Connectionist Syntactic Parsing Using Temporal Variable Binding*

James Henderson
Department of Computer Science
University of Pennsylvania

Abstract

Recent developments in connectionist architectures for symbolic computation have made it possible to investigate parsing in a connectionist network while still taking advantage of the large body of work on parsing in symbolic frameworks. The work discussed here investigates syntactic parsing in the temporal synchrony variable binding model of symbolic computation in a connectionist network. This computational architecture solves the basic problem with previous connectionist architectures, while keeping their advantages. However, the architecture does have some limitations, which impose constraints on parsing in this architecture. Despite these constraints, the architecture is computationally adequate for syntactic parsing. In addition, the constraints make some significant linguistic predictions. These arguments are made using a specific parsing model. The extensive use of partial descriptions of phrase structure trees is crucial to the ability of this model to recover the syntactic structure of sentences within the constraints imposed by the architecture.

1 Introduction

The ability of connectionist networks to learn and to combine multiple sources of soft constraints has made them important tools for cognitive modeling. On the other hand, their inability to dynamically manipulate complex compositional representations has prevented them from being successfully applied to many problems. Recovering the syntactic structure of natural language sentences requires both these abilities. Recent work on how to support symbolic computation within a connectionist computational architecture has made the combination of these abilities possible, but these architectures have limitations. This article discusses one such computational architecture, proposed by Shastri and Ajjanagadde (Shastri, Ajjanagadde, 1993), and the implications of its limitations for syntactic parsing. These limitations do not prevent syntactic parsing, and they make some significant linguistic predictions.

Like other purely connectionist architectures, the Shastri and Ajjanagadde (S&A) architecture uses many simple computing units that communicate with each other using only an output activation value. The pattern of activation over these units represents the predications that are being stored, and the interconnection pattern implements the rules of the system. Like recurrent connectionist networks, these rules may compute sequentially in time, and the order of input items is represented by presenting the input sequentially in time. Such networks have been used to parse simple syntactic constructions (Elman, 1991) and to model the interaction of syntactic and semantic constraints (St. John, McClelland, 1992), but because they are unable to capture all the relevant generalizations, they have been unable to handle the full diversity and complexity of natural language syntax. In particular, these networks cannot capture generalizations over phrase structure

*This paper will appear in the Journal of Psycholinguistic Research, probably volume 23, number 6, 1994.

constituents. For example, a rule which allows a subject noun phrase to be modified cannot also be used to modify an object noun phrase, or else the sentence-level semantic effects of these two events would be identical. To capture these generalizations, an architecture needs to represent constituent identity in a way that allows each rule to apply to each constituent, rather than to the phrase structure as a whole. The Shastri and Ajjanagadde architecture solves this problem by using fine grained temporal distinctions to represent different constituents. If units are firing synchronously, then they are representing features of the same constituent, and otherwise not. By using temporal distinctions (rather than unit distinctions) to represent constituent identity, rules inherently generalize across constituents, since the same interconnection pattern is present at every time. The addition of this temporal synchrony variable binding mechanism makes the S&A architecture computationally adequate for syntactic parsing.

While the S&A architecture solves the basic problem with connectionist networks and still keeps the advantages that have made connectionist networks attractive for cognitive modeling, it does have some limitations. These limitations impose computational constraints on syntactic parsing in this architecture. Interestingly, most of these constraints have previously been proposed based on linguistic and psychological data. The parser must have a bounded memory (Chomsky, 1959), and in particular can only store information about a bounded number of things (Miller, 1956). The parser cannot explicitly represent disjunction, which in conjunction with the requirement that the parser's output be incrementally interpretable, means the parser must be deterministic in the sense proposed by Marcus (Marcus, 1980). Also, rules implemented in the network can only test and modify information about a single phrase structure node and the phrase structure tree as a whole. This locality constraint prevents rules which manipulate pairs of nodes, which has some significant linguistic implications that have not previously been investigated.

