A Hybrid Model of Human Sentence Processing: Parsing Right-Branching, Center-Embedded and Cross-Serial Dependencies

Theo Vosse
Gerard Kempen
University of Nijmegen, The Netherlands

December 1989

Abstract

A new cognitive architecture for the syntactic aspects of human sentence processing (the Unification Space) is tested against experimental data from human subjects. The data, originally collected by Bach, Brown and Marslen-Wilson (1986), concern comprehensibility ratings for three types of verb dependency constructions in Dutch and German. A satisfactory fit is obtained between comprehensibility data and parsability scores in the model.

Published as: Vosse, Theo & Kempen, Gerard (1991). A hybrid model of human sentence processing: parsing rightbranching, center-embedded and cross-serial dependencies. In: *Proceedings of the Second International Workshop on Parsing Technologies* (Cancun, Mexico, February 1991).

1. Introduction

In a recent paper (Kempen & Vosse, 1990), we have proposed a new cognitive architecture for the syntactic aspects of human sentence processing. The model is "hybrid" in the sense that it combines symbolic structures (parse trees) with non-symbolic processing (simulated annealing). The computer model of this architecture — called the *Unification Space* — is capable of simulating well-known psycholinguistic sentence understanding phenomena such as the effects of Minimal Attachment, Right Association and Lexical Ambiguity (cf. Frazier, 1987).

In this paper we test the Unification Space architecture against a set of psycholinguistic data on the difficulty of understanding three types of verb dependency constructions of various levels of embedding. The data were collected by Bach, Brown and Marslen-Wilson (1986) and concern comprehensibility ratings of cross-serial, center-embedded and right-branching constructions as illustrated by (1). Subjects rated two types of verb dependencies: right-branching and either center-embedded (German) or cross-serial (Dutch) dependencies.

		Dependency type
(1a)	… when John saw Peter walk	Right-branching
(1b)	… als Johan Peter laufen sah	Center-embedded (nested)
(1c)	… toen Jan Peter zag lopen	Cross-serial (crossed)

The right-branching constructions are quite common in Dutch and German. German sentences were rated only by native speakers of German, Dutch sentences only by native speakers of Dutch. Figure 1 shows the obtained comprehensibility (or rather, *in*comprehensibility) ratings for four "levels" (the term level refers to the depth of embedding; level 1: one clause, without embeddings; level 2: two clauses, one embedded in the other as in (1), etc.). Notice that the (Dutch) crossed dependencies were consistently rated easier to understand than the (German) nested dependencies. From level 3 onward, the right-branching structures are much easier to understand then their crossed or nested counterparts.

In Section 2 we outline briefly the type of grammar we use to represent syntactic structures. The parsing mechanism capable of building such structures is described in Section 3. Section 4 is devoted to design and results of the computer simulation. In Section 5, finally, we draw some comparisons and conclusions.

