Language Without A Central Pushdown Stack

Carson T. Schütze and Peter A. Reich
Department of Linguistics
University of Toronto
Toronto, Canada M5S 1A1
E-mail: carsontr@dgp.toronto.edu

Abstract

We will attempt to show how human performance limitations on various types of syntactic embedding constructions in Germanic languages can be modelled in a relational network linguistic framework. After arguing against centralized data stores such as pushdown stacks and queues, we will demonstrate how interconnections among levels of linguistic structure can account for many of the psycholinguistic facts.

1. Introduction

The long-range goal of our research project is to develop and implement in computer simulation a unified, psycholinguistically realistic model of language behaviour—specifically, language production, language comprehension, and language development. Our model is constructed in the framework of relational network linguistics (also known as cognitive-stratificational grammar (Copeland & Davis 1980)), while also incorporating features of spreading activation (Collins & Loftus 1975), competition (Bates & MacWhinney 1987), and other models. In this paradigm, linguistic information is represented in the form of an asynchronous, massively parallel network (in the sense of Schnelle et al. 1988) whose nature can be seen as intermediate between that of mainstream connectionist networks and traditional generative grammars, incorporating aspects of both formalisms. The basics of relational network grammars will be set forth in §3.

Our specific goal here is to describe our attempts to simulate the psychological facts about the production of syntactic embedding in a non-ad hoc way within the above framework. We will show that adequately accounting for human performance requires an examination of at least two other representational levels (or *strata*) in addition to the syntax, namely the *event* level and a level we call the *lexotactics*. In the process we also hope to demonstrate the unified nature of our theory across linguistic levels.

Particular consideration will be given to the cognitive data structures and processing mechanisms required. Our claim in this regard is that temporary memory in the form of a centralized data store, whether it be a pushdown stack, a queue, a collection of stacks, or whatever, is an inappropriate component for modelling human linguistic processing. We will argue instead for a large collection of very simple, localized processing units which possess small amounts of storage by virtue of the possible states which they may be in; in short, a network of finite-state devices. These processors will be general, in the sense that they are useful for cognitive processes outside the domain of language.

2. The Phenomena

The types of sentences we want to account for include centrally embedded and crossed serial structures (see below), in contrast to right- and left-branching structures. This is one area where computational linguists find it useful to look at human limitations, because if there were strict limits on the former forms of embedding, they could allow us to build simpler language processors. Many linguists continue to claim that because of potentially indefinite embedding, something more powerful than a finite-state device is needed. But in some sense all computational linguistics is done on machines which have a limited number of states, and some of us believe that the human brain also has this property.

The matter first arose when Chomsky (1957) argued that natural language cannot be produced by a finite-state device, because of sentences with central embedding to an arbitrary depth. Chomsky's argument was that such sentences are all grammatical because they are "formed by processes of sentence construction so simple that even the most rudimentary English grammar would contain them" (1957: 23). What Chomsky meant by rudimentary processes was recursion, and within the generative framework there was no way to account for people's performance limitations in terms of recursive phrase structure rules.

The issue of whether embedding to arbitrary depth is part of natural language has been debated in the psychological and linguistic literature ever since (e.g., Miller 1962; Labov 1973; de Roeck et al. 1982). While it is still a point of contention, recent carefully controlled experiments suggest that the human ***syntactic*** mechanism, without semantic or pragmatic cues, and without the aid of pencil and paper to construct sentences, does have a sharp limit of one or two levels of embedding (Bruner & Cromer 1967; Reich & Dell 1977). In this paper we make no claim as to how many levels people are able to process, only that there is some small, finite bound. See Reich & Schütze 1990 for further discussion.

The following are some examples of the types of sentences we are particularly concerned with. Although some of these are judged to be marginal or unacceptable by some informants, none of the subsequent discussion depends on precisely where the limits of acceptability are drawn.

- Centrally embedded relative clause constructions in English. A clause B is *centrally embedded* within a clause A if and only if part of A occurs before B and the remainder of A occurs after B.

1. The man *that the dog chased* ate the malt.

2. The man *that the dog* ***that bit the cat*** *chased* ate the malt.

• Verb-final complement constructions in German, which involve *nested dependencies*. Nested dependencies occur when the verbs at the end of the sentence are produced in reverse order from their associated arguments at the beginning. Thus, they must be paired up by working from the edges in toward the middle.

3. Die Männer haben Hans *die Pferde füttern* lehren.
   [the men have Hans the horses to feed taught]
   "The men taught Hans *to feed the horses*."

4. Johanna hat die Männer *Hans* ***die Pferde füttern*** *lehren* helfen.
   [Joanna has the men Hans the horses to feed to teach helped]
   "Joanna helped the men *to teach Hans* ***to feed the horses.***"

• Verb-final complement constructions in Dutch, which involve *crossed serial dependencies*. Crossed serial dependencies occur when the verbs at the end of the sentence are produced in the same sequence as their associated arguments at the beginning. Sentences 5 and 6 are synonymous with 3 and 4 respectively.

5. De mannen hebben Hans *de paarden* leren *voeren*.
   [the men have Hans the horses taught to feed]

6. Jeanine heeft de mannen *Hans* ***de paarden*** helpen ***leren voeren.***
   [Joanna has the men Hans the horses helped to teach feed] (Bach et al. 1986)

• Additional variations which are claimed to occur in Swiss-German (Shieber 1985): sentence 7 is in crossed serial order, 8 is in nested order, and 9 shows a variation of crossed serial order with the upper subject and dative object transposed; all three are synonymous. Analogous variations on 10 are also claimed to be possible.

7. Jan säit, das *mer em Hans* ***es huus*** *hälfed* ***aastriiche.***
   [Jan says that we Hans the house helped to paint]

8. Jan säit, das *mer em Hans* ***es huus aastriiche*** *hälfed.*
   [Jan says that we Hans the house to paint helped]

9. Jan säit, das *em Hans mer* ***es huus*** *hälfed* ***aastriiche.***
   [Jan says that Hans we the house helped to paint]
   "Jan says that *we helped Hans* ***to paint the house.***"

10. Jan säit, das **mer d'chind** *em Hans* ***es huus*** **lönd** *hälfe* ***aastriiche.***
    [Jan says that we the children Hans the house let help paint]
    "Jan says that **we let the children** *help Hans* ***paint the house.***"

3. The Basics of Relational Networks

A relational network consists of a collection of nodes of various types, and connections between them, known as *wires*. Language processing consists of activity flowing through the network or, in the case of acquisition, growing new wires and nodes in the network. Signals move in both directions along the wires from one node to the next. Each node is an independently operating processing unit. All nodes of the same type have the same finite-state definition. Their behaviour consists of sending signals out along the wires to which they are connected, and possibly changing state. Signals are one of a small number of possible types, e.g. *production* (also called *positive*), *feedback*, *failure* (or *negative*) and *anticipation*.

There are currently approximately 25 types of nodes required in our system, expressing various explicit and implicit features found in context-free phrase structure grammars and other formalisms. Each type of node is represented graphically by a distinct shape. One basic building block is the *concatenation node*, equivalent to the phrase structure rule $a \rightarrow b\ c$ (see Figure 1). When a positive signal comes into the node via its top wire, which we label the a-wire, a positive signal will be sent down the b-wire to produce the first element, and the node will change state to 'remember' this fact. If the production of that element succeeds (as indicated by a positive feedback signal returning on the b-wire), a positive signal will be sent down the c-wire to produce the second element in the concatenated sequence, and the node will change state again. Upon the c-wire's successful completion, positive feedback is sent up the a-wire and the node returns to the initial state, known as *state zero*.

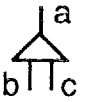

FIGURE 1: Concatenation Node

Other major types of nodes include: *disjunction*, which allows a choice of one alternative among many paths of production (e.g., a verb may be realized as "sing", "think", "walk", etc.); *precedence disjunction*, which also allows a choice of alternatives, but tries them one at a time, stopping as soon as one succeeds; *conjunction* (used in the Boolean sense), which simply fires all its downward wires at once when a production signal comes in the top; and *inverse conjunction*, which requires that two or more separate conditions must be signalling for a path to be followed (e.g., the pronoun "we" can only be produced if plural, first person, and nominative case are all being signalled).