Despite the computational constraints imposed by the S&A architecture, the architecture is powerful enough for syntactic parsing. This argument is made using a specific parsing model which has been implemented in the architecture and tested on a broad range of natural language phenomena. Following Description Theory (Marcus, *et al.*, 1983), this parsing model uses partial descriptions of phrase structure trees to allow deterministic parsing. Partial descriptions allow some kinds of information to be specified independently of other kinds of information, thereby allowing the parser to state information which it can be sure of without stating information which it can't be sure of. The partial descriptions used here allow multiple kinds of grammatical features, expectations, iteration restrictions, and structural constraints to all be specified independently of each other. All but the last of these kinds of information are local to individual phrase structure nodes, thereby isolating the information which is difficult to handle given the architecture's strict locality constraint on rules. This locality of information is also important for dealing with the parser's bounded memory, since it allows individual nodes to be removed from the memory without interfering with computations that involve other nodes.

In addition to allowing acceptable sentences to be parsed, the computational constraints imposed by the S&A architecture predict the unacceptability of some sentences. These predictions are mostly in the areas of long distance dependencies and center embedding, and are largely due to the techniques used to comply with the locality constraint on rules. Because this article concentrates on the bounded memory and determinism constraints, these results will only be outlined here.

2 The Connectionist Architecture

The Shastri and Ajjanagadde connectionist computational architecture has several characteristics which make it well suited for investigating natural language parsing. As argued in (Shastri, Aj-

janagadde, 1993), the architecture is biologically motivated, supports the massively parallel use of knowledge, supports evidential reasoning, has psychologically plausible limitations, and supports symbolic computation. All of these characteristics are important for cognitive modeling, and the relationship between the architecture and biology shows particular promise for the integration of lower level and higher level investigations of cognitive processes. However, for our purposes it is the support of symbolic computation that is most important. This property makes it possible to investigate syntactic parsing in the S&A architecture at an appropriate level of abstraction, which allows this investigation to make use of previous work on the nature of the language comprehension process.

To support symbolic computation, it must be possible to represent, and compute with, multiple properties of multiple things. Such a representation must have a mechanism for distinguishing which properties are for which things. This is called the variable binding problem. For example, to represent the situation at the top left of figure 1, we need to represent that the square is striped and the triangle is spotted. This can be done with the following logical formula.

$$\exists x, \exists y, \text{striped}(x) \wedge \text{square}(x) \wedge \text{spotted}(y) \wedge \text{triangle}(y)$$

In this formula, the variables are used to represent the bindings between predications. The name "x" does not mean anything in and of itself, but the sharing of it represents that the thing which is striped is the same as the thing which is square, and possibly different from the thing which is spotted and a triangle. This information can be represented in the S&A architecture as shown in the rest of figure 1. As in many connectionist architectures, different predicates are represented with different units.[1] The pattern of activation over these units represents the predications which are true (or the probability of their truth). The problem with this simple representation is that there is no representation of which predicates are true of which thing. To represent the depicted situation, all four units would have to be active, but then we would not know whether it is the square or the triangle which is striped. The S&A architecture solves this problem by using units which, rather than producing sustained output, produce a pulse train of activation (as do neurons).[2] If two units are pulsing synchronously, then they are representing predications about the same thing, and if they are not pulsing synchronously, then they are representing predications about possibly different things. Thus the temporal synchrony of unit activation is used to represent the bindings between predications, just as variables are used to do this in logical formulae. This mechanism is called temporal synchrony variable binding, and it is the core feature of the S&A architecture. For the purposes of this investigation I will be assuming that these units all fire at the same frequency, so temporal synchrony reduces to having the same phase in the periodic pattern of activation. These phases, then, are equivalent to variables, as shown in figure 1. In the parsing model discussed below, variables refer to phrase structure constituents, so these phases represent constituent identities.

As in other connectionist architectures, computation in the S&A architecture is done using links between units. Links multiply the output of their input unit by their weight to get their activation. Some links provide this activation as input to another unit, where it is summed with the activation from other input links. The S&A architecture also allows links which use their activation to inhibit the activation of another link. A primary link's activation is multiplied by one minus the activation of each inhibiting link.[3] Sets of interconnected links are used to implement

[1] To prevent confusion, I will refer to nodes in a connectionist network as "units", and nodes in a phrase structure tree as "nodes".

[2] There are other kinds of units which do produce sustained output. These units represent predications about the situation as a whole, rather than information about individual entities.