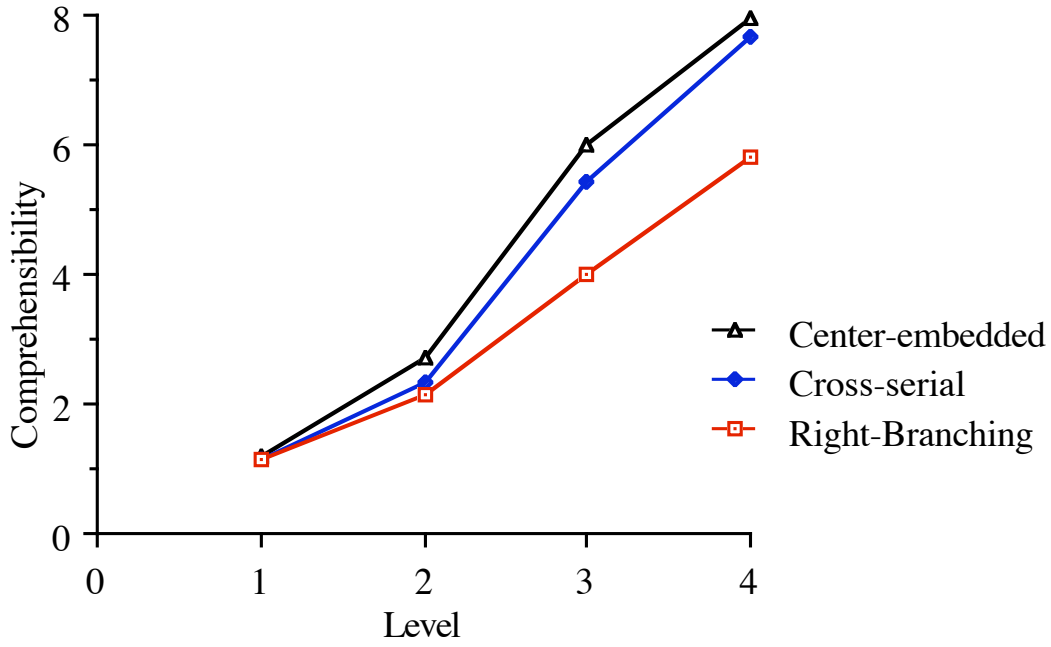

Figure 2. Comprehensibility ratings for various construction types and depths (1 = very easy, 9 = very hard).

2. Segment Grammar

Kempen (1989) introduced Segment Grammar as a formalism for generating syntactic trees out of so-called segments. A segment is a node-arc-node triple, the top node being called "root" and the bottom node "foot". Both root and foot nodes are labelled by a syntactic category (e.g. S, NP) and have associated with them a matrix of features (i.e., attribute-value pairs). Arc labels represent grammatical functions. See Figure 2 for some examples. All syntactic knowledge a segment needs (including ordering rules) is represented in features.

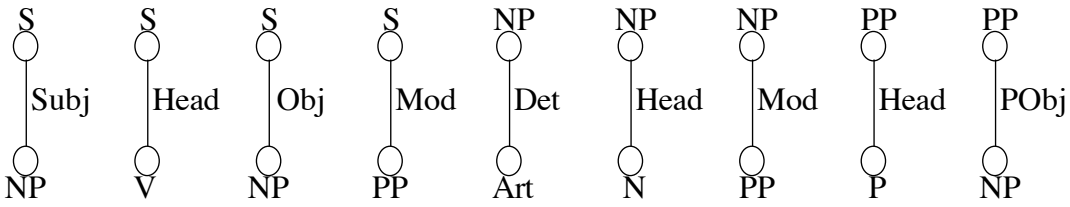

Figure 2. Various types of syntactic segments.

The basic tree formation operation is *unification* of the feature matrices of nodes which carry the same category label. In Figure 3 successful unification has been visualized as the merger of the corresponding nodes.

Segment Grammar is completely lexicalized. Every lexical entry specifies a single segment or a sub-tree consisting of several segments. For instance, one entry for the English verb "eat" looks like Figure 4. It specifies the subcategorization features for

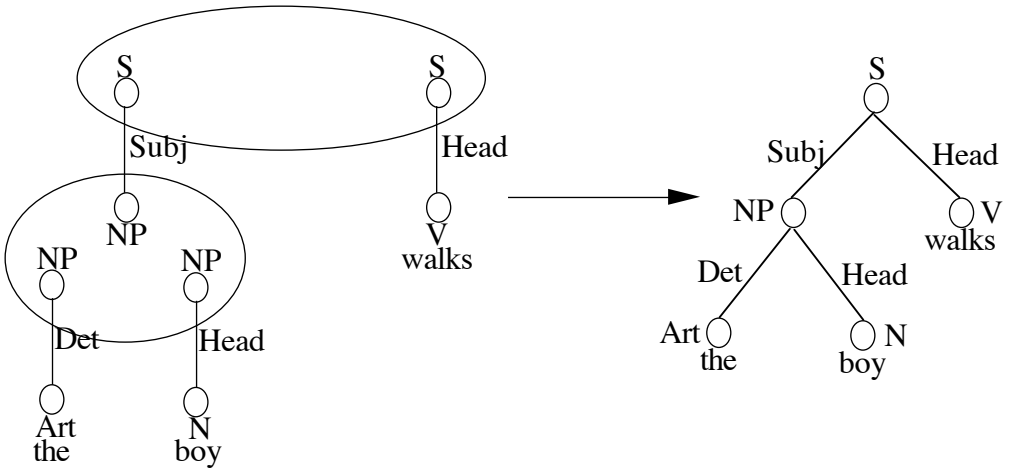

Figure 3. Building a tree through unification.

this verb, including the fact that it can take zero or more modifiers (Mod*) in the form of prepositional or adverbial phrases. For more details about Segment Grammar (including the Dutch sentence generator based on it) see De Smedt (1990).

3. The Unification Space

The dynamics of the Unification Space model were inspired by the metaphor of bio-chemical synthesis. Think of the segments as molecules floating around in a test-tube and entering into chemical bonds with other molecules (unification of nodes). The resulting larger structure may be insufficiently stable and fall apart again. After that, the segments continue their search for suitable unification partners until a stable configuration — that is, the final parse tree — has been reached.

