TECHNICAL CORRESPONDENCE

PARSING DISCONTINUOUS CONSTITUENTS IN DEPENDENCY GRAMMAR

Discontinuous constituents—for example, a noun and its modifying adjective separated by words unrelated to them—are common in variable-word-order languages; Figure 1 shows examples. But phrase structure grammars, including ID/LP grammars, require each constituent to be a contiguous series of words. Insofar as standard parsing algorithms are based on phrase structure rules, they are inadequate for parsing such languages.[1]

The algorithm presented here, however, does not require constituents to be continuous, but merely prefers them so. It can therefore parse languages in which conventional parsing techniques do not work. At the same time, because of its preference for nearby attachments, it prefers to make constituents continuous when more than one analysis is possible. The new algorithm has been used successfully to parse Russian and Latin (Covington 1988, 1990).

This algorithm uses dependency grammar. That is, instead of breaking the sentence into phrases and subphrases, it establishes links between individual words. Each link connects a word (the "head") with one of its "dependents" (an argument or modifier). Figure 2 shows how this works. The arrows point from head to dependent; a head can have many dependents, but each dependent can have only one head. Of course the same word can be the head in one link and the dependent in another.[2]

Dependency grammar is equivalent to an X-bar theory with only one phrasal bar level (Figure 3)—the dependents of a word are the heads of its sisters. Thus dependency grammar captures the increasingly recognized importance of headship in syntax. At the same time, the absence of phrasal nodes from the dependency representation streamlines the search process during parsing.

The parser presupposes a grammar that specifies which words can depend on which. In the prototype, the grammar consists of unification-based dependency rules (called D-rules) such as:

$$\begin{bmatrix} \textit{category:noun} \\ \textit{person:X} \\ \textit{number:Y} \\ \textit{case:nominative} \end{bmatrix} \leftarrow \begin{bmatrix} \textit{category:verb} \\ \textit{person:X} \\ \textit{number:Y} \end{bmatrix}$$

This rule sanctions a dependency relation between any two words whose features unify with the structures shown—in this case, the verb and its subject in a language such as Russian or Latin. The arrow means "can depend on" and the word order is not specified. X and Y are variables. D-rules take the place of the phrase structure rules used by Shieber (1986) and others; semantic information can easily be added to them, and the whole power of unification-based grammar is available.

The parser accepts words from the input string and keeps track of whether or not each word is "independent" (not yet known to depend on another word), indicated by + or − in Figure 4. On accepting a word W, the parser does the following:

(1) Search the independent words (those marked +), most recent first, for words that can depend on W. If any are found, establish the dependencies and change the marking of the dependents from + to −.

(2) Search all words so far seen, most recent first, for a word on which W can depend. If one is found, establish the dependency and mark W as −. Otherwise mark W as +.

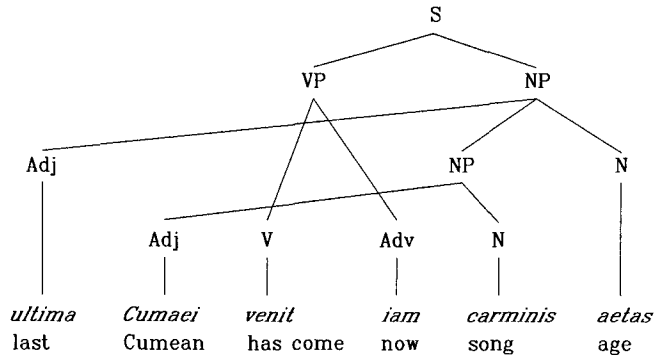
Figure 4 shows the process in action. The first three words, *ultima Cumaei venit,* are accepted without creating any links. Then the parser accepts *iam* and makes it depend on *venit*. Next the parser accepts *carminis,* on which *Cumaei,* already in the list, depends. Finally it accepts *aetas,* which becomes a dependent of *venit* and the head of *ultima* and *carminis.*

The most-recent-first search order gives the parser its preference for continuous constituents. The search order is significant because it is assumed that the parser can backtrack, i.e., whenever there are alternatives it can back up and try them. This is necessary to avoid "garden paths" such as taking *animalia* (ambiguously nominative or accusative) to be the subject of *animalia vident pueri* "boys see animals."