[3] The use of inhibitory links is necessary to implement signal gates that don't introduce significant propagation delays, and to allow dynamically calculated probabilities to be multiplied.

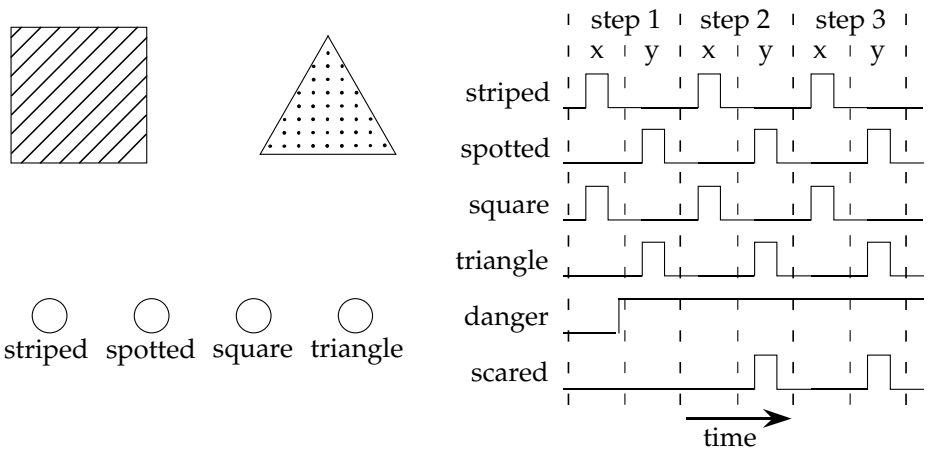


Figure 1: An example of temporal synchrony variable binding, and the sequential application of two rules.

pattern-action rules, which test and modify the predications stored in the temporal pattern of activation of the units. Because the same links are present during each variable's phase, these pattern-action rules inherently generalize across variables. Thus the rules of the parser inherently generalize across phrase structure constituents. It is this ability to capture generalizations which distinguishes the S&A architecture from other solutions to the variable binding problem, such as (Smolensky, 1990).

An example of computation in the S&A architecture is also given in figure 1. The bottom two lines of the timing diagram show the effects of the application of the following two rules.

$$\begin{aligned} \forall x, \mathit{striped}(x) \wedge \mathit{square}(x) &\Rightarrow \mathit{danger} \\ \forall x, \mathit{danger} \wedge \mathit{triangle}(x) &\Rightarrow \mathit{scared}(x) \end{aligned}$$

First the synchronous activation of *striped* and *square* trigger the activation of the unit representing the predicate *danger*, which is true of the situation as a whole. Then in y's phase the simultaneous activation of *danger* and *triangle* cause the *scared* unit to change its state. This change shows up in the output activation of the *scared* unit in the subsequent period, in the phase of y. Because the effects of a rule generally show up in the pattern of activation one period after the rule applies, periods can be thought of as steps in the computation, as shown in figure 1. By supporting predications over variables and supporting sequential computation with pattern-action rules that generalize over variables, the S&A architecture supports the kind of symbolic computation necessary for syntactic parsing.

3 The Computational Constraints

As discussed above, this article argues for the computational adequacy and linguistic significance of the Shastri and Ajjanagadde connectionist computational architecture for syntactic parsing. While these arguments involve the development of a specific parsing model which is implemented in the primitive computational devices of the architecture, that level of description is too detailed to provide a useful basis for discussing the motivations for, and implications of, aspects of the parser's design. Fortunately, computation in this architecture has been characterized in terms of symbolic computation, which has been found to be an appropriate level of abstraction for this

type of investigation. This prevents the irrelevant details of the architecture from interfering with the investigation of parsing issues.[4] The characteristics of the architecture which are relevant for parsing can be characterized at the level of symbolic computation in terms of a set of computational constraints. These constraints, plus the constraints from the nature of the parsing task, form the set of computational constraints on a model of syntactic parsing in the S&A architecture.

