SRI International

A NEW CHARACTERIZATION OF ATTACHMENT PREFERENCES

Technical Note 296

March 1983

By: Fernando C. N. Pereira, Computer Scientist

Artificial Intelligence Center
Computer Science and Technology Division

This paper will appear in *Natural Language Processing, Psycholinguistic, Computational, and Theoretical*, Cambridge University Press (1983).

This research was partial supported by the Defense Advanced Research Projects Agency under Contract N00039-80-C-0575 with the Naval Electronic Systems Command. The views and conclusions contained in this document are those of the authors and should not be interpreted as representative of the official policies, either expressed or implied of the Defense Advanced Research Projects Agency or the United States Government.

SRI International®

333 Ravenswood Ave. • Menlo Park, CA 94025
(415) 326-6200 • TWX: 910-373-2046 • Telex: 334-486

A New Characterization of Attachment Preferences[1]

Fernando C. N. Pereira
Artificial Intelligence Center
SRI International
March 1983

Abstract

Several authors have tried to model attachment preferences for structurally ambiguous sentences, which cannot be disambiguated from semantic information. These models lack rigor and have been widely criticized. By starting from a precise choice of parsing model, it is possible to give a simple and rigorous description of Minimal Attachment and Right Association that avoids some of the problems of other models.

1. Introduction

Kimball's parsing principles [8], Frazier and Fodor's Sausage Machine [5, 6], and Wanner's ATN model [13] have tried to explain why certain readings of structurally ambiguous sentences are preferred to others, in the absence of semantic information. The kinds of ambiguity under discussion are exemplified by the following two sentences

(1) Tom said that Bill had taken the cleaning out yesterday
(2) John bought the book for Susan

For sentence (1), the reading in which "yesterday Bill took the cleaning out" is preferred to the one in which "Tom spoke yesterday about Bill taking the cleaning out." Kimball [8] introduced the principle of Right Association (RA) to account for this kind of preferences. The basic idea of the Right Association principle is that, in the absence of other information, phrases are attached to a partial analysis as far right as possible.

For sentence (2), the reading in which "the book was bought for Susan" is preferred to the one

[1]This research was partially supported by the Defense Advanced Research Projects Agency under Contract N00039-80-C-0575 with the Naval Electronic Systems Command. The views and conclusions contained in this document are those of the author and should not be interpreted as representative of the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

in which "John bought a book that had been beforehand destined for Susan." To account for this preference, Frazier and Fodor [5] introduced the principle of Minimal Attachment (MA), which may be summarized as stating that, in the absence of other information, phrases are attached so as to minimize the complexity of the analysis.

Much of the debate about the formulation and interaction of such principles is caused by their lack of precision and, at the same time, by their being too specific. I propose a simple, precise, and general framework in which improved versions of Right Association and Minimal Attachment can be formulated. It will turn out that the two principles correspond to two precise rules on how to choose between alternative parsing actions in the parsing model.

Apart from the concrete results shown, this article has two further purposes. First, to show that it is possible to describe principles such as RA and MA starting from very few assumptions about grammar and parsing mechanisms. Second, that, by making all the assumptions clear, the principles are much easier to formulate and to discuss.

From the material I present, one should not infer that I am proposing certain specific mechanisms as models of human sentence processing. Rather, I am presenting a possible general framework within which precise falsifiable models can be formulated.

Underlying the proposed framework is the assumption of a bottom-up parsing mechanism. This is much less restrictive than it may appear from the literature. Kimball, and Frazier and Fodor, discount purely bottom-up mechanisms on the basis that they are not "predictive" enough. This criticism is founded on the incorrect assumption that bottom-up parsers cannot take account of left context, and in particular of top-down predictions, to narrow their choice of parsing actions. Frazier and Fodor state that [5]

> "...Since higher nodes cannot be entered until after all the words they dominate have been received, these nodes cannot be made use of for the 'forwards' prediction of words or nodes within that portion of the lexical string."

This assumes that the parser has no internal state and its only sources of information are the partially built phrase marker and the input. Of course, this need not be so. For example, the LR ("left-to-right scan, rightmost derivation") context-free parsing theory [2], which we will discuss

in more detail in Section 3, provides a very powerful method to encode possible left contexts into a finite set of parser states, even for ambiguous languages [1]. I am not suggesting that LR parsing is a suitable model for people's performance. I am just pointing out that the world of bottom-up parsers is much wider than is all too often assumed. I will come back to this point later.

2. Shift-Reduce Parsing

The class of bottom-up parsers that I will use to demonstrate the proposals in this paper is the shift-reduce parsers, which are well understood in the theory of context-free parsing [2]. The shift-reduce method is a general framework for bottom-up parsing, which can in fact be used for grammar formalisms other than context-free grammars, such as type-0 grammars [3] and definite-clause grammars [10]. In the present discussion, however, it is enough to look at context-free parsing.

Any shift-reduce method has two basic ingredients: a **stack** and an **oracle**.

The stack is a string of grammar symbols, which is empty when analysis starts and which becomes empty again at the end of a successful analysis. The stack is only accessed at its right end, called the **top** of the stack. The left end is the **bottom**. Symbols can be appended or **pushed** onto the top of the stack, and removed or **popped** from the top of the stack. A string is said to **match** the top of the stack if it is a final segment of the stack. The combination of a current stack and a partially consumed input string is called a **configuration** or **point**. I will write a stack with its top to the right of the page. Empty stacks or strings will be denoted by □.

A shift-reduce parser changes its configuration by doing **moves**. There are two kinds of moves: a single **shift** move and, for each grammar rule, a **reduce** move. The shift move consists in reading the next terminal from the input, and pushing it onto the stack. A reduce move consists in taking a grammar rule whose right-hand side matches the top of the stack, and substituting the matched string on the stack by the left-hand side of the rule.

For example, given the grammar

(1) S → NP VP
(2) NP → Det N
(3) VP → V NP

(4) Det → the
(5) N → cat
(6) N → dog
(7) V → loves

and the input string

the cat loves the dog

the table below shows a sequence of valid moves. The move at the end of each line is applied to the configuration in that line to give the configuration on the next line.

Stack	Remaining input	Move
□	the cat loves the dog	shift
the	cat loves the dog	reduce by (4)
Det	cat loves the dog	shift
Det cat	loves the dog	reduce by (5)
Det N	loves the dog	reduce by (2)
NP	loves the dog	shift
NP loves	the dog	reduce by (7)
NP V	the dog	shift
NP V the	dog	reduce by (4)
NP V Det	dog	shift
NP V Det dog	□	reduce by (6)
NP V Det N	□	reduce by (2)
NP V NP	□	reduce by (3)
NP VP	□	reduce by (1)
S		

A shift-reduce parser is said to reach the **success** state (or succeed), after some sequence of moves on some input, when the stack contains only the initial symbol of the grammar and the input is exhausted. In the previous example, the sequence of moves given leads to the success state. If some sequence of moves cannot lead to the success state and cannot be extended by some new moves, it has **blocked**.

3. Oracles and Conflicts

A shift-reduce parser based only on the preceding definitions would, of course, be impractical, because it would have no means of choosing moves that lead to success on a well formed input. There are just too many possible moves for any sizeable grammar. That is the reason why we need the previously mentioned oracle. An oracle examines the moves that are possible in a

configuration and **forbids** moves that it knows will only lead to blocking move sequences. It **allows** all the other moves. Notice that an oracle is not allowed to forbid moves that may lead to success. Therefore, all the analyses of a given string according to the given grammar correspond to some sequence of allowed moves from the initial configuration.

If oracles were unrestricted, it would be trivial to provide good oracles (just explore all possible move sequences until a successful one is found). Therefore, I will restrict oracles to finite-state machines, and this is assumed from now on. Of course, the existence of suitable oracles is a precondition for accepting shift-reduce parsing as a reasonable model. In general, this is a difficult question, which requires substantial further research.

There are important classes of grammars for which good oracles are possible. I call an oracle **exact** when it allows, at most, one move in any configuration. The context-free languages which can be parsed with exact oracles are called **deterministic** languages [2]. Clearly, deterministic languages must be unambiguous, so they are too small a class of languages for our purposes. Nevertheless, the methods developed for handling deterministic languages, suitably relaxed,[2] may be useful to derive plausible oracles for natural-language analysis. In particular, the LR parser theory, because of its close connection with the general parsing method of Earley [4], is an interesting candidate for investigation. I use LR theory to derive the example of an oracle in the next section.

The connection between LR parsing and the Earley algorithm shows, in an instructive way, how predictions and left context are incorporated in an oracle. The finite automaton that implements the oracle in an LR parser has states corresponding to sets of partially applied rules — called **dotted rules** or **items** — in the Earley parsing algorithm. A dotted rule is just a grammar rule with a dot in the right-hand side separating the symbols to its left, which have been found, from the symbols to its right, which are still needed for the rule to be applied. A state transition in the LR automaton corresponds to moving the dot in the dotted rules for the old state over the symbols that might be on the top of the stack after the parsing action, to get the dotted rules for

---

[2]Relaxation might mean using the theory to produce an imperfect oracle and supplementing it with preference rules (see below).

the new state.

The dotted rules in the Earley algorithm are generated by a combination of top-down predictions and bottom-up reductions. All the dotted rules that are generated for a given input are compatible with an analysis of some string that shares an initial segment with the input. Therefore, the states in the LR oracle are finite encodings of sets of partial parse trees for initial segments of the input. In the case of deterministic languages, this encoding can be exact, but, in the case of ambiguous languages, the finite encoding will collapse incompatible partial parse trees into a single state, thus making the oracle inexact. Furthermore, by restricting the number of states, more incompatible partial analyses would be merged, giving a potential model for the effects of limited memory (see Marcus' theory of deterministic parsing [9]).

Given that any oracle for a natural-language grammar is, in general, going to allow more than one move at each point, when the parser wants to make a move it will have to solve **conflicts**. Two kinds of conflicts can occur: a **shift-reduce** conflict, in which the shift move and some reduce move are allowed at the same point, and a **reduce-reduce** conflict in which no shift but several reduce moves are allowed at the same point. From these definitions, it follows that shift-reduce and reduce-reduce conflicts are mutually exclusive. If a parser is to avoid backtracking or parallel elaboration of analyses, it must resolve conflicts in some way. Of course, by doing so, it may be excluding exactly those moves that lead to success. On the other hand, if backtracking is allowed, conflict resolution may be used to specify an order in which moves are to be tried.

4. Example of an Oracle

To make the notion of an oracle more concrete, I will now give a simple example, consisting of an inexact oracle for the grammar

(1) S → NP VP
(2) NP → Det N
(3) NP → ProperN
(4) VP → V NP
(5) Det → Art
(6) Det → NP 's

As noted before, oracles are finite-state devices. The particular oracle presented here is a

nondeterministic finite automaton. At any configuration during an analysis, the oracle takes its current state and the grammar symbol currently on top of the parser's stack (or □ if the stack is empty) and produces a set of pairs (**parser move, oracle stack**). From this set, the parser chooses a pair which defines its next move and the next state of the oracle. This mapping can be described by **transitions** of the form

state : top of stack ⇒ parser move, next state

where ": **top of stack**" is omitted if the transition applies for any symbol on the top of the stack.

The example oracle is given by the following transitions, where A is the initial state:

```
  A ⇒ shift, B
  B : Art ⇒ reduce by (5), C
  B : ProperN ⇒ reduce by (3), D
  C ⇒ shift, E
† D ⇒ shift, F
† D ⇒ reduce by (4), G
  E ⇒ reduce by (2), D
  F : 's ⇒ reduce by (6), C
  F : V ⇒ shift, B
  G ⇒ reduce by (1), success
```

In constructing this oracle, I have used the methods outlined in Section 3. Each state corresponds to a set of dotted rules, and state transitions correspond to shift or reduce moves that advance the dot in the dotted rules in a state. More precisely, the states are the "LR(0) sets of items" [2] for the grammar.

This oracle is exact but for a shift-reduce conflict in state D, from which there are the two alternative transitions marked with †. This conflict can be solved by adding lookahead to the oracle, and, in fact, the grammar can be parsed deterministically with one symbol of lookahead. A suitable exact oracle can be derived using a specialization of the LR techniques known as LALR(1) parsing [1, 12].

I will now give an example of the oracle's operation with the input sentence
The boy stole Mary's cat.

State	Stack	Move	Remaining input

	A	□	shift	the boy stole Mary's cat
	B	Art	reduce by (5)	boy stole Mary's cat
	C	Det	shift	boy stole Mary's cat
	E	Det N	reduce by (2)	stole Mary's cat
†	D	NP	shift	stole Mary's cat
	F	NP V	shift	Mary's cat
	B	NP V ProperN	reduce by (3)	's cat
†	D	NP V NP	shift	's cat
	F	NP V NP 's	reduce by (4)	cat
	C	NP V Det	shift	cat
	E	NP V Det N	reduce by (2)	□
	D	NP V NP	reduce by (4)	□
	G	NP VP	reduce by (1)	□
	H	S		

At the points in the analysis marked with †, the oracle allows a shift or a reduction by rule (4). The parser has to choose between (or develop in parallel) those moves. In the example, shift moves are preferred. This is the right choice for a nonblocking sequence of moves (I am here behaving as an additional oracle.)

5. Preference

Precise versions of the Right Association and Minimal Attachment strategies can now be presented.

- RA corresponds to solving all shift-reduce conflicts in favor of shifting;
- MA corresponds to solving all reduce-reduce conflicts in favor of the reduce move that pops most symbols from the stack[3].

To see how these definitions work, I assume that the oracle will produce exactly the conflicts that correspond to the intuitively perceived ambiguities in the examples. To avoid uninteresting detail, I will also assume the context-free rules necessary to analyze the relevant part of each example, and I will enter words in the stack as their lexical categories. The reader should not assume that the particular rules chosen have any particular merit apart from reflecting

[3]What about several allowed productions popping the same amount? This means that there are several rules with the same right-hand sides, undistinguishable by the oracle. Apart from terminal rules (those with only terminals in the right-hand side) for homographs, such rule conflicts do not seem to occur in natural-language grammars. In particular, if an X-bar grammar is used, right-hand sides will at least differ in their head words.

reasonably the intuitive structure of the examples[4].

I will first discuss the following shift-reduce conflict:

(FF1-1) Tom said that Bill had taken the cleaning out yesterday.

(1) S → NP VP
(2) VP → V Sbar
(3) S → S Adv
(4) Sbar → that S

Stack: NP V that S

When the input has been read up to the ^, after some necessary reductions that have to be done at this point (the oracle must see to that), the parser will have the choice of reducing by rule (4) or of shifting the Adv "yesterday" onto the stack. The preference for shifting will lead to the configuration

Tom said that Bill had taken the cleaning out yesterday.

Stack: NP V that S Adv

At this point, only rule (3) may be used, and the adverb is therefore attached to the lower S.

A reduce-reduce conflict occurs in (FF1-13)[5]:

(FF1-13) John bought the book for Susan.

(5) NP → NP PP
(6) VP → V NP (PP)

Stack: NP V NP PP

At this point in the analysis, the parser has to reduce either by rule (5) (and then rule (6) without the optional PP) or by rule (6) with the PP. The latter option consumes more from the stack, and therefore is chosen, causing the PP to be attached to the VP node.

Rules (5) and (6) above are implicit both in Frazier and Fodor's and in Wanner's analyses for the sentence. Unfortunately, my view of MA is sensitive to the choice of grammar rules in the

---

[4]All the examples are taken from Frazier and Fodor [5, 6] and Wanner [13]. To simplify the references, they will be numbered by (A-N), where A is FF1 for [5], FF2 for [6] and ATN for Wanner's article, and N is the number within the article.

[5]A nonterminal in parenthesis is optional.

same way as Wanner's. If rule (5) was substituted by (5′)[6]

(5′) NP → Det N PP*

the preference for shifting over reducing would cause the final PP to be attached to the NP[7]. Of course, my proposal still explains the data, but only together with certain grammar rules. And it might be argued that rule (5′) is highly dubious, because it suggests that a NP node is only found when all the PPs that modify it have been found. This runs against the intuition that, at each stage through the string of PPs, one has a perfectly well formed NP, which is the situation portrayed by rule (5).

The same problem would occur in Wanner's ATN, if he had used a right-recursive (or a loop) network to attach the PPs, instead of a left-recursive network analogous to rule (5). But whereas shift-reduce parsing has no problem with left-recursive constructions, the top-down regime implicit in Wanner's proposal is inadequate for the NP part of his network. In a phrase with the final analysis of Figure 1,

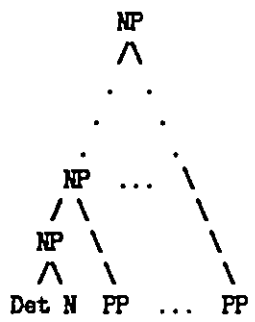

Figure 1: Left Recursive Analysis

the ATN will attach every intermediate NP node as if it were the top NP, before finding that there are more PPs to include in the NP and that, therefore, the intermediate NP has been wrongly attached. This is very unreasonable behavior for a parser that purports to model human performance.

Wanner criticizes Frazier and Fodor's use of sentence length to explain the preferred reading of

---

[6]The notation X^* denotes any string of zero or more Xs.

[7]I am indebted to David Warren for pointing this out.

(FF1-15).

(FF1-15) Joe bought the book that I had been trying to obtain
for Susan.
^

Like Wanner's, my proposals make that preference independent of sentence length.

(7) NP → NP Sbar/NP
(8) VP → V to VP
(9) PP → Prep NP

(4a) Sbar/NP → Comp S/NP
(1a) S/NP → VP
(1b) S/NP → NP VP/NP
(6a) VP/NP → V (PP)
(8a) VP/NP → V to VP/NP
Stack: NP V NP Comp NP V to V

The rules use X/Y nonterminals to represent "an X with a Y hole". This is an approximation, sufficient for the purposes of this paper, of Gazdar's [7] "derived category" description of relative clauses. In (FF1-15), there is a noun phrase gap for the object of "obtain" in "obtain—for Susan", which must thus be analyzed as a VP/NP, a verb phrase missing a noun phrase.

In the given configuration, there is a shift-reduce conflict, where the parser may either reduce by rule (6a) (without the optional PP) or shift the word "for". After the shift, there is only one possible sequence of moves, sketched below:

Stack	Remaining input	Move
NP V NP Comp NP V to V Prep	Susan	shift
NP V NP Comp NP V to V Prep NP	□	reduce by (9)
NP V NP Comp NP V to V PP		reduce by (6a)
NP V NP Comp NP V to VP/NP		reduce by (8a)
NP V NP Comp NP VP/NP		...

We see thus that the PP "for Susan" is attached to the lower VP, giving the preferred reading in which "it was for Susan that I was trying to obtain the book," rather than the reading in which "John bought for Susan the book I had been trying to obtain."

Although Wanner's model can explain this example, it cannot, as noted in [6], explain the RA-induced preference in

(FF2-27) Joe took the book that I had wanted to include in my
birthday gift for Susan

The problem with Wanner's proposal is that his top-down model is forced to apply his preference

rules *before* the phrases being attached have been scanned. Wanner's rule for enforcing MA, the CAT-before-SEEK rule in which looking for a word takes precedence over looking for a complex phrase, operates before the parser has seen what follows "a birthday gift." The original proposals of Frazier and Fodor cannot explain (FF2-27) either, and so they are forced to go into a rather complicated explanation of a new principle of "local association."

In contrast, (FF2-27) causes no problem to the present formulation. The preposition "for" is reached with the following configuration

Stack: NP V NP Comp NP V to V Prep NP Input: for Susan

At this point, there will be a shift-reduce conflict between shifting "for" onto the stack or reducing Prep NP to a PP. The shift move is preferred, leading to the following sequence of moves

Stack	Input	Move
NP V NP Comp NP V to V Prep NP Prep	Susan	shift
NP V NP Comp NP V to V Prep NP Prep NP		reduce by (9)
NP V NP Comp NP V to V Prep NP PP		reduce by (5)
NP V NP Comp NP V to V Prep NP		reduce by (9)
NP V NP Comp NP V to V PP		reduce by (6a)
NP V NP Comp NP V to VP/NP		reduce by (8a)
NP V NP Comp NP VP/NP		reduce by (1b)
NP V NP Comp S/NP		reduce by (4a)
NP V NP Sbar/NP		reduce by (7)
NP V NP		reduce by (6)
NP VP		reduce by (1)
S		success

This example shows that, in the shift-reduce model, conflicts between Minimal Attachment and Right Association are automatically eliminated because the two principles operate necessarily at different points in the analysis. This is a consequence of general properties of bottom-up parsers that will be discussed in the next section.

Another case of reduce-reduce conflict happens in example (FF2-12). The initial segment given there can be extended in two different ways that force different lexical category assignments for "that." The preferred reading seems to be the one used in "That silly old-fashioned hat is cheap", in which the fragment is the initial segment of a noun phrase, in contrast with the reading in "That silly old fashioned hats are cheap is well known," for which "that" is a complementizer. Frazier and Fodor argue that Wanner's CAT-before-SEEK principle for ATNs cannot explain the preferred reading because the example does not involve a conflict between

looking for a word and looking for a phrase, but a conflict between two different phrase types, which could be called a SEEK-SEEK conflict in ATN terminology[8]. Choosing the largest reduction again fits the data.

(FF2-12) That silly old-fashioned

(10) NP → (Det) Adj* N

Stack: that Adj Adj (N)

Reducing by rule (10) with the optional Det will pop the most from the stack, so that will be the preferred move. I am assuming here that a word is only given a lexical category when it is needed for a reduction, otherwise the choice between Det or Comp for "that" would have to be done immediately after shifting "that" onto the stack. Delaying the choice of lexical category seems reasonable, and fits the general notion that bottom-up parsing avoids making decisions too early.

6. In Defence of Bottom-Up Parsers

The examples of the last section show the crucial difference between top-down and bottom-up models of sentence parsing. Although there is a superficial similarity between arc preferences in an ATN and shift-reduce conflict resolution, the two mechanisms operate in entirely different ways: whereas in the top-down ATN model preferences are exercised before the phrases to which they apply are scanned, in the shift-reduce formulation preferences operate only when phrases could be closed by a reduce move. Because of this, situations that appear as a conflict between two preference principles in the ATN model do not arise in a shift-reduce parser: the conflict has disappeared by the time the parser has parsed the relevant phrases.

I have also shown in the last section that two important preference principles are naturally described as conflict-resolution rules in a very general bottom-up parsing framework[9]. The effects of these conflict resolution rules are only indirectly dependent on the actual grammar used, and, in fact, the rules I used are consistent with the phrase markers used in the discussion of the

---

[8] A SEEK action is the top-down ATN counterpart of a reduction in a bottom-up, shift-reduce, parser.

[9] The resolution of shift-reduce conflicts in favor of shifting is actually used when parsing ambiguous programming language constructs, such as dangling 'else' statements in Algol-like languages, to achieve the analysis users find most natural [1].

examples by Frazier and Fodor and Wanner.

As I have explained in Section 3, the criticism that bottom-up parsers have no predictive capabilities is unfounded. The predictive ability of a shift-reduce parser is embodied in its oracle, which will take note of the left context in its internal finite-state network. This finite-state network encodes that "after such and such phrases have been found, and given that the next few words are such and such, we are building such and such phrase, and therefore such and such shifts or reduces are required." In fact, shift-reduce parsing as a model of performance might be criticized not because it can predict too little, but because it can predict too much: the class of languages that can be parsed bottom-up without backtracking[10] is strictly larger than the class of languages parsable top-down without backtracking [2].

The fact that an oracle tries to encode, in a finite (and necessarily small) number of states, an infinity of situations means that there may be some loss of information. That is, when deep down in the analysis of some subphrase, the detail of the higher found or predicted nodes of the left context may be lost. This is precisely the effect of limited memory that Frazier and Fodor try to model in the Sausage Machine.

Shift-reduce parsers require more stack for deep right branching constructions than for deep left branching ones. This has been used by Kimball to discount them as useful in modeling performance. However, his argument depends crucially on what grammar rules are chosen, whereas the arguments here are of a more general nature and less dependent on the grammar used.

7. Conclusion

I have explained how it is possible to describe rigorously the attachment preferences of Right Association and Minimal Attachment in a way that seems to satisfy both Frazier and Fodor's and Wanner's requirements, and that clarifies the interaction between the two principles. I have started from very limited assumptions that clearly distinguish between parsing strategy (the

[10] Or parallel elaboration.

oracle), grammar, and preferences. In contrast, Frazier and Fodor's SM mixes strategy and preferences, and Wanner's ATN mixes preferences with grammar formulation.

My formulation of the principles, however, is not totally independent of the grammar rules. This dependence on the grammar might be caused by the shift-reduce strategy making decisions too early. It would be interesting to investigate extending the idea of delayed assignment of lexical categories used in example (FF2-12) into a method for delaying certain reductions. A careful discussion of the role of delayed reductions in handling lexical preferences in a shift-reduce parsing model has been given by Shieber [11]. His model gives the correct predictions for some cases in which the mechanisms presented here are too coarse.

Acknowledgments

I would like to thank the following people for their comments and advice: David Warren, Henry Thompson, Stu Shieber, Lauri Karttunen, Janet Fodor, Jane Robinson and Barbara Grosz. In no way do I want to suggest that they agree with all the contents of the paper, whose views (and mistakes!) are my own.

References

[1] A. V. Aho and S. C. Johnson, "LR Parsing," *Computing Surveys*, Vol. 6, No. 2, pp. 99-124 (1974).

[2] A. V. Aho and J. D. Ullman, *The Theory of Parsing, Translation and Compiling*, (Prentice-Hall, Englewood Cliffs, New Jersey, 1972).

[3] P. Deussen, "A Unified Approach to the Generation and the Acception of Formal Languages," *Acta Informatica*, Vol. 8, pp. 377-390 (1978).

[4] J. Earley, "An Efficient Context-Free Parsing Algorithm," *Communications of the ACM*, Vol. 13, No. 2, pp. 94-102 (1970).

[5] L. Frazier and J. D. Fodor, "The Sausage Machine: A New Two-Stage Parsing Model," *Cognition*, Vol. 6, pp. 291-325 (1978).

[6] J. D. Fodor and L. Frazier, "Is the Human Sentence Parsing Mechanism an ATN?," *Cognition*, Vol. 8, pp. 417-459 (1980).

[7] G. Gazdar, "Unbounded Dependencies and Coordinate Structure," *Linguistic Inquiry*, Vol. 12, No. 2, pp. 155-184 (1981).

[8] J. Kimball, "Seven Principles of Surface Structure Parsing in Natural Language," *Cognition*, Vol. 2, No. 1, pp. 15-47 (1973).

[9] M. Marcus, *A Theory of Syntactic Recognition for Natural Language*, (MIT Press, Cambridge, Massachussets, 1980).

[10] F. C. N. Pereira and D. H. D. Warren, "Definite Clause Grammars for Language Analysis - a Survey of the Formalism and a Comparison with Augmented Transition Networks," *Artificial Intelligence*, Vol. 13, pp. 231-278 (1980).

[11] S. Shieber, A Method for Disambiguating Sentences. Forthcoming.

[12] S. Shieber, Shift-Reduce Scheduling and Syntactic Closure. Forthcoming.

[13] E. Wanner, "The ATN and the Sausage Machine: Which One Is Baloney?," *Cognition*, Vol. 8, pp. 209-225 (1980).