throughout the formula. However, this time the effects spread beyond the y-automaton. When the possibility y/T is eliminated from the first pair-set in Figure 3, the satisfaction automaton can no longer support state 2 between the y and z. This leaves $\langle 1,z/T,2\rangle$ as the only active triple for the satisfaction automaton at the second character position. Thus z/F is eliminated and z is forced to truth. When everything settles down, the "easy" formula `yz,x-y-z,-x,-y` has received the satisfying truth-assignment `FT,F-F-T,-F,-F`.

ALGORITHM CHARACTERISTICS

The constraint-propagation algorithm shares with the Waltz labelling procedure a number of characteristics that prevent combinatorial blowup:[5]

- The initial possibilities at each point are limited and non-combinatorial; in this case, the triples at some position for an automaton can do no worse than to encode the whole automaton, and there will usually be only a few triples. It is particularly significant that the number of triples does not grow combinatorially as more automata are added.
- Possibilities are eliminated monotonically, so the limited number of initial possibilities guarantees a limited number of eliminations.
- After initialization, propagation to the neighbors of a visited element takes place only if a possibility is eliminated, so the limited number of eliminations guarantees a limited number of visits.
- Limited effort is required for each propagator visit.

However, we have not done a formal analysis of our implementation, in part because many details are subject to change. It would be desirable to replace the weak notion of monotonic possibility-elimination with some (stronger) notion of indelible construction of representation, based if possible on phonological features. Methods have also been envisioned for reducing the distance that information must be propagated in the algorithm.

The relative decoupling of the automata and the general nature of constraint-propagation methods suggests that a significantly parallel implementation is feasible. However, it is uncertain whether the constraint-propagation method enjoys an advantage on serial machines. It is clear that the Kimmo machinery does combinatorial search while the constraint-propagation machinery does not, but

[5]Throughout this paper, we are ignoring complications related to the possibility of nulls.

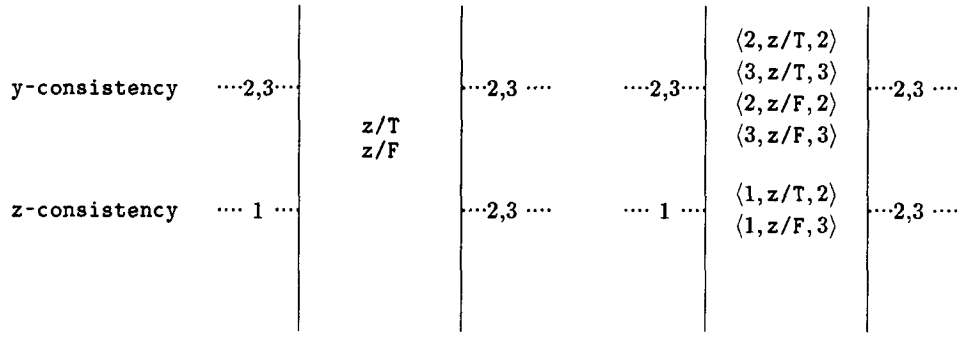

Figure 4: When the active transitions of each automaton are represented by triples, it is easy to enforce the constraints that relate the left and right state-sets and the pair set. The left configuration is excerpted from Figure 3, while the right configuration shows the underlying triples. The set of triples for the y-consistency automaton could easily be represented in more concise form.

we have not investigated such questions as whether an analogue to BIGMACHINE precompilation (Gajek *et al.*, 1983) is possible for the constraint-propagation method. BIGMACHINE precompilation speeds up the Kimmo machinery at a potentially large cost in storage space, though it does not reduce the amount of search.

The constraint-propagation algorithm for generation has been tested with previously constructed Kimmo automata for English, Warlpiri, and Turkish. Preliminary results suggest that the method works. However, we have not been able to test our recognition algorithm with previously constructed automata. The reason is that existing Kimmo automata rely heavily on the dictionary when used for recognition. We do not yet have our Kimmo dictionaries hooked up to the constraint-propagation algorithms, and consequently an attempt at recognition produces meaningless results. For instance, without constraints from the dictionary the machinery may choose to insert suffix-boundary markers + anywhere because the automata do not seriously constrain their occurrence.

Figure 5 shows the columns visited by the algorithm when running the Warlpiri generator on a typical example, in this case a past-tense verb form ('scatter-PAST') taken from Nash (1980:85). The special lexical characters I and <u2> implement a right-to-left vowel assimilation process. The last two occurrences of I surface as u under the influence of <u2>, but the boundary # blocks assimilation of the first two occurrences. Here the propagation of constraints has gone backwards twice, once to resolve each of the two sets of I-characters. The final result is ambiguous because our automata optionally allow underlying hyphens to appear on the surface, in accordance with the way morpheme boundaries are indicated in many articles on Warlpiri.

The generation and recognition algorithms have also been run on mathematical SAT formulas, with the desired result that they can handle "easy" but not "difficult" formulas as described above.[6] For the easy formula $(\overline{y} \vee z)\&(x \vee y)\&\overline{x}$ constraint propagation determines the solution $(\overline{\mathrm{T}} \vee \mathrm{T})\&(\mathrm{F} \vee \mathrm{T})\&\overline{\mathrm{F}}$. But for the hard formula

$$(x \vee y \vee z)\&(\overline{x} \vee \overline{z})\&(\overline{x} \vee z)\&(\overline{y} \vee \overline{z})\&(\overline{y} \vee z)\&(\overline{z} \vee y)$$

constraint propagation produces only the wholly uninformative truth-assignment

$$\begin{aligned}(\{\mathrm{T},\mathrm{F}\} \vee \{\mathrm{T},\mathrm{F}\} \vee \{\mathrm{T},\mathrm{F}\})\&(\overline{\{\mathrm{T},\mathrm{F}\}} \vee \overline{\{\mathrm{T},\mathrm{F}\}}) \\ \&(\overline{\{\mathrm{T},\mathrm{F}\}} \vee \{\mathrm{T},\mathrm{F}\})\&(\overline{\{\mathrm{T},\mathrm{F}\}} \vee \overline{\{\mathrm{T},\mathrm{F}\}}) \\ \&(\overline{\{\mathrm{T},\mathrm{F}\}} \vee \{\mathrm{T},\mathrm{F}\})\&(\overline{\{\mathrm{T},\mathrm{F}\}} \vee \{\mathrm{T},\mathrm{F}\})\end{aligned}$$

Since we believe linguistic problems are likely to be more like the easy problem than the hard one, we believe the constraint-propagation system is an appropriate step toward the goal of developing algorithms that exploit the information structure of linguistic problems.

[6]Note that the current classification of formulas as "easy" is different from polynomial-time satisfiability. In particular, the restricted problem 2SAT can be solved in polynomial time by resolution, but not every 2SAT formula is "easy" in the current sense.

```
0 1 2 3 4 5
  1 2 3 4
    2 3 4 5 6 7 8 9 10 11 12 13
              7 8 9 10 11 12
                8 9 10 11 12 13 14
pIrrI#kIjI-rn<u2>: result ambiguous, pirri{0,-}kuju{-,0}rnu
```

Figure 5: This display shows the columns visited by the constraint-propagation algorithm when the Warlpiri generator is used on the form pIrrI#kIjI-rn<u2> 'scatter-PAST'. Each reversal of direction begins a new line. Leftward movement always begins with a position adjacent to the current position, but it is an accidental property of this example that rightward movement does also. The final result is ambiguous because the automata are written to allow underlying hyphens to appear optionally on the surface.

ACKNOWLEDGEMENTS

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research has been provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-80-C-0505. This research has benefited from guidance and commentary from Bob Berwick.

REFERENCES

Barton, E. (1986). "Computational Complexity in Two-Level Morphology," ACL-86 proceedings (this volume).

Gajek, O., H. Beck, D. Elder, and G. Whittemore (1983). "LISP Implementation [of the KIMMO system]," *Texas Linguistic Forum* 22:187–202.

Halle, M. (1985). "Speculations about the Representation of Words in Memory," in V. Fromkin, ed., *Phonetic Linguistics: Essays in Honor of Peter Ladefoged*, pp. 101–114. New York: Academic Press.

Karttunen, L. (1983). "KIMMO: A Two-Level Morphological Analyzer," *Texas Linguistic Forum* 22:165–186.

Nash, D. (1980). *Topics in Warlpiri Grammar*. Ph.D. thesis, Department of Linguistics and Philosophy, M.I.T., Cambridge, Mass.

Winston, P. (1984). *Artificial Intelligence*, second edition. Reading, Mass.: Addison-Wesley.