3.1 Constraints from the Architecture

While the S&A architecture provides a rather general purpose computing framework, it does have significant limitations. One very significant limitation of this architecture is that it has a bounded memory capacity.[5] The ability to detect whether or not units are pulsing synchronously has bounded precision, and units can only maintain periodic firing for a bounded range of frequencies, so only a bounded number of distinguishable phases can fit in one period. Thus predications can only be stored for a bounded number of variables. From biological evidence this bound is at most ten, probably a little less (Shastri, Ajjanagadde, 1993). For this investigation the bound will be assumed to be ten.

Another significant limitation of the S&A architecture is that it has no explicit representation of logical connectives. Thus only the default logical connective can be used. Conjunction is the most useful connective for syntactic parsing, so it will be used as the default connective. This means the architecture cannot explicitly represent a disjunction of predications. However, it can implicitly represent disjunctive information, and disjunction can be manifested in parallel computations. The rules of the network or an external observer can give a disjunctive interpretation to a predicate, but in terms of the explicit representation, the predications will still simply be conjoined with other predications. Also, the lack of a predication can be interpreted as a disjunction between that predication and other incompatible predications, but again the predications which are specified are treated as conjoined. The parallel computation of pattern-action rules can also manifest disjunction, in that different patterns can be doing tests which pertain to different possible continuations of the computation. Thus if a bounded amount of disjunction is needed for a bounded amount of time, it can be eliminated by compiling all that computation into one pattern-action rule for each of the disjuncts. These computations, however, must be atomic, in that they cannot store intermediate state, and thus cannot be composed of multiple steps. This limits the feasibility of this method to only short computations, although there can be a large number of disjuncts.

The last limitation of the S&A architecture is its locality constraint on rules.[6] Because links do not have any memory, the rules implemented by links can only test and modify information which is represented in the activation pattern at a given instant. This includes information about an individual variable, and information about the situation as a whole. Thus this is the only information that can be tested or modified by a given rule. Computations which involve multiple variables can be done by using different rules to set and test predications about the situation as a

[4] This approach represents a departure from standard connectionist methodology. Even if the reader is not convinced by the above argument, hopefully they will find the results of this investigation sufficient to justify this divergence.

[5] The architecture allows for multiple computing modules, each with its own memory. NNEP is implemented as one of these modules. The memory bounds apply to each computing module independently, so the parser does not have to share its memory resources with other cognitive activities.

[6] There is actually one more constraint imposed by the S&A architecture, but this constraint ends up being subsumed by the bounds on the data structures used to handle the locality constraint on rules. This other constraint limits the storage of relationships between variables, and is related to the way the locality constraint on rules limits the processing of relationships between variables. See (Shastri, Ajjanagadde, 1993) for a discussion of this constraint, and (Henderson, 1994) for a discussion of how it applies to the work presented here.

whole (as was illustrated in section 2), but computations which require the manipulation of pairs of variables (or triples, etc.) cannot be directly implemented. However, they can be indirectly implemented if at any given time the relationships represented by the pairs can also be represented with unary predicates. This can be guaranteed if at any given time one of the variables in each pair can be uniquely identified. Then the variables which are in the relationship to this unique node can be specified with a unary predicate, and the computation can be done using these unary predicates. For example, calculating long distance dependencies requires calculating relationships between a trace node and other nodes, but because (as it turns out) only one trace node needs to be involved in these computations at any given time, these calculations can be done within the locality constraint on rules. If a node is uniquely identifiable, then a rule can refer to it with a constant rather than a variable, so this locality constraint on rules can be expressed as a constraint that no rules involve more than one variable. While this constraint has no precedent in work on syntactic parsing, it turns out to have a number of significant linguistic implications.

3.2 Constraints from the Task

To understand the implications of the computational constraints imposed by the S&A architecture, we need to take into consideration the computational constraints imposed by the nature of the syntactic parsing task. The words which are input to the parser become available one at a time, in the order in which they appear in the sentence. Thus the parser must accept incremental input. The modules which receive the output of the parser need to compute the sentence's interpretation incrementally. In order to provide for incremental interpretation, the parser's output must be incremental and monotonic. If the output isn't monotonic, then the interpreter can't make commitments on the basis of the output without risking having to retract those commitments. While such retractions do occur under some circumstances, I assume that there is always some evidence that something has gone wrong. Typically the person will be consciously aware of a problem, although other evidence (such as regressions in eye movements) can also be used to determine these cases. Since we are concerned here with the normal case in which nothing goes wrong, the parser's output must be monotonic.[7] Thus the nature of the syntactic parsing process requires that a parsing model accept incremental input, and produce incremental monotonic output.