Henceforth, we denote the test-tube by the term Unification Space. Words recognized in the input string are immediately looked up in the mental lexicon and the lexical entry listed there is immediately entered into the Unification Space. In case of an ambiguous input word, all entries are fed into the system simultaneously.

The following principles control the events in the Unification Space:

- *Activation decay.* When the nodes are entered into the Unification Space they are assigned an initial activation level by their lexicon entry. This activation level decays over time.

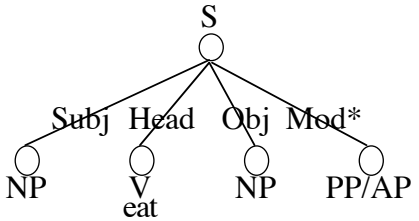

Figure 4. Lexical entry for the transitive verb "eat".

- *Stochastic optimization*. Generally, on the basis of its feature composition, a node could unify with several other nodes present in the Unification Space. In order to make the best possible choice, Simulated Annealing is used as a stochastic optimization technique. If two nodes *can* unify, they actually unify with probability p_U. This probability depends, among others, on the activation level of both nodes and on the grammatical "goodness of fit". (We cannot go into the syntactic and semantic factors which are at stake here). On the other hand, once unified nodes may also break up again, with probability p_B. This probability increases accordingly as the activation of the nodes and/or their grammatical goodness of fit decrease. One consequence of this scheme is a bias in favor of semantically and syntactically well-formed syntactic trees encompassing recent nodes.
- *Global excitation*. Due to the spontaneous decay of node activation and the concomitant rising p_B, all unifications would ultimately be annulled in the absence of a mechanism for intercepting and "freezing" high-quality parse trees. In standard versions of simulated annealing one obtains this effect by making both p_U and p_B dependent on a global "temperature" variable T. In a similar vein, we define a parameter E (global Excitation) whose value, at any point, is proportional to the summed activations of all nodes that currently populate the Unification Space.

  The relation between E on one hand and p_U and p_B on the other is such that, after E has fallen below a threshold value, no unifications are attempted, nor can unified nodes break up again. If the resulting frozen configuration consists of exactly one tree, the parsing process is said to have succeeded. If several disconnected, partial trees result, the parsing has failed. (We cannot discuss here what happens in case of lexical or structural ambiguity. At any rate, the workings of the Unification Space prevent the growth of multiple parse trees spanning the same input string. See Kempen & Vosse (1990) for details.)

We now describe the essence of the computer implementation of the Unification Space model.

1. Time is sliced up into intervals of equal duration. During each cycle, one iteration of the basic algorithm is carried out. This process stops when E has fallen below the threshold value.
2. Words recognized in the input sentence are stored in an input buffer for a limited period of time, T_B. Individual words are read out from left to right at fixed intervals $T_w << T_B$. Their corresponding lexical entries are immediately entered into the Unification Space.
3. During each cycle, two nodes, n_1 and n_2, are picked at random. If their feature composition permits unification, they actually unify with a probability of p_U which

covaries with n_1's and n_2's activation levels. The activation level of the resulting single node is higher than the activation level of either n_1 or n_2.

4. Then, for each segment in the Unification Space, it is determined whether or not it will break away from its unification partner (if any). This event takes place with probability p_B which correlates negatively with the activation level. Whenever *lexical* segments are are involved in a break-up (lexical segments have word classes rather than phrases as their foot labels), their lexical entries are reentered into the Unification Space without delay. Thus they are given a new chance to find a suitable unification partner. The activation levels of reentering nodes are reset to the initial value stored in the lexicon. However, if a word has already been dropped from the input buffer, its lexical entry is *not* reentered.
5. The activation levels of all nodes are adjusted on the basis of the decay parameter and the new value for E are computed.

Due to space limitations, we cannot provide a more detailed description of the model here. We refer the interested reader to Kempen & Vosse (1990).

4. The Simulation Study

In order to avoid controversial assumptions about the shape of the syntactic trees that underlie the three verb dependency constructions, we have devised a simple artificial grammar. It generates right-branching, center-embedded and cross-serial dependencies among pairs of an opening and a closing bracket, e.g. "(){}", "({})" or "({)}". The grammar contains two types of lexical segments (with arc labels Left and Right) and two types of non-lexical segments (with arc labels Mod1 and Mod2). Mod segments are optional but, unlike modifiers of natural languages, no more than one Mod1 and/or Mod2 segment is permitted (cf. Figure 4). We have arbitrarily decided to attach Mod1 and Mod2 segments to the lexical entries of opening rather than closing brackets (see Figure 5). The reader will have noticed that it is the Mod segments that give the grammar a recursive flavor. Center-embeddings are obtained by attaching a segment to the foot of the Mod1 segment. This is controlled by an ordering rule which assigns Right segments a position inbetween the Mod1 and Mod2 segments. By virtue of the same rule, attachment to a Mod2 segment causes right-branching dependencies. Cross-serial dependencies are structurally ambiguous in this grammar. For example, the sentence "({)}" could be assigned a parse tree in which the curly brackets are dominated by the Mod1 segment of the top-level S (rather than by the Mod2 segment as shown in Figure 6). The simulation results for the 15 sentences are depicted in Figure 7. They show the same general pattern as the comprehensibility ratings displayed in Figure 2 above. That is, (1) comprehensibility decreases with increasing depth of embedding, (2) center-embedded dependencies are

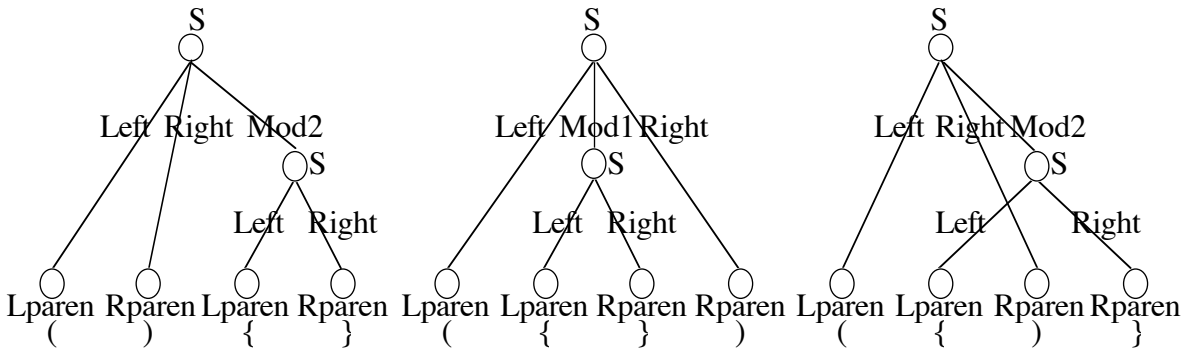

Figure 6. Example parse trees of level 2: respectively right-branching, center-embedded and cross-serial.

harder than cross-serial dependencies, and (3) right-branching dependencies take a strong lead, being much easier to understand than both other constructions.

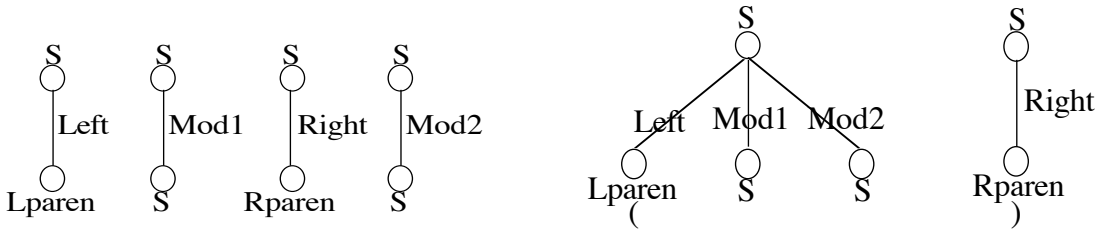

Figure 5. Lexical segments of the grammar, and the lexical entries for '(' and ')'.

The S nodes have associated with them a “bracket type” feature whose value is 'round', 'curly', 'square', etc. This prevents unification of S nodes that dominate brackets of different types, e.g. S-Left-{ with S-Right-]. The reader will have noticed that “correct” parse trees are guaranteed only for sentences which contain no more than one pair of brackets of a certain type.

The actual simulations were run with 5 (levels) times 3 (dependency types) equals 15 different input sentences. Each sentence was fed into the Unification Space 400 times. The parameter settings were exactly equal to those used in the earlier Kempen & Vosse (1990) paper†.

There are also differences between the human data and computer simulation, however. First of all, the comprehension scores for the three dependency types fan out more rapidly in our simulation than in the human subjects. Second, in the human data the first signs of a differentiation between sentence types manifest themselves already at level 2, whereas in our simulation the percentages start diverging at level 3 only. From our previous study (Kempen & Vosse, 1990), we know that the Unification Space is rather sensitive to sentence length. If this applies to human readers as well,

†For Chaos parameter *C* (not discussed in the present paper) we had four different values: .1, .2, .3 and .4. There were 100 runs for each value of *C*. In Figure 7 we show percentages averaged over *C* values.

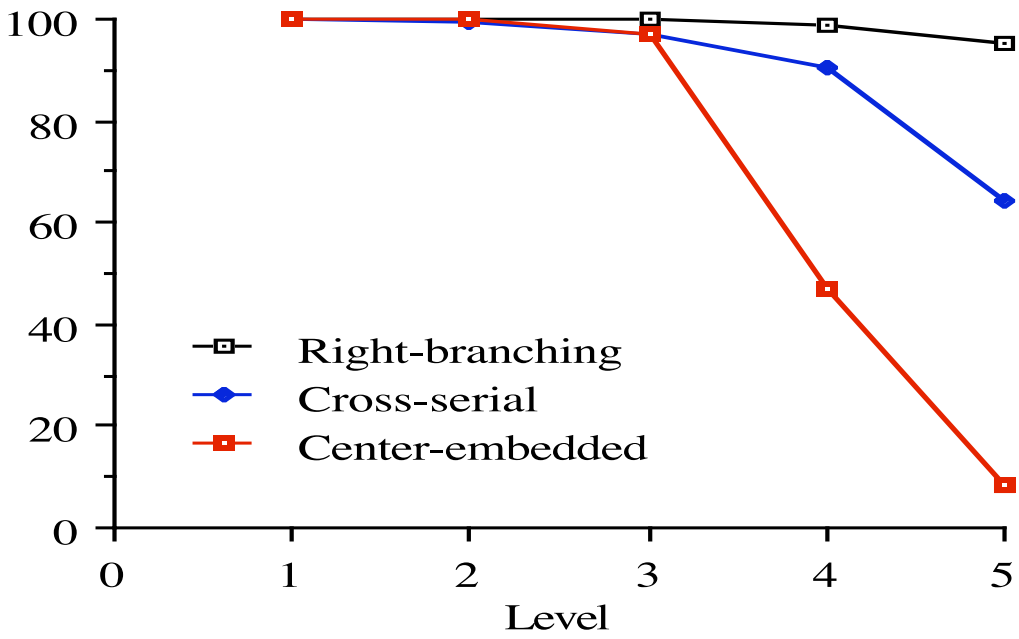

Figure 7. The influence of dependency type and depth of embedding on parsability in the Unification Space model.

we could argue that our level 1 and level 2 scores are too good (in Bach *et al.*'s study, these levels were tested through sentence of 6 to 8 words long).

5. Conclusion

The Unification Space has further substantiated its capability of simulating phenomena of human sentence processing and thereby enhanced its plausibility as a psycholinguistic model. As far as we know, there is no competing model of comparable wide coverage.

A recent paper by Joshi (1989) motivated us to do this present study. He succeeds in obtaining a good fit between Bach *et al.*'s data and a complexity measure deriving from his model, which is based on Tree Adjoining Grammar (TAG) in conjunction with an Embedded Push-Down Automaton (EPDA). We are looking forward to further tests of this type of model against the classical psycholinguistic phenomena reported in the literature on human sentence processing.

6. References

Bach, E., C. Brown and W. Marslen-Wilson (1986) Crossed and nested dependencies in German and Dutch: a psycholinguistic study. *Language and Cognitive Processes*, *1*, 249-262.

De Smedt, K. (1990) *Incremental sentence generation: Representational and computational aspects*. Ph.D. thesis, University of Nijmegen.

Frazier, L. (1987) Theories of sentence processing. In: J.L. Garfield (Ed.), *Modularity in knowledge representation and natural language understanding*. Cambridge, MA: M.I.T. Press.

Joshi, Aravind K. (1989) *Processing crossed and nested dependencies: an automaton perspective on the psycholinguistic results*. Technical Report, Department of Computer and Information Science, University of Pennsylvania.

Kempen, G. (1987) A framework for incremental syntactic tree formation. In: *Proceedings of the Tenth International Joint Conference on Artificial Intelligence* (IJCAI-87), Milan, p. 655-660.

Kempen, G., and T. Vosse (1990) Incremental Syntactic Tree Formation in Human Sentence Processing: an Interactive Architecture Based on Activation Decay and Simulated Annealing. *Connection Science*, *1(3)*.