In its gross structure, the network we propose connects a general memory for events to a *semotactic* level, a *lexotactic* level, a *syntactic* level, a *phonological* level, an *articulatory* level, and an *auditory* level. The same type of structure is found in all the levels, and except for the last two, all strata are used in both production and understanding. The syntax defines the sequence in which morphemes are built into words, phrases, and clauses. The phonotactics defines the sequence in which the sounds are combined into

clusters, syllables, and phonological feet. The semotactics defines which concepts can be associated with which types of participants. (For example, the concept "fill" allows an agent and an affected participant, among others, and the affected must be a type of container.) The lexotactics constrains the choice of vocabulary and its syntactic position on the basis of which elements of an event are to be expressed. The *lexicon* of the language, itself in the form of a network, is connected to all major areas of language structure, and is in part what binds the levels to one another. Each word, morpheme, or idiom connects to meaning, syntax, and phonological representation. In addition, there are some wires which pass control information between strata.

Thus, the major differences between relational networks and connectionist ones à la Rummelhart and McClelland (1986) are that the former use several types of nodes with different behaviour, and the actions of each node depend on an internal state in addition to the incoming signals. Furthermore, output signals can be a more complex function of the input signals than simple weighted sums.

4. Right Branching: Iteration with Clean-Up

We now focus our attention on the ***production*** of embedded clauses in relational networks. Relational network syntax makes a strong distinction between right-branching clauses and centrally embedded ones. (Left-branching is handled similarly to right-branching.) In a *right-branching* structure, once an embedded clause is complete, the superordinate clause will also be complete (as in "This is the cat [that killed the rat [that ate the malt [that was stored in the house [that Jack built]]]]"). In such sentences there is no need to preserve any information about the superordinate clause once an embedded one has begun; in fact, from a psychological point of view, it is undesirable—we do not wish to posit more demands on working memory than are actually required to do the job. Hence, in our model, the superordinate clause is explicitly *cleaned up* before a right-branching embedded clause is begun. By cleaned up, we mean that the nodes involved in its production are returned to state zero by sending a positive feedback signal up through the syntactic network; that signal eventually reaches the top of the clause structure, at which point the embedded clause may begin.

For this clean-up to happen when it does, namely ***before*** the start of an embedded clause, the syntax must 'know' whether the current syntactic constituent is the final element of its superordinate clause. There is no independent way for the syntax to make this determination, since a direct object, say, might be followed by an indirect object, or by any number of prepositional phrases. Therefore, we must add something elsewhere in the network to allow this condition to be recognized. What we add is a control wire from the lexotactics to the clause-heading node in the syntax, which will signal when the final participant (defined broadly) is underway. (The lexotactics already has access to all participants of a clause right from its start, and is notified when each participant begins to be realized, for independent reasons.) We note that in some sense the syntax is no longer completely autonomous. Whether this is actually a drawback is partly an empirical psycholinguistic question; studies of Wernicke's aphasics could be relevant.

5. From Iteration to Recursion: Central Embedding

We have now described how, in cases of right-branching, each clause starts with an essentially pristine syntactic network, and therefore little more needs to be said about how indefinite iteration is possible in a finite-state device. The more difficult cases are centrally embedded and crossed serial constructions. In such structures, it is clear that a portion of some clauses is delayed, i.e. prevented from being realized, until some time after its usual (simplex clause) position. In most computational approaches, a centralized (though possibly implicit) data structure, be it a stack or a queue, is used to store these elements until it comes time to realize them. We see a number of problems with this approach.

First of all, there is a certain intuitive appeal to the suggestion that in people, currently active information is distributed in shallow storage across the cognitive network, rather than localized in a single, deep data store. (For instance, parking your car multiple times leads you to forget previous parking spots, but not how much money is in your wallet.) Secondly, it is not clear what a central store should look like in order for it to account for both 'queue' and 'stack' types of languages, i.e. crossed serial and strictly nested-order ones, especially since both orders may occur in a single language. Models which have been proposed to handle both cases have typically involved powerful formalisms, such as a sequence of stacks in Joshi's case (1985, 1990). The resulting processor is more powerful than a pushdown automaton (it can recognize some strictly context-sensitive languages), and it is our belief that structures in the brain are simply not this powerful. These two arguments are independent of what has often seemed to be the central quarrel relational network theorists have with other computational models, namely that they allow for nesting to unlimited depth. It might be simple enough, if somewhat arbitrary, to impose a finite limit on the size of stacks or queues in other models, and thus limit their generative power to that which we believe humans are capable of. However, this would not address our other objections.

Our proposal is as follows. When the cognitive representation of an event becomes active, all its composing elements, i.e. the action and all the participants in it, are fired simultaneously. However, the realization of those elements is held 'in check' until the syntax allows them to come out, one at a time. The realization of any given participant may involve producing one or more clauses which describe it (e.g., relative clauses, sentential complements), and these expansions may be produced before all the elements of their superordinate clause (in particular, the verb) have come out. However, since all aspects of an event fire simultaneously, the superordinate verb will have already been signalled. This is necessary because the choice of verb may affect the realization of its associated participants; in particular, it may determine

in which syntactic position they must be realized. For example, the sentences "George borrowed the book from Sue" and "Sue loaned the book to George" both describe the same event, but the choice of verb has forced George into subject position in the former case, and indirect object in the latter.

The already-signalled superordinate verb will have generated an anticipation signal up towards the *verb-completion* wire of the syntax. (German and Dutch syntax allow only one verb immediately after the subject. Any remaining verbal elements are placed at the end of a clause, in what we will call the verb-completion position.) It is the handling of this signal, and any subsequent verb anticipations which come up, which determines the eventual ***order*** of production. (We are assuming for simplicity that verbs come out only in verb or verb-completion positions, although possibly as part of the 'wrong' clause.) The ***limit*** on how many nested clauses are possible turns out to be totally unrelated to this structure, deriving instead from the finite-state definitions of the nodes themselves (see § 7).

How does the network keep track of which order verbs should come out in? The dashed box of Figure 2 shows the relevant structure. This structure contains ***n*** *placeholder nodes,* where n = 1 + the number of possible embeddings in the language. (In this discussion, we will assume n = 3.) The placeholders are labelled p1, p2 and p3 in the figure. Each is connected to every verb of the language, and acts as a 'slot' for remembering one verb. Whenever a verb-completion is required by the syntax, the network attempts to realize the first verb slot, then the second if the first was empty, and so on. These realizations will succeed if verbs have already been signalled from the events which they describe, the E's in the diagram. A verb signalling in this way tries to 'turn on' one of the positions in the sequence of possible verb-completions, and succeeds at doing so if and only if no other verb has already filled that position. In the case of crossed serial orders such as Dutch, verbs try to occupy the first slot first, exactly as shown in Figure 2. For nested-order languages such as German and English, the wiring from r's to p's is exactly reversed, so that a verb first tries to fill the ***last*** available slot, then works its way forward to earlier slots until it finds one unoccupied (see Figure 3, which would replace the dashed box of Figure 2).

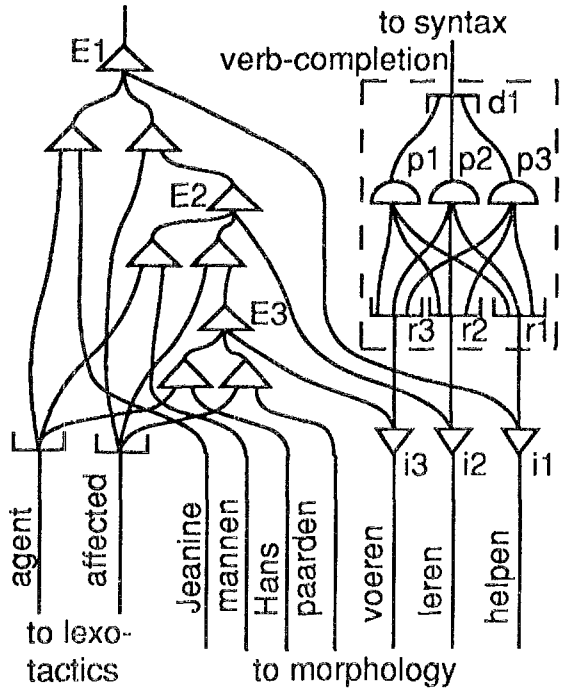

FIGURE 2: Fragment of the network for Dutch

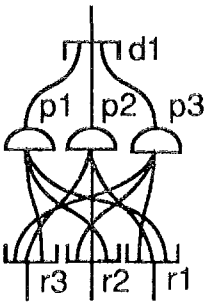

FIGURE 3: German network fragment

6. A Detailed Example

As an example, let us consider the production of the Dutch sentence 6, assuming sentences of such complexity to be possible under some circumstances. The sentence involves three events, represented by the conjunction nodes E1, E2 and E3 in Figure 2. The events are: «Joanna helping the men» (E1), «the men teaching Hans» (E2), and «Hans feeding the horses» (E3). These events will fire in the order just stated, since this is how they are hierarchically arranged in the event structure (E2 and E3 each modify or constitute a participant of the next higher event). As production begins, "Jeanine", "helpen" and E2 fire simultaneously. "Jeanine" is realized immediately, but "helpen" cannot be immediately realized, for the following reason. The first verb position of the clause has been filled by the auxiliary verb "heeft", which is the realization of a semantic element which marks this scenario as having taken place in the past. Since "helpen" could not be realized in this position, it must come out in the verb-completion position. Therefore, it causes an anticipation to fire up from inverse conjunction node i1 towards that position. That anticipation will be directed up to the first verb position, namely placeholder node p1, by the routing node r1. (A *routing node* is an inverted precedence disjunction which attempts to send a signal up its leftmost wire to the placeholder at the other end. If that fails, it tries to send the signal up its remaining wires in order from left to right.) Since "helpen" is the first verb to signal in this sentence, p1 will accept the anticipation and remember which of its wires the signal came in on.

While all this is taking place, the syntax has begun realizing the subordinate event E2, «the men teach Hans». "Mannen" can be immediately realized. "Leren" cannot, but it will again send an anticipation signal, this time via i2 up into the verb structure. This will be routed by r2 to the first position (p1) once again, but this time the anticipation will be rejected, because there is already a verb pending for this position. A failure signal is sent down by p1 to signal this fact, and r2, seeing that cancellation, now tries sending the anticipation up to the second position, p2. This time the anticipation will be accepted, since nothing has previously come up

to this point, and the verb's wire will be remembered. Similarly the third verb, "voeren", will be routed to the third slot when E3 fires, and remembered by p3.

Now, as E3 is realized by the syntax, the syntax will license a verb-completion following the object "paarden", since there are no more participants in the lowest clause. As the verb-completion signal comes down, it passes through the precedence disjunction node d1, which tries each of its output wires in turn from left to right until one succeeds. Its first output leads to p1, which will succeed (since an anticipation has previously come up to it), and finally permit the first verb, namely "helpen", to be phonetically realized. (Node p1 remembered the wire which led to the morphological representation of that verb.) Positive feedback from this production will trigger p1 to return to state zero, and pass the feedback on up. Since the first alternative wire of the precedence disjunction d1 succeeded, none of the others will be touched. The end of the verb-completion is signalled by d1, to which the syntax responds by 'unwinding' the complement clause loop (not shown in Figure 2) once.

We are now at the point of having finished all the participants in the middle clause (the «the men teaching Hans» clause), so all that remains at this level is the verb. Again the syntax signals the verb-completion wire, again the precedence disjunction d1 tries the first path, but this time it will fail, because no verb is waiting at p1. In state zero, this placeholder node sends a negative signal up to the precedence disjunction, which must therefore try its next wire. This one will succeed, producing the verb which was remembered by p2, namely "leren". Similarly, as the syntax unwinds once more and signals for a verb-completion once more, the precedence disjunction will eventually find the verb held by p3 at the third position, namely "voeren", and the sentence will be complete.

Incidentally, a close analogue to this method can be used to account for the order of appearance of noun phrases across embedded clauses as well. In the case of English, we can use a structure like Figure 3 to hold onto the direct object of a superordinate clause until after the direct object of an embedded relative clause has come out. For example, in "The man who liked the dog hated the cat", "the cat" is the first direct object to be made available by the event structure, since it is a participant in the superordinate event, but the first direct object to come out is "the dog", so object NPs are realized in last-in, first-out order. Thus the handling of participants provides additional motivation for the types of nodes and structure that we have posited to handle verbs.

7. Some Theoretical Issues

In a syntax in which nodes are finite state devices, the job of remembering the status of a clause falls on each and every node in the network, as follows. Suppose a node requires a set of *s* states to handle processing within one clause. Then in the worst case, for each of those states it will need a copy of the entire set to use for processing an embedded clause. Each set corresponds to remembering a different place where the superordinate clause was left off. The total number of states in the node will be s^n, where n once again is the number of possible pending clauses. This approach is analogous to a programming language that does not allow subroutines. In such a language, a copy of a recurring block of code must appear at each place where it could be needed. This tendency to exponential growth could account for why languages seem to impose such severe restrictions on the amount of central embedding or crossed serial dependency.

The interesting and crucial thing about the way the process described in §6 was carried out is that the mechanism for remembering verbs was totally independent of the mechanism which ordered them for output. That is, while a particular set of nodes each remembered which one verb was associated with its slot, the ***connections between*** nodes determined the order of output relative to the order of signalling. This means that all the various orderings which occur cross-linguistically can be accounted for by the same inventory of nodes. No additional data structure is required; all that we must do to 'convert' from Dutch to German word order is to rewire the connections between the upward routing nodes and the placeholder nodes, so that slots are tried in exactly the opposite order. To handle the fact that a single language (like Swiss-German) may use different orders depending on syntactic context (or even stylistic factors), all we need to do is have the verb-completion wire branch into all the options, each of which will have its own precedence disjunction and set of placeholder nodes. The upward anticipation from a particular verb will be sent simultaneously to all the different orderings, and the syntax will choose the appropriate one and cancel the others.

The close symmetry between German and Dutch in our model would seem to be a psycholinguistic shortcoming, given Bach et al.'s (1986) result that Dutch is easier to process than German. However, we believe that, to the extent that their results are meaningful, they are ***not*** attributable to a queue versus stack difference, but rather to something along the lines of Joshi's (1990) proposed Principle of Partial Interpretation, whereby the syntax can't forget about a clause unless an argument slot to place it in has already been processed.

One could argue that, viewed somewhat abstractly, our collection of nodes and wires in fact implements a finite-sized convertible queue/stack. Our basic response to this is to point out once again that that is essentially an artifact, having been built up out of independent, lower-level components. As for the particular size ('depth') of the data structure being stipulated, this really is not troubling. Note that such a structure could be any size—nothing in the node definitions would limit it to size three or four, since expanding it only requires adding more nodes. However, more than some small fixed number of verbs can never be realized nestedly, because the syntax simply will not be able to call for them. As described above, the definitions of the nodes which the syntax makes use of simply break down after a couple of nestings. It is therefore reasonable to postulate that the acquisition process would have no reason to build the verb structure any larger than the syntax had ever called for. And with regard to the node definitions

themselves being arbitrary in their maximum nesting limitations, this is certainly true in the sense that we define them to be precisely powerful enough to do what humans can do with syntax. (It is possible to imagine that humans could have evolved with the capacity for, say, one fewer or one more nesting; we would not expect that number to follow from anything else.) The point once again is that this limitation is distributed throughout the network, rather than being a function of the total amount of storage available.

8. Areas for Further Research

Through detailed computational modelling we have made significant progress in analyzing our theory, finding flaws and oversights, and making it more rigorous. We believe that, with the complexity of linguistic models as it is today, no theory can lay strong claims to adequacy, completeness, correctness, etc. unless it has been tested in a computer simulation. Having reworked the theory several times over a period of only a few months, we cannot stress this point vigorously enough.

There are several important questions which our research has not yet addressed. One major issue involving high-level control between strata is that of precisely where and how the decision is made that a subordinate clause is to be produced. In a highly interconnected semantic network of events, there will almost always be 'extra' information available which could be used to expand the description of any participant in the form of a relative clause. We believe that many factors go into the decision as to whether or not to carry out this expansion. These would include pragmatic issues such as the purpose of communication, urgency of the conversation, amount of relevant knowledge believed to be possessed by the audience, etc. Even assuming we can wire in the relevant decision criteria, it still remains to show how the lexotactic and syntactic strata are notified that an additional clause is being produced. One possibility is that the firing of a new action (and/or the associated verb) is the trigger.

Additionally, if we look back at the stated goals of the theory in our introduction, it is evident that only one of the three main areas of language behaviour has been explored, namely production. The whole question of how this system works for comprehension has barely been addressed for relational network models in general. The specific issue of embedding is sure to add more wrinkles. Furthermore, accounting for the acquisition of both iteration and recursion is a serious hurdle for any connectionist model of language to overcome. In our case, it will involve the network growing new structure, in addition to modifying connection weights. So far we have concentrated on convincing ourselves that a viable language processor can be created in network form, whereas connectionists more often are concerned with exploring how much information can be acquired when starting from a tabula rasa. While we have no clear ideas on how acquisition should proceed in our framework, we believe we have at least come up with a possible structure as an end-goal for future acquisition models to strive towards.

Acknowledgements

We would like to thank Elizabeth Cowper, Jan Wiebe and Graeme Hirst for their comments on a draft of this paper. This research was supported by a grant to the second author from the Social Sciences and Humanities Research Council of Canada.

References

Bach, Emmon, Colin Brown & William Marslen-Wilson (1986) Crossed and nested dependencies in German and Dutch: A psycholinguistic study. *Language and Cognitive Processes* **1:4**, 249-262.

Bates, Elizabeth & Brian MacWhinney (1987) Competition, variation, and language learning. In B. MacWhinney, *Mechanisms of language acquisition*, Hillsdale, N.J.: Lawrence Erlbaum, 157-193.

Bruner, Jerome S. & R. Cromer (1967) An unpublished study of eye movements reported in Harvard Center for Cognitive Studies Seventh Annual Report, p. 7.

Chomsky, Noam (1957) *Syntactic Structures*. The Hague: Mouton.

Collins, Allan M. & Elizabeth Loftus (1975) A spreading activation model of semantic processing. *Psychological Review* **82**, 407-428.

Copeland, James E. & Philip W. Davis, Eds. (1980) *Papers in Cognitive-Stratificational Linguistics*. Rice University Studies, Vol. 66. Houston, TX: Rice University.

Joshi, Aravind K. (1985) Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In D. Dowty, L. Karttunen & A. Zwicky, eds., *Natural Language Parsing: Psychological, computational and theoretical perspectives*, New York: Cambridge University Press, 206-250.

Joshi, Aravind K. (1990) Processing crossed and nested dependencies: An automaton perspective on the psycholinguistic results. *Language and Cognitive Processes*, to appear.

Labov, William (1973) The place of linguistics research in American society. In Eric Hamp, ed., *Themes in linguistics: The 1970s*, The Hague: Mouton.

Miller, George A. (1962) Some psychological studies of grammar. *American Psychologist* **17**, 748-762.

Reich, P.A. & G.S. Dell (1977) Finiteness and embedding. In R.J. DiPietro & E.L. Blansett, Jr., eds., *The third LACUS forum*, Columbia, S.C.: Hornbeam Press, 438-447.

Reich, Peter A. & Carson T. Schütze (1990) Syntactic Embedding: What Can People Really Do? Working paper in the Computer Applications Group, Department of Linguistics, University of Toronto.

de Roeck, Anne, Roderick Johnson, Margaret King, Michael Rosner, Geoffrey Sampson & Nino Varile (1982) A Myth About Centre-Embedding. *Lingua* **58**, 327-340.

Rumelhart, David E. & McClelland, James L. (1986) *Parallel distributed processing: Explorations in the microstructures of cognition*. Cambridge, MA: MIT Press.

Schnelle, Helmut (moderator), with (alphabetically) G. Cottrell, P. Dey, J. Diederich, P. A. Reich, L. Shastri & A. Yonezawa (panelists) (1988) Panel: Parallel Processing in Computational Linguistics. In Dénes Vargha, ed., *Proceedings of Coling Budapest*, Association for Computational Linguistics, 595-598.

Shieber, Stuart M. (1985) Evidence against the context-freeness of natural language. *Linguistics and Philosophy* **8**, 333-343.