Connectionist Parsing

Garrison W. Cottrell[1]
Department of Computer Science
University of Rochester

Abstract

We have proposed a neural network style model of language processing in an effort to build a cognitive model which would simultaneously satisfy constraints from psychology and neurophysiology. This model was successful in disambiguating word senses in semantically determined sentences, but was unable to distinguish Agent from Object in semantically reversible sentences such as "John loves Mary." In this paper we rectify the matter by specifying the syntactic portion of the model, which is a massively parallel, completely distributed connectionist parser. We also describe the results of a simulation of the model.

1. Introduction

We have proposed a massively parallel, highly distributed neural network model of language comprehension (Small, Cottrell & Shastri, 1982; Cottrell & Small, 1983; Cottrell 1984, 1985) based on the connectionist paradigm of Feldman & Ballard (1982). Similar systems have been proposed by Gigley (1982), Pollack & Waltz (1982; 1985) and Selman & Hirst (1985). Our model specifies independent pathways from the lexicon to syntactic and semantic modules that operate in parallel, mutually constraining one another's activities. Previous work has concentrated on the semantic module and the lexical access mechanism. In this paper we present the design of the syntactic processor, and the results of a simulation of the model. While this is a preliminary version, lacking many of the features one would want in a complete parsing system, it demonstrates the feasibility of the approach, and there are aspects of this model which are rather general and interesting as partial specifications of a parsing mechanism in their own right. These include the mechanism for assigning constituents to their roles which has a natural interface with our model of semantic role assignment, and the ability to represent syntactic attachment preferences. Also we might include here the fact that this parser uses the massive parallelism inherent in the connectionist approach, with the concommitant distributed decision making. And, unlike the parser of Pollack & Waltz (1982; 1985), there is no interpreter that builds a network based on the input sentence and then runs it in parallel; this parser uses a network that is fixed, yet responds flexibly to the input.

In order to implement this parser, we developed a grammar formalism suitable to our needs, and developed a LISP program that takes as input a dictionary and a set of grammar rules, and outputs commands to the ISCON simulator (Small, et al. 1982) to build the network. The network implements a top-down parser. Expectations compatible witha an S are set up. As input comes in, (words are activated at fixed intervals) the structures compatible with the input continue to be active, while productions that are incompatible with the input are turned off. After a settling period, a stable coalition in the network represents the parse tree. Ambiguous lexical items at the leaves are disambiguated through feedback from the parse tree as it develops. One important difference from previous parsers of this type (Pollack & Waltz, 1985) is that the assignment of constituents to their syntactic roles is explicitly represented by nodes in the tree called *binding nodes*. The combination of top-down expectations and bottom-up evidence comes about at the binding nodes, which combine the evidence and compete with one another through mutual inhibition. These nodes can be used to interface with case role assignment in the semantic analyzer.

---

[1]Author's present address: Institute for Cognitive Science C-015, U.C.S.D., La Jolla, California 92093.

As a first cut, this parser is less elegant than one would hope, and less powerful than what is needed for a "real" parser. Although it can handle recursive constructs, and thus has a mechanism for "recruiting" constituent recognizers as needed, no attempt was made to do the same for buffer positions, so it accepts only a sentences of a fixed maximum length. Also, the fact that the network is of fixed size means that there are only a fixed number of constituent recognizers, so the number of constituents of each type that can occur in a sentence is limited. We believe that McClelland's programmable buffers (McClelland, 1985) might be adaptable to resolving these problems, but exactly how is not obvious. Also, this parser has not been tested on an extensive grammar, therefore, take any claims with a grain of salt. Finally, we haven't done anything about: (1) optional repetition of constituents, (2) feature marking of constituents (hence no feature agreement checking), (3) anaphora, or (4) gapping. We would hasten to point out that this doesn't mean that these features can't be implemented in this framework; simply that we haven't done it yet.

The rest of this paper is organized as follows: We first provide a bit of background material on our model to motivate the parser's design, and briefly review some of the psycholinguistic and neurolinguistic data on syntax. Then we describe our grammar formalism, leading into a description of our parser, followed by a sample run of the implementation on a sentence. We will not give an introduction to connectionism due to space limitations. Therefore we will be assuming some familiarity with the paradigm[2].

2. Background

Overview of the model

The overall model consists of a four component, three level network of units that operate in parallel by spreading activation and mutual inhibition, shown in Figure 1. The Lexical level consists of units representing the words in the language, presumably activated by phoneme or letter recognition networks at a lower level, such as those developed by McClelland & Rumelhart (1981). These units in turn, activate units at the word sense level, which buffers the syntactic and semantic features of their definitions. If a word is ambiguous, features corresponding to all of its definitions are activated and compete with one another. From here, the semantic features activate relational nodes in the case network, which makes the assignments of conceptual constituents to their case roles (Fillmore, 1968; Cook, 1979) based on the "best fit" among the (possibly several) predicates, fillers, and case roles. The representation in this network is order-less, and thus it can operate by itself if the roles can be determined from semantic constraints alone, but not when syntactic information is necessary. For example, there is no way for a purely semantic analyzer to make the assignment of Agent and Object in *John loves Mary*, since both John and Mary are equally likely candidates for both roles. In the current design (Cottrell, 1985), fillers are assigned to their roles through the competition

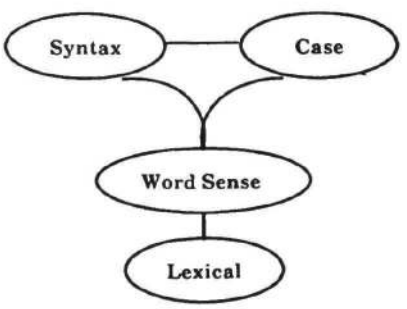

Figure 1. Overview of the system.

[2]The uninitiated reader may consult Feldman & Ballard (1982).

of *binder* units, which explicitly encode the assignments (see Figure 2). Through mutual inhibition, one of these binder units will "win", representing the assignment of, say, Concept1 to the Agent role. It is these units which motivated the design of the syntactic processor, since the kind of information now needed from syntax is of the form "NP1 is the Subject, and NP1 corresponds to Concept1, and the verb is Passive". We will not worry about computing that NP1 corresponds to Concept1 here. Our parser will compute that NP1 is the Subject, however, in the same form as such constituent-role assignments are done in the case network, i.e., through the competition of binding nodes. The result gives a simple interface between the two systems: the binding nodes in one constrain the binding nodes in the other. For example, we can implement the Passive transformation as in Figure 3. Thus, our design of the parser started "from the inside out", with the necessity of binding nodes.

The other thing we would like from syntax is the syntactic disambiguation of lexical items, to prevent spurious predicates and fillers from confusing the semantic system. This can be done through feedback to the syntactic features in the word sense buffer which form a grammatically correct sentence. The decision machinery in this buffer will then kill off the meaning features for the meanings corresponding to the wrong syntactic class, which then will stop sending input to the semantic analyzer.

Psycholinguistic and Neurolinguistic Data

We will very briefly (due to space limitations) review some relevant psycholinguistic and neurolinguistic data. We will try to point out the relevance to the rest of the paper here, rather than return to it later. For a deeper treatment, see (Cottrell, 1985). The current best estimate of what

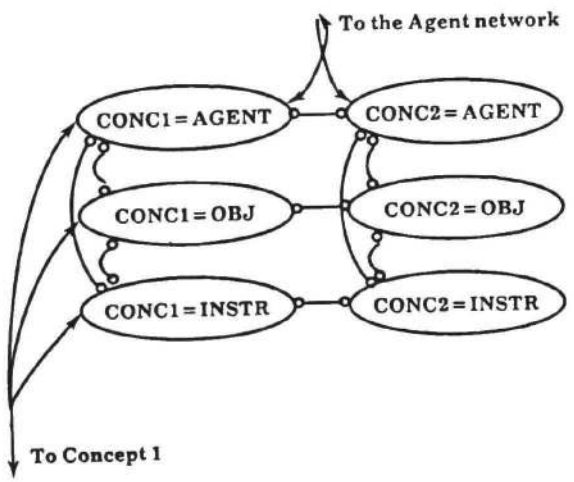

Figure 2. Binding nodes in the case network.

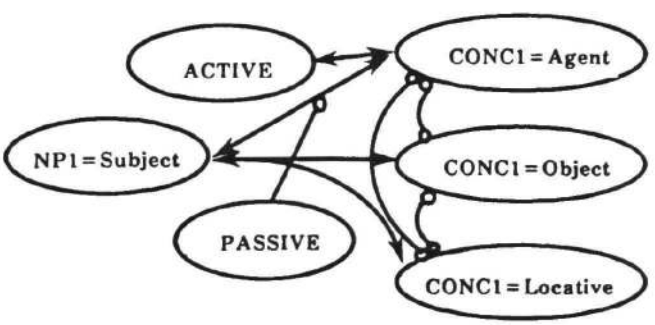

Figure 3. The Passive transformation (assuming the features "Active" and "Passive").

happens when a person hears a word is that all meanings are initially activated, and then by 200 milliseconds after the end of the word, only the appropriate one remains active (Seidenberg et al., 1982). This result is for sentence-final noun-verb and noun-noun ambiguous words, regardless of the strength of context. The exception is the case of noun-noun ambiguous words that are preceded by a word that is strongly related to one of its meanings, in which case it appears that only one meaning is activated. On the other hand, for noun-noun ambiguous words in the first clause of a two clause sentence, where the disambiguating information is contained in the second clause (as in *the teacher looked at her* **pupils** *and noticed that they were dilated*) people appear to be able to maintain both meanings active for as long as 500 milliseconds (Hudson & Tanenhaus, 1984). Thus a model of the human parsing mechanism should activate all meanings of an ambiguous word, and allow them to continue as long as they are compatible with the rest of the sentence, or until a decision is necessary, for example when the sentence has ended.

Turning to higher-level considerations, Frazier (1979) has proposed several principles of the human parsing mechanism including the *Minimal Attachment Principle*: Attach incoming material into the phrase marker being constructed using the fewest nodes consistent with the well-formedness rules of the language. The Minimal Attachment Principle is simply a statement of what appears to be a sound strategy: use the fewest nodes possible to parse a sentence. Without having described the model yet, we can explain Minimal Attachment in terms of it as follows: Imagine a grammar representation which is *active*, in the sense that as parts of productions are recognized, activation spreads to the next part of the production. Different productions in the grammar *compete* for the attachments of constituents that are found in the input. The more nodes involved in a particular interpretation, the farther the activation has to spread, and the longer it takes to activate those nodes that the input actually attaches to. Meanwhile, if there is a representation that matches the input that involves fewer nodes, this will become activated faster and get a head start over the representations involving more nodes. Thus our model explains Minimal Attachment as a *timing* phenomenon, involving the latency of activating simple versus complex representations through a spreading activation network.

Rayner, Carlson & Frazier (1983) have shown that there are purely syntactic preferences for attachment of PP's to NP's vs. VP's in such sentences as *the cop saw the burglar with the binoculars*. When the semantically preferred attachment is different from the syntactically preferred one, people appear to "backtrack" briefly to recover. This argues for an independent contribution of syntax and semantics to the attachment process. It does not necessarily mean that one precedes the other; one system may just operate more slowly than the other. Although we will not delve into it here, there is a natural way to represent these preferences in our system as inhibitory bias (presumably learned through frequency of attachments) between nodes representing competing attachments for a constituent.

Recent data on agrammatic aphasics also supports an independent syntactic processor; Linebarger et al. (1983) have shown that these patients who appear unable to use syntactic clues to comprehend sentences can nevertheless make relatively complex grammaticality judgements. This result implies that agrammatics can *compute* syntactic representations, but can't *use* them to form semantic interpretations. We interpret this data as support for our overall model, given that the observed behavior can be roughly accounted for by a lesion to the constraints between the syntactic and semantic modules.

The Grammar Formalism

Recall that what we want from syntax is information such as "NP1=Subject", represented explicitly as units with this as their value. The grammar we have developed to enable us to automatically generate the parsing network explicitly represents roles and constituents that can fill those roles. Hence we call it a *role-constituent grammar*. We are not aware of any formalism in linguistics that is similar to this although it appears to be weakly equivalent to a context free grammar. However, as has often been remarked, the notation one uses can affect the way one thinks about a problem. In this case, the notation is handy for generating a network which contains binders for constituents to their roles in parent constituents. An example grammar is shown in Figure 4. The left hand side of a production in this grammar is a constituent. The right hand side is a list of roles

S-> Subject.{NP} Predicate.{VP}
Predicate.{VP}

VP-> Main.{VERB} DirObj.{NP}
Main.{VERB} IndObj.{NP DirObj.{NP}
Main.{VERB}

NP-> DetPhrase.{DET} Head.{NOUN}
Head.{NOUN PRO}

Figure 4. A sample role-constituent grammar.

followed by a set of constituents that can fill those roles. In this example, the Head of a noun phrase can be filled by either a PRO a NOUN.

Since there are sometimes a number of alternative constituents that may fill a role, the dot between the role and the set of possible constituents corresponds to a set of binders that must compete for that role. On the other hand, every set that a constituent is in corresponds to a possible role for that constituent, so that binders for that constituent to these roles must compete as well. For example, any particular NP could be the Direct Object, Indirect Object, etc., so binders to these roles must be a) built in to the network and b) mutually inhibitory. The network is set up so that unless a role is expected, then that binder doesn't become activated, so in practice competition does not involve all of the binders.

Another thing to notice about this grammar formalism is that the role a constituent can fill is context sensitive within a production. For example, in the first production for an NP, a NOUN can fill the role of the Head, but not a PRO. In the second production, the Head can either be a NOUN or a PRO. Thus there must be two "Head recognizers", one for each kind of Head.

Now given this grammar, a dictionary containing the possible syntactic classes of the lexical items, and some "magic numbers" (how many copies of each kind of constituent recognizer there will be, and how long the lexical buffer will be), we can generate a network that will recognize sentences that match these rules. After a tour of the lexical buffer, we shall describe this network in greater detail.

The Lexical Buffer

We used a slightly different version of lexical access from the model reported in (Cottrell, 1984) basically because it was easier to implement with respect to the interface with the syntactic analyzer. However, it has some interesting properties in its own right, which we will describe here. For comparison purposes, we show the network for "deck" in Figure 5. The lexical node "deck" activates "grandmother cells" for each of its definitions. These are self-stimulating, to keep the definition around after the lexical node decays. (Thus the lexical node can be re-used later. Activation of successive buffer positions is enabled by control nodes that sequence through the buffer. This figure represents one buffer position.) The definition nodes, in turn, are connected to feature nodes representing syntactic class and meaning. The definition nodes are also mutually inhibitory, but the inhibition weights are such that they can't defeat one another until decisive feedback to the meaning nodes supports one "DEF" node over another. The "meaning" nodes are connected to the case network, and have no further connections to the syntax network. The syntactic class nodes are shared between meanings of the same class. Thus feedback to one of these from a role node above will support both definitions within a class, and kill off the out-of-class syntactic node, followed by the out-of-class meaning and definition nodes. (Of course, if there is only one meaning for a syntactic class, then feedback to that class is enough to cause that "DEF" and meaning node to win.)

Feedback to one of the meaning nodes, e.g. SHIP'S, on the other hand, will kill off the other meaning nodes, and the supported definition node (DEF1) will increase the activation of NOUN, which causes it to win over VERB. However, "DEF2" will get both extra support and extra inhibition.

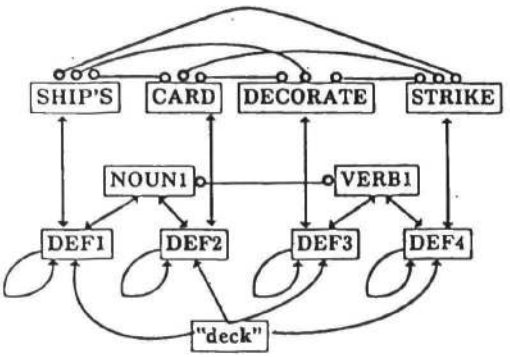

Figure 5. A word sense buffer position. Inhibitory links between DEF nodes omitted.

It gets extra inhibition from "DEF1", but since there is a positive pathway between the two, through the NOUN node, it also gets extra support. The result (with certain parameters) is that "DEF2" gets inhibited, but not below threshold. However, since "CARD" has been killed by "SHIP'S", "DEF2" is invisible to the rest of the network. Only the features compatible with the "DEF1" are visible. A prediction this network makes is that it would be easier to recover from a *within class* mistaken meaning choice, since the "DEF2" node is still firing.

This is the decision machinery for lexical disambiguation. Now, all we need is feedback.

3. The Parser

The parsing network is generated by a LISP program that reads the grammar and a dictionary and outputs commands to the ISCON simulator to build the network. We will start by explaining at a high level what is generated by a grammar rule for a constituent with several productions. Then we will go into slightly more detail about how productions for a single constituent compete with each other, and how the binding nodes compete with each other. Following this, we will skim over some of the ugly details of the whole process, and mention some suggestions as to how they might be circumvented in future designs. This will conclude the "Parser" section; the following section will describe a sample run of the parser.

Overview of the Network Generated by a Production

Figure 6 shows a production and a high level description of the network fragment generated by it. The nodes in Figure 6 are actually representative of networks in the implementation. For the purposes of exposition, however, this level of detail is more appropriate. The S recognizer is connected to two "production recognizers" which correspond to the two productions for an S in the grammar. These implement a simple voting scheme for comparing evidence between the two productions. If the difference in the amount of evidence for the two productions is greater than a certain amount (called the *competition window*) the one with less evidence gives up.

The production recognizers are connected to "role recognizers", that sequence through the roles of the production. As bottom-up evidence comes in for the Subject role, for example, expectations are set up for the Predicate role, activating that recognizer. As mentioned above, since roles can be filled by different constituents depending on what production they are in, there have to be two Predicate recognizers. In any case the control is different for the two; the Predicate recognizer in the first production has to wait for the Subject to be recognized.

The role recognizers operate by enabling the binder nodes for their role, and setting up expectations for the possible role-filling constituents. These constituents are recognized by another set of constituent recognizers that are activated by the role recognizer from a pool of inactive constituent recognizers of that class. As input comes in, it activates the constituents appropriate to it

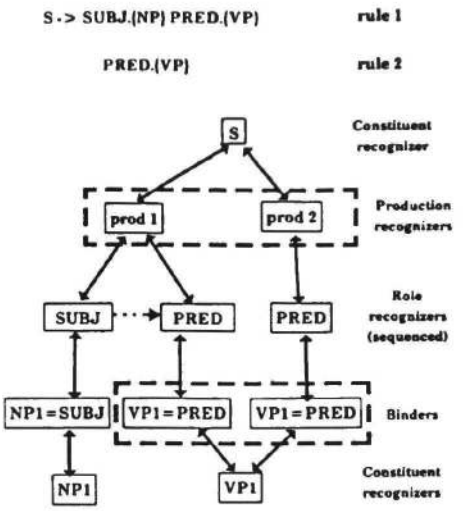

Figure 6. The network generated by a grammar rule.
Dashed rectangles indicate possible competitors.

that are expected, and the binders for that constituent compete. If it was possible, for example, for both Predicate recognizers to be active (it isn't, given that the presence of a Subject would kill off production 2), the two VP1=Predicate nodes would compete with one another. In practice, the competition would be between binders from a constituent to roles in different higher level constituents, such as ones with common prefixes. Binders get feedback from the production recognizers, so that a binding to a well satisfied production will tend to win over others.

This network is repeated for every production in the grammar. It "bottoms out" on syntactic class nodes, for example NOUN or VERB, which are directly connected to the definition node for a lexical item.

Implicit in this description has been the fact that this parser starts out with top-down expectations, and merges them with bottom-up input. So, in the beginning, expectations are generated for everything that could start an S. This means that the constituent recognizers are enabled and ready to go. Then the input that matches with those compete through the binder nodes. The binder nodes represent a merge of bottom-up input with top-down expectations. Whether or not we have a good algorithm for combining these types of evidence, we at least have a paradigm for testing different evidence combination functions.

The end result of a successful parse is that there is a unique binder that has won over its competition for every constituent, and one production that has won for every constituent, from the top level S on down. In the next sections we cover some of the aspects of this model in more detail.

Production Competition

The production competition is a simple voting scheme which favors longer productions over shorter ones, and thus favors "late closure" (Frazier, 1978) of constituents. It works as follows: Every production gets two votes for every "filled" role. "Filled" here means that a binder for a constituent filling that role has won over all competing binders. This is communicated to the production competition network by a higher firing rate for that binder than is possible while it is in competition with other binders. Each production gets one vote for an unresolved role. "Unresolved" here means that there is evidence that a constituent for this role exists, that is, a binder for a role-filling constituent is firing, but it is still competing with other binders. The evidence rule for each production then is:

```
if Max(votes for all productions) - MyVotes < Competition Window
then continue competing;
else lose;
```

In the current implementation, the competition window is 2 votes. Since the votes for this production are included in the Max of votes for all productions, if this is the winning production, the left hand side is zero, and the production continues. If another production gets two votes ahead, it kills this production. This evidence combination rule favors writing grammar productions with alternatives that are not greatly different in length; very long productions, even if only partially satisfied, would always win over shorter ones. However, this seems to be a natural restriction on grammars.

While particular rules like this are certainly arguable (compare to the "normalized" rule used by Selman & Hirst (this volume)), the point is that we have a good framework for evaluating such rules; they are intrinsically interesting because they are completely *local* to the production competition network; there is no global interpreter making decisions about the "best fit" to the grammar. Thus this is a testbed for exploring decision algorithms for a parser that works in a completely distributed manner.

The careful reader will have noticed a problem with the rules as given. See Figure 7 (please forgive the "mix and match" linguistics). Given that these two productions have a common prefix, and given input that matches the prefix, i.e. a "lone" NP, how does production 2 ever win? One answer is to add a "Closure" role to the end of every production that requires no filler (see Figure 8). Then, if no PP comes along, production 2 gets an extra 2 votes and wins. The problem is deciding when the Closure role should fire. Simply having it start firing after an arbitrary interval won't work, since different PP's will have different recognition latencies; they may arrive quickly, unfairly beating production 2 before the Closure role fires (the proper attachment isn't necessarily to this NPbar) or it may arrive too late to come to the rescue of production 1. The answer is to make the Closure role fire when either the PP is recognized, or input inconsistent with a PP is recognized[3]. Then, the shorter production will not win too soon. We can then depend on the semantic feedback to resolve the attachment of the PP (the constraints between syntax and semantics are a two-way street; bindings in semantics will support one attachment over another) and the reader can check that our evidence rule will make the appropriate decision once this binding has been resolved (and not until then). While this appears plausible, we won't feel confident in it until we have tested it on a more extensive grammar[4].

```
Nbar-> Head.{NP} Mod.{PP}    (1)

       Head.{NP}             (2)
```

Figure 7. Production competition: the closure problem.

```
Nbar-> Head.{NP} Mod.{PP} Clos.{}    (1)

       Head.{NP} Clos.{}             (2)
```

Figure 8. The closure problem: A solution.

---

[3]This can be computed directly from the grammar, using the "Follow set" (Aho & Ullman, 1977) (used in predictive parsers of computer languages) of the NP, in this example. Members of the Follow set inconsistent with the PP would cause the Closure role to fire. This is not currently implemented.

[4]We would appreciate counterexamples.

Binders

Binding nodes use a similar rule to the production competition one:

```
if (inhibition - support < window)
{ /* keep going */
  if (inhibition == 0 && support > criterion) then win;
  else continue;
}
else lose;
```

One minor difference from the production competition code is that the inhibition[5] doesn't include what this node sends out. The competitors of a binder are determined from the rule that the same constituent can't be assigned to more than one role, and one role can't be filled by more than one constituent. The inhibition is the maximum of the input from these competitors. "Support" is the sum of bottom-up and top-down input. Top-down input comes from the production evidence network, and reflects how well the **production** is doing. As more roles in a grammar production get filled, the binders to those roles get more feedback. Thus if a binder for another role in the production wins, this is communicated indirectly to the other binders through increased feedback.

Missing Details and Weaknesses

There are several details that have been suppressed in this exposition; most of these stem from the use of a fixed network: Mechanisms had to be implemented to allow role recognizers to select constituent recognizers from a pool of them; the word sense buffer is of fixed length, although a similar recruiting mechanism might work here too; and just the existence of copies of constituent recognizers with identical control structures is not particularly palatable. However, recent advances in connectionist tools make the future look brighter. McClelland (1985) has developed a system called Connection Information Distribution (CID) which allows the storage of connection information in one central network (a *knowledge source*) to be "loaded" into a programmable buffer (like a Hearsay blackboard, Lesser & Erman, 1977) as needed. While the mapping of our system into the CID framework is not immediately obvious, the idea holds promise for avoiding many of the "fixed network" uglinesses.

Also missing here is a detailed description of all of the unit functions and the parameters used (e.g., connection weights and unit thresholds). To get this model to work, some "weight twiddling" was necessary. This unsavory practice is necessitated in part by the unconstrained formalism used, which at present is not amenable to formal analysis. Use of a more constrained formalism can lead to better methods for determining such parameters; see (Selman & Hirst, 1985). Their model also uses pure context-free grammars, avoiding the problems of within-production context sensitivity that forced us to use more copies of recognizers than their model.

Finally, we don't have a convincing demonstration that the networks generated by our program will always converge to a single coherent global interpretation. This points to the need to either (1) convert to a analyzable formalism such as Boltmann machines (Hinton & Sejnowski, 1983) or (2) develop more powerful analytical tools. For the present, we will have to be content with empirical demonstrations.

4. An Example Run

This system was implemented on a VAX/750 running Franz Lisp and C; the network builder was coded in Lisp and fed commands to the ISCON simulator which actually built the network. This in turn was "compiled" into a representation suitable for a much faster network simulator written in C. The actual network for the simple example we will present contained several hundred nodes and over a thousand connections. Hence, we will only give a high level description of the network's behavior.

Given the sentence *he cut the roll*, there are two cases of syntactic ambiguity, *cut* and *roll*.

---

[5]Here we are using the absolute value of the inhibition, which is usually negative.

There are no interesting closure problems, as any difference between this and *he cut*[6] can be handled by using "period" as a lexical item. We simulate reading this sentence by activating each lexical item every 30 steps of the simulation. These are then "read in" to the word sense buffer described earlier, and after 7 more steps, the syntactic class and meaning nodes are activated (these and the other nodes in the buffer accumulate activation slowly). About the time that PRO is activated in the buffer, an NP is expected by the Subject role of the S production. Since "he" is unambiguous, PRO has no competitors and gets highly active quickly. It then rapidly becomes bound to the Head role in the first NP (by iteration 16) since there is no competition for it.

The network then expects a VP, and the two productions for a VP (with and without a Direct Object) set up expectations for a VERB. When "cut"'s features (NOUN and VERB) become activated at clock step 38, they inhibit one another, driving each other below threshold. Since this cuts off the inhibition, they rise up again, like flickering bits, but now feedback from the binder for the VERB to the Main Verb role in the VP gives extra support to the VERB node, allowing it to remain above threshold while NOUN doesn't, resulting in a win for VERB in the second buffer position. Thus, "cut" has been disambiguated as a verb. By a few steps later, the node corresponding to the "meaning" of "cut" as a noun also loses.

After this is propagated up the network, expectations are set up for either a "period" or a Direct Object which can be filled by an NP. The Direct Object role recognizer selects the NP2 recognizer (the next available NP recognizer) which sets up expectations for either a DET, NOUN or PRO. After "the" has been processed, the NP recognizer is only expecting a NOUN, and so when "roll" comes in, it is quickly disambiguated. Eventually the production corresponding to a VP with a Direct Object wins, and the resulting stable coalition represents this parse with the appropriate binding nodes in a highly active state.

Thus our parser has disambiguated the two ambiguous lexical items on the basis of their "fit" into the developing parse tree. Anything incompatible with the expectations developed at the "frontier" of the developing tree was quickly extinguished. If there had been more structural ambiguity, lexical items compatible with either structure would have remained ambiguous until some production won over another. While this may seem implausible, it is compatible with the results of Hudson & Tanenhaus (1984).

5. Conclusions

We have presented a parsing model which has the following features:

1) completely distributed
2) massively parallel
3) automatically generated from a grammar
4) does not use symbol passing
5) matches neurolinguistic data.
6) has a "natural" explanation of Minimal Attachment.

Acknowledgements

We would like to acknowledge helpful discussions with James Allen, Gary Dell, Mike Tanenhaus, and Mark Fanty. The C simulator was written by Sumit Bandopadyay and Mark Fanty. This work was supported under DARPA grant NOO 14-82-K-0913 and NSF grant IST-8208571.

References

Aho, A. & Ullman, J. *Principles of Compiler Design*, Addison-Wesley, Reading, Mass., 1978.

Cook, W.A. Case grammar: The development of the matrix model (1970-1978). Georgetown University Press, Washington, D.C., 1979.

---

[6]Recall that we haven't implemented features other than syntactic class. In particular, there are no verb subcategorization features, so if we wanted to represent that *cut* is different from *left*, we would need to encode them as different syntactic classes in the current parser. Since we haven't, *he cut* is acceptable to our parser.

Cottrell, G. W. A model of lexical access of ambiguous words. In *Proceedings of the National Conference on Artificial Intelligence*, Austin, Texas, August, 1984.

Cottrell, G. W. A connectionist approach to word sense disambiguation. Unpublished Ph.D. Thesis, Dept. of Computer Science, University of Rochester, May 1985.

Cottrell, G. W. and Small S. A connectionist scheme for modelling word sense disambiguation. *Cognition and Brain Theory*, 1983, *6*, 89-120.

Feldman, Jerome A. and Dana Ballard. Connectionist Models and their Properties. *Cognitive Science*, 1982, *6* 205-254.

Fillmore, C.J. The case for case. In Bach and Harms (Eds.), *Universals in Linguistic Theory*. Holt, Rinehart and Winston, 1968.

Frazier, Lyn. On comprehending sentences: Syntactic parsing strategies. Ph.D. Thesis, University of Connecticut, 1978.

Gigley, H. M. Neurolinguistically Constrained Simulation of Sentence Comprehension: Integrating Artificial Intelligence and Brain Theory. Ph.D. Thesis, Department of Computer and Information Science, University of Massachusetts, September 1982.

Hinton, G.E. and T. Sejnowski. Analyzing cooperative computation. In *Proceedings of the Fifth Annual Cognitive Science Society Conference*, Rochester, N.Y., May 1983.

Hudson, S. & Tanenhaus, M. Ambiguity resolution in the absence of contextual bias. In *Proceedings of the Sixth Annual Conference of the Cognitive Science Society*, Boulder, Colorado, June, 1984.

Lesser, V.R., and Erman, L. D. A retrospective view of the Hearsay-II architecture. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 1977.

Linebarger, M.C., Schwartz, M.C., and Saffran, E.M. Sensitivity to grammatical structure in so-called agrammatic aphasics. *Cognition*, 1983, *13*, 361-392.

McClelland, J. L. Putting knowledge in its place: A scheme for programming parallel processing structures on the fly. *Cognitive Science, 9*, 113-146, 1985.

McClelland, James L. and David E. Rumelhart. An interactive activation model of the effect of context in perception: Part I, An account of basic findings. *Psych. Review, 88*,

Pollack, Jordan and Waltz, David. Natural language processing using spreading activation and lateral inhibition. In *Proceedings of the Fourth Annual Conference of the Cognitive Science Society*, Ann Arbor, Michigan, August, 1982.

Pollack, Jordan and Waltz, David. Massively parallel parsing: A strongly interactive model of natural language interpretation. *Cognitive Science, 9*, 51-74, 1985.

Rayner, K. Carlson, M. and Frazier, L. The interaction of syntax and semantics during sentence processing: Eye movements in the analysis of semantically biased sentences. *Journal of Verbal Learning and Verbal Behavior, 22*, 358-374, 1983.

Seidenberg, M. S., Tanenhaus M., Leiman, J. and Bienkowski, M. Automatic access of the meanings of ambiguous words in context: Some limitations of knowledge-based processing. *Cognitive Psychology*, 1982, *14*, 489-537.

Selman, Bart and Hirst, Graeme. A rule-based connectionist parsing system. In *Proceedings of the Seventh Annual Conference of the Cognitive Science Society*, Irvine, California, 1985.

Seidenberg, M. S., Tanenhaus M., Leiman, J. and Bienkowski, M. Automatic access of the meanings of ambiguous words in context: Some limitations of knowledge-based processing. *Cognitive Psychology*, 1982, *14*, 489-537.

Small, S. L., Cottrell, G.W., and Shastri L. Toward Connectionist Parsing. *Proceedings of the National Conference on Artificial Intelligence*, Piitsburgh, Pa., August 1982.

Small, S. L., Shastri L., Brucks M., Kaufman S., Cottrell, G., and Addanki, S. ISCON: An Interactive Simulator For Connectionist Networks. Technical Report 109, Department of Computer Science, University of Rochester, Dec. 1982.