A BREADTH-FIRST PARSING *MODEL*

John Bear

Linguistics Research Center
University of Texas
Austin, Texas 78712 U.S.A.

ABSTRACT

Recent attempts at modeling humans' abilities at processing natural language have centered around depth first parsing algorithms, and control strategies for making the best choices for disambiguation and attachment. This paper proposes a breadth-first algorithm as a model. The algorithm avoids some of the common pitfalls of depth-first approaches regarding ambiguity, and by using more pre-ccmputed information about the grammar, avoids same of the usual problems of parallel parsing algorithms as well.

1. Parsing Models

In the study of computational models of human language processing, cognitive scientists seem to have given little attention to all-paths parsers, focusing instead on depth-first algorithms. This restriction is imposed so that the models will be consistent with the fact that people do not generally perceive ambiguities. In addition, it is an attempt to stay in line with the hypothesis that people parse sentences in linear time. The idea is that the fewer alternatives considered, the faster the parse time should be. The fastest way, of course, would be a depth first parse which made the right choice at every step of the way, hence the interest in deterministic parsers. The attempt to find principles which would guide a parser correctly through a depth-first search, led (Kimball, 1973) to formulate the principles of Right Association and Closure. (Frazier and Fodor, 1978) propose their own principles: Minimal Attachment and Local Association. In addition (Church, 1980) proposes the A-over-A Early Closure Principle, and (Ford, Bresnan, and Kaplan, 1981) propose the principles of Lexical Preference and Final Arguments. All of these principles try to account for how a top-down depth-first parser could get the preferred readings of sentences. However, the point I would like to make here is that for each choice point in an ambiguous example, there is a second alternative which the parser needs to be able to get at least some of the time. As (Crain and Steedman, 1981) point out, the fact that people generally only perceive one reading of a sentence is perfectly consistent with a parsing model which finds all the possible syntactic ramifications from looking at a word, does some contextual filtering to decide which alternative(s) to keep, and then looks at a new word and repeats. Given this, it is not obvious that the breadth-first approach is inferior.

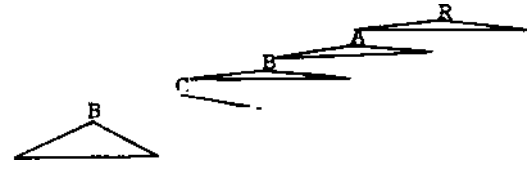
2. A Breadth First Parser

Since (Earley's, 1970) and (Pratt's, 1975) demonstrations of how a parser can work both bottom-up, and top-down, there have been several proposals for how this information might be used to good effect in a psychological model. (Chester, 1980) proposes a depth-first left-corner parser which uses top-down information. (Martin, Church, and Patil, 1981) propose an all paths parser very similar to the one presented here. Both of the parsers just mentioned fit roughly into the framework presented in (Kay, 1980) , which allows for intelligent left-corner parsers which can be bottom-up or top-down, or anything in between. The basic idea behind these parsers is that they always have access to two kinds of information: what categories may come next (top-down), and what word actually is next (bottom-up). The modification suggested here is the incorporation of two look-ahead buffers, and the abolishment of all inactive edges.

2.1. Reachability

We start with a discussion of what (Pratt, 1975) calls "precedence", and (Kay 1980) calls "reachability". A category A is reachable from a category R iff there is a derivation tree of R which has A on its left branch. In addition, a left corner of a rule [X —> Y_1, Y_2 ... Y_n] is the first element on the right side of the arrow, Y_1. In the example below, A, B, and C are all reachable from R.

1.

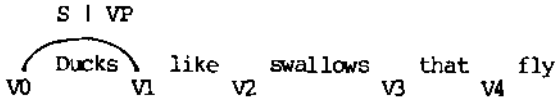


2.2. Basic Algorithm

The parser uses a chart as in (Kay, 1980). The junctures between the words are called vertices and are labelled with numbers.

The parser stores information on the vertices in the form of edges. These edges represent partially completed constituents. In Kay's terms, they would be "active". Each edge contains a category, a completion, and a start vertex. The category tells what the constituent will build up to when the rest of the daughters have been found. The completion, to use a term from (Winograd, 1983), indicates which of the daughters still need to be found. The start vertex shows the left end of the partial constituent. For instance consider the example below.

2.



Given that "ducks" can build to an NP, and that there is a rule [S —> NP VP] in the grammar, the example shows that an edge may be stored at VI going back to V0, and representing an S missing a VP.

The algorithm uses three main functions, Parse, Extend-Edges, and New-Constit. The function Parse looks at the words in the input string one word at a time, considering all the ramifications from one word before going on to the next. It keeps track of where it is with the Current Vertex, the vertex immediately to the right of the newest word looked at. The Previous Vertex is the vertex immediately before it. For each word, Parse calls New-Constit.

Informally, for words and for each new constituent built, the procedure New-Constit does three things. 1) It builds edges which are incomplete but which have just found their first daughter. 2) It tries to add the new-constituent as a daughter to other incomplete constituents to its left via the function Ext end-Edges. And 3) for unit productions which can build up the constituent, the procedure New-Constit calls itself recursively.

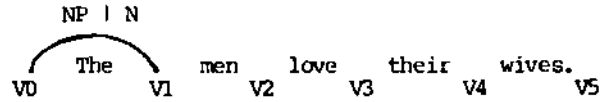
The function Extend-Edges combines an incomplete constituent with a completed constituent to its right. If the result is a completed constituent, it calls New-Constit. If the new constituent still lacks one or more daughters, a new edge is created and stored at the vertex just beyond the word the parser has most recently scanned. An edge is always stored at the vertex representing its right end. At the time it is stored this is always the Current Vertex. Newly completed constituents always end at that Vertex too.

When a new constituent is built that spans the whole string, it is put into a list of interpretations, not on the chart. Only incomplete constituents are stored on the chart.

2.3. Top-down Filtering

To incorporate top-down information, the vertices also need to keep track of which symbols could come next, based on the edges stored at a given vertex, and the precomputed information about reachability. Given a constituent going from V_i to V_j, and a list of edges stored at V_i, we can restrict the construction of constituents and edges even further. Now an edge is only proposed for a rule, if the root of the rule is reachable from a category which is first on the completion of one of the edges at V_i. Similarly, a constituent starting at V_i is only built by some unit production if the root of the rule is reachable from a category which is first on the completion of one of the edges at V_i.

3.



At least some of the time, [men] can build to an NP. However in the example in (3), a constituent starting from VI nay only be built if it will build to an N, or to something which is reachable from N. Since an NP is presumably not reachable from N, the NP [men] does not get built.

So far, the procedure outlined fits within the schemata that (Kay, 1980) describes. The next modification goes a little further. In addition to the machinery already described, two one-word look-ahead buffers are added. Now the word to the right of the Current Vertex is in the W-buffer (Word-buffer). The other buffer, the M-buffer (Meaning-buffer) is initially empty for each word, and is a receptacle for information about a word's syntactic category or categories.

Now before the parser decides that it can build a new edge or extend an old one, it checks the completion against the word in the W-buffer. If the word is not reachable from what would be the first element of the new completion, the edge does not get constructed. If the word in the buffer is syntactically ambiguous, i.e. has more than one possible syntactic category, then the category which allows the edge to be constructed is saved in the M-buffer. So by the time the parser is ready to advance, it has already narrowed down the lexical ambiguity somewhat. If there is still more than one possible category for the word by the time the parser gets to it, the new word in the look-ahead buffer could conceivably filter out all the edges which would be proposed for one or more of the undesirable readings. If such is not the case, then the parser must resort to some sort of contextual filtering as alluded to above.

The fact that the parser is breadth-first allows the parser to dispense with inactive edges. This turns out to have an interesting effect. Although the parser may build constituents that it cannot use in the final parse tree, it does not keep them around. For instance in the sentence, "Bill likes the woman who jogs," the parser will build up the VP [likes the woman] and then combine it with [Bill] to form a sentence. If there are no rules in the grammar of the form, [X —> S Y], though, nothing more happens to the S. It simply never gets saved.

3. Conclusion

The fact that regular languages may be parsed in linear time is due to the fact that for every nondeterministic finite state machine, there is an equivalent deterministic one. Or put in terms of grammars, for every regular language there is an unambiguous linear grammar that generates it. The problem with natural language, though, is that it has ambiguities. One may point out the need, in any model, to resort to contextual information to choose between different alternatives when ambiguity is encountered. To do this however, requires that the different choices exist in the model, i.e., in terms of finite state machines it requires that the states of the machine have not been expanded to get rid of the nondeterminism. In terms of grammars again, even though there exists a linear grammar for any regular language, it is certainly not true that all grammars of regular languages are linear. Hence, the claim that natural language is regular cannot really be said to account for the fact that people seem to be able to parse natural languages in linear time. One needs to make the stronger claim that linear grammars can adequatly describe natural language. This precludes the existence of any ambiguity at all, and seems to be excessive considering the facts.

As an alternative to the deterministic parsers that have been proposed, we have suggested a breadth-first parser for context-free languages. It only pursues alternatives which are consistent with information about what has come before, and with what the next word is. If the grammar taken as a whole specifies that there is only one alternative at some point, even though locally there might be more than one, then the parser only pursues that one alternative. If there is global ambiguity, the parser allows for it. To account for the observation that ambiguity is usually not perceived, the parser only needs to have access to the same sort of contextual information that depth-first parsers need.

4. REFERENCES

[1] Kimball, John (1973) Seven Principles of Surface Structure Parsing in Natural Language. Cognition, Vol 2, pp. 15-47.

[2] Frazier, Lyn and Janet Fodor (1979) The Sausage Machine: A New Two-Stage Parsing Model. Cognition 6, pp.291-325.

[3] Church, Kenneth (1980) On Memory Limitations in Natural Language Processing. MIT Laboratory for Computer Science: Cambridge, Mass.

[4] Ford, Marilyn, Joan Bresnan, and Ronald Kaplan (1981) A Competence-Based Theory of Syntactic Closure. Paper presented at the Symposium on Modelling Human Parsing Strategies. Center for Cognitive Science, University of Texas, Austin.

[5] Crain, Steven and Mark Steedman (1981) The Use of Context by the Psychological Parser. Paper presented at the Symposium on Modelling Human Parsing Strategies. Center for Cognitive Science, University of Texas, Austin.

[6] Earley, Jay (1970) An Efficient Context-Free Parsing Algorithm. Communications of the ACM, Vol 13, No 2.

[7] Pratt, Vaughn (1975) Lingol-A progress report. UCAI 4.

[8] Chester, Daniel (1980) A Parsing Algorithm that Extends Phrases. American Journal of Computational Linguistics, Vol 6, No 2.

[9] Martin, W.A., Church, K., and Ramesh Patil (1981) Preliminary Analysis of a Breadth-First Parsing Algorithm: Theoretical and Experimental Results. Paper presented at the Symposium on Modelling Human Parsing Strategies. Center for Cognitive Science, University of Texas, Austin.

[10] Kay, Martin (1980) Algorithm Schemata and Data Structures in Syntactic Processing. To appear in The Proceedings of the Nobel Symposium on Text Processing, Gothenburg, sweden.

[11] Winograd, Terry (1983) Language as a Cognitive Process, Vol 1. Addison-Wesley: Reading, Mass.