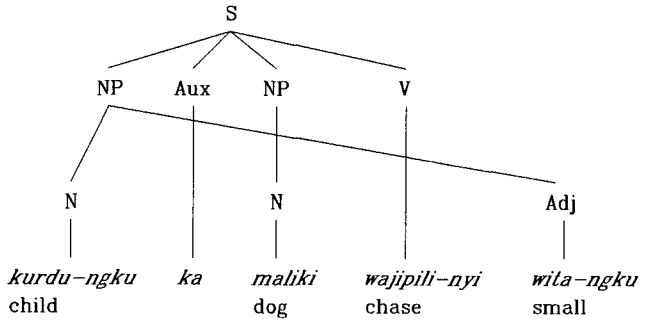
With ordinary sentences, however, backtracking is relatively seldom necessary. Further, there appear to be other constraints on variable word order. Ades and Steedman (1982) propose that all discontinuities can be resolved by a pushdown stack. (For example, pick up *ultima,* then *Cumaei,* then put down *Cumaei* next to *carminis,* then put down *ultima* next to *aetas.* Crossing movements are not permitted.) Moreover, there appears to be an absolute constraint against mixing clauses together.[3] If these hypotheses hold true, the parser can be modified to restrict the search process accordingly.

Most dependency parsers have followed a "principle of adjacency" that requires every word plus all its direct and indirect dependents to form a contiguous substring (Hays and Ziehe 1960; Starosta and Nomura 1986; Fraser 1989; but not Hellwig 1986 and possibly not Jäppinen et al. 1986). This is equivalent to requiring constituents to be continuous. This parser imposes no such requirement. To add the adjacency requirement, one would modify it as follows:

(1) When looking for potential dependents of W, never



'The last era of the Cumean song has now arrived'
(Latin; Vergil, Eclogues IV.4)



'The small child is chasing the dog'
(Warlpiri; Siewierska 1988:158, citing Nash)

Figure 1. Examples of discontinuous constituents.

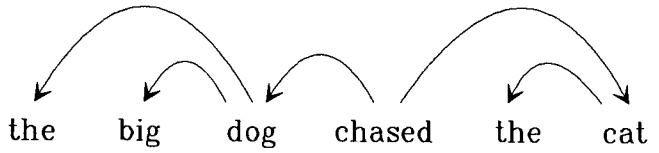


Figure 2. Dependency representation of a sentence. Arrows point from each word to its dependents (modifiers or arguments).

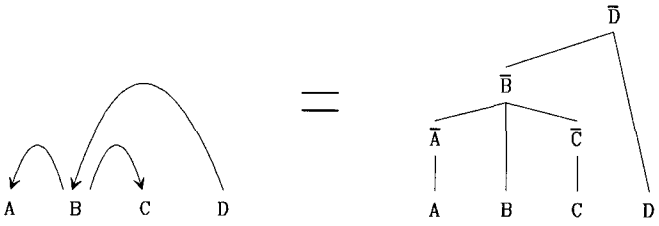


Figure 3. Equivalence of dependency network to X-bar tree.

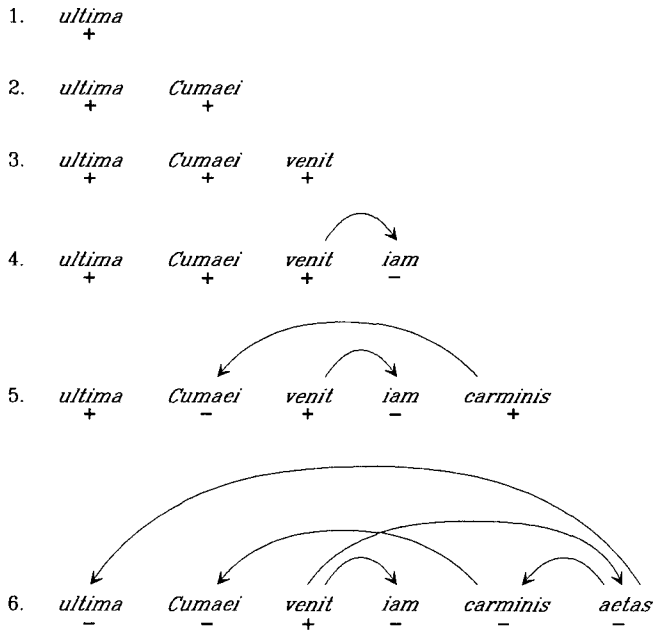


Figure 4. The parser accepts words one by one and tries to link them together; '+' marks words that do not (yet) depend on other words.

skip over an independent word. That is, if an independent word is found that cannot depend on *W*, then neither can any earlier independent word.

(2) When looking for the word on which *W* depends, consider only the previous word, that word's head, the head's head if any, and so on.

With these requirements added, the algorithm would be the same as one implemented by Hudson (1989).

Formal complexity analysis has not been carried out, but my algorithm is simpler, at least conceptually, than the variable-word-order parsers of Johnson (1985), Kashket (1986), and Abramson and Dahl (1989). Johnson's parser and Abramson and Dahl's parser use constituency trees with explicit discontinuity ("tangled trees"), with all their inherent unwieldiness. Kashket's parser, though based on GB theory, is effectively a dependency parser since it relies on case assignment and subcategorization rather than tree structure.

Michael A. Covington
Artificial Intelligence Programs
The University of Georgia
Athens, GA 30602

References

Abramson, Harvey and Dahl, Veronica. (1989). *Logic Grammars.* Springer.

Ades, Anthony E. and Steedman, Mark J. (1982). "On the order of words." *Linguistics and Philosophy,* 4: 517–558.

Covington, Michael A. (1990). "A dependency parser for variable-word-order languages." In *Computer Assisted Analysis and Modeling on the IBM 3090,* edited by Hilton U. Brown, MIT Press.

Covington, Michael A. (1988). "Parsing variable-word-order languages with unification-based dependency grammar." Research report 01-0022, Artificial Intelligence Programs, The University of Georgia.

Fraser, Norman M. (1989). "Parsing and dependency grammar." *UCL Working Papers in Linguistics,* 1: 296–319.

Hays, David G. (1964). "Dependency theory: a formalism and some observations." *Language* 40: 511–525.

Hays, David G. and Ziehe, T. W. (1960). "Studies in machine translation, 10—Russian sentence-structure determination." Research memorandum RM-2358, The RAND Corporation, Santa Monica, CA.

Hellwig, Peter. (1986). "Dependency unification grammar." In *Proceedings of the 11th International Conference on Computational Linguistics* (COLING-86). 195–198.

Hudson, Richard. (1989). "Towards a computer-testable word grammar of English." *UCL Working Papers in Linguistics,* 1:321–339.

Hudson, Richard. (1984). *Word Grammar.* Blackwell.

Jäppinen, Harri; Lehtola, Aarno; and Valkonen, Kari. (1986). "Functional structures for parsing dependency constraints." In *Proceedings of the 11th International Conference on Computational Linguistics* (COLING-86), 461–463.

Johnson, Mark. (1985). "Parsing with discontinuous constituents." *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics,* 127–132.

Kashket, Michael B. (1986). "Parsing a free-word-order language: Warlpiri." *Proceedings of the 24th Annual Meeting of the Association for Computational Linguistics,* 60–66.

Mel'čuk, I. A. (1988). *Dependency Syntax: Theory and Practice.* State University Press of New York.

Robinson, Jane J. (1970). "Dependency structures and transformational rules." *Language* 46:259–285.

Schubert, Klaus. (1987). *Metataxis: Contrastive Dependency Syntax for Machine Translation.* Foris.

Shieber, Stuart M. (1986). *An Introduction to Unification-Based Approaches to Grammar.* (CSLI Lecture Notes, 4.) Stanford: CSLI.

Starosta, Stanley. (1988). *The Case for Lexicase.* Pinter.

Starosta, Stanley and Nomura, Hirosato. (1986). "Lexicase parsing: a lexicon-driven approach to syntactic analysis. In *Proceedings of the 11th International Conference on Computational Linguistics* (COLING-86).

Tesnière, Lucien. (1959). *Éléments de la Syntaxe Structurale.* Klincksieck.

Notes

1. The early stages of this work were supported by National Science Foundation grant IST-85-02477. I am grateful to Norman Fraser and Richard Hudson for comments and encouragement.
2. On dependency grammar in general see Tesnière 1959, Hays 1964, Robinson 1970, Hudson 1986, Schubert 1987, Mel'čuk 1988, and Starosta 1988. In Hudson's system, a single word can have two heads provided the grammatical relations connecting it to them are distinct.
3. As pointed out by an anonymous reviewer for *Computational Linguistics.*