3.3 The Relationship to Previously Proposed Constraints

Combining the constraints from the nature of the parsing task with the constraints from the architecture, we get the following set of constraints on a model of syntactic parsing in the S&A architecture. In additional to being consequences of using an independently motivated computational architecture, these computational constraints are interesting because of their relation to previously proposed computational constraints on natural language.

1. at most ten variables stored at a time
2. no explicit representation of disjunction
3. rules can only use one variable
4. incremental input
5. incremental output
6. monotonic output

[7] Note that this constraint is only being claimed to apply to the calculation of syntactic constituent structure. Other levels of representation, such as predicate-argument structure, may not be subject to the monotonicity constraint.

The first constraint is an example of a bounded memory requirement. It has generally been assumed that at some level of abstraction the syntactic parser has a bounded memory (Chomsky, 1959). Church (Church, 1980) showed that this constraint applies at a level which takes into consideration performance constraints, such as restrictions on the depth of center embedding and on the availability of phrases for posthead modification. The particular form of the bounded memory constraint given above has not previously been successfully applied to syntactic parsing, but it has extensive precedence in other investigations of cognition. Miller proposed a bound of seven plus or minus two on the number of things which can be stored in short term memory (Miller, 1956), and this result has been replicated for a surprising number of tasks. The bound given above is precisely the same form of constraint, and although here I'm assuming ten things can be stored, Miller's results are within the resolution of the biological arguments which were used to derive that bound. See (Shastri, Ajjanagadde, 1993) for a more extensive discussion of this relationship.

Another interesting correlation with previously proposed computational constraints on natural language is due to the restriction on disjunction and the requirement for incremental monotonic output. These constraints imply that the syntactic parser must parse deterministically. This constraint was first proposed by Marcus (Marcus, 1980), and has been argued for by several researchers since ((Church 1980), (Marcus, *et.al.* 1983), (Berwick, Weinberg, 1984)). It requires that the parser deterministically pursue a single analysis. This means that multiple analyses can't be pursued in parallel, and that once the parser commits to an aspect of the analysis it can't retract that commitment. Explicitly pursuing multiple analyses in parallel is equivalent to having explicit disjunction in the representation of the analysis, which is ruled out by the second constraint above. The retraction of commitments is ruled out because all commitments must be immediately output in order for the output to be maximally incremental, and once information has been output it can't be retracted or the output wouldn't be monotonic. Thus the determinism constraint can be derived from the independently motivated constraints that there be no explicit representation of disjunction and that the parser's output be incremental and monotonic.

Because in this article I am emphasizing the way investigations using the S&A architecture fit with other work in psycholinguistics, the following discussion will concentrate on the implications of the bounded memory and determinism requirements. The locality constraint on rules also has significant consequences, and these consequences will be mentioned, but they will not be the focus of discussion. See (Henderson, 1994) for an extensive discussion of these issues.

4 The Parsing Model

The argument for the adequacy and linguistic significance of the Shastri and Ajjanagadde connectionist architecture is made using a specific example of a parser implemented in this architecture. The previous section identified the characteristics of this architecture which are significant for syntactic parsing in terms of a set of constraints on symbolic computation. Given this characterization, it is possible to make the arguments for adequacy and significance at the level of symbolic computation. This greatly simplifies the discussion of the relevant characteristics of the parser, and it allows results from work in linguistics, computational linguistics, and psycholinguistics (which has almost all been done in terms of symbolic representations) to be applied to this investigation. Accordingly, this section will concentrate on how this parsing model, called a Neural-network Node Equating Parser (NNEP), is designed to comply with the computational constraints discussed in the previous section.

4.1 Representing Phrase Structure Trees

The constraints outlined in the previous section place several requirements on the parser's representation of grammatical information. First, because the parser must be deterministic, the representation should allow the parser to avoid saying what it doesn't know. Following Description Theory (Marcus, *et al.*, 1983), partial descriptions of phrase structure trees are used to satisfy this requirement. Partial descriptions allow the parser to underspecify phrase structure information, rather than either overcommitting or using a disjunction of more completely specified alternatives. In addition, in order to produce incremental output and only allow syntactically well-formed analyses, the parser must be able to say what it does know. Again the use of partial descriptions is important for this requirement, because they allow different kinds of information to be specified independently of each other. To satisfy both these requirements, the grammatical representation must allow information which the parser does know at a given time to be specified independently of the information which the parser does not know. The grammatical representation used here allows different kinds of grammatical features (e.g. +*nominative*, +*plural*), expectations (e.g. obligatory arguments), iteration restrictions (e.g. one determiner per NP), and structural constraints (e.g. linear order) to all be specified independently of each other.

The locality constraint on rules and the parser's bounded memory both place another requirement on the parser's representation of grammatical information. Because of the locality constraint on rules, the representation should allow as much information as possible to be local to individual phrase structure nodes. Thus we want a relatively flat phrase structure representation, provided it still expresses the compositional nature of syntax. This compact representation also makes it easier to stay within the parser's bounded memory, because it reduces the number nodes in a tree's representation. The grammatical representation used here allows flexibility in the grouping of information into nodes because multiple kinds of expectations and iteration restrictions can be specified for a single node. In many formalisms this is not true. For example, in Context Free Grammars, the node on the left side of a rule cannot have any more nodes attached to it (thereby restricting iteration), and the nodes on the right side of the rule must have other nodes attached to them (thereby expressing expectations). For constituents which can iterate, like optional modifiers, Chomsky adjunction needs to be used. This results in multiple copies of the modified node. Also, in order to control the iteration of things like determiners separately from controlling the iteration of things like head nouns, Context Free Grammars have to have separate nodes for these two purposes (i.e., NP and N, or DP and NP). These problems also apply to the expression of expectations. Optional arguments require two grammar rules, one with the argument and one without, and expressing the expectation for a determiner separately from expressing the expectation for a head noun requires two separate nodes for these purposes.

The locality constraint on rules and the parser's bounded memory interact in another interesting way to constrain the parser's representations. Not only should as much information as possible be local to individual nodes, as little information as possible should be expressed as relationships between nodes. Of the four kinds of information mentioned above, only structural constraints involve multiple nodes. By allowing most ordering constraints to be stated with respect to terminals (rather than other nonterminals), many structural constraints can also be localized to individual nodes.[8] By identifying the minimal set of relations that are needed to parse, special mechanisms which allow all rules to use only one variable can be devised for these few cases. This localization of computation in turn makes it possible to stay within the parser's bounded memory. Because computations which do not directly involve a node are independent of the information about that

[8]Only nonterminal nodes are represented as entities in the parser's memory. Information about terminals is represented with features and constraints on the use of grammar entries.

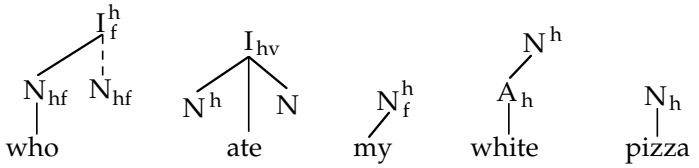
node, a node which will not be directly involved in any more parser operations can be safely removed from the parser's memory. By removing nodes as they are completed during a parse, the parser can parse arbitrarily long sentences using only a bounded number of nodes at any given time. The grammatical representation used here allow relationships between nodes to be minimized because structural constraints are specified independently of other kinds of grammatical information. Grammar formalisms based on Context Free Grammars do not have this property because expectations and iteration restrictions are specified in terms of a node's structural position in the grammar rule, as discussed above.

4.1.1 Structure Unification Grammar

In order to comply with the above requirements, NNEP uses Structure Unification Grammar (Henderson, 1990) as its grammar formalism. Structure Unification Grammar (SUG) is a formalization of accumulating partial information about the phrase structure of a sentence until a complete description of the sentence's phrase structure tree is constructed. As such it is similar to other unification based or constraint based grammar formalisms. These include Description Theory (Marcus, *et al.*, 1983), Head-Driven Phrase Structure Grammar (Pollard, Sag, 1987), Construction Grammar (Fillmore, *et al.*, 1988), and Segment Grammar (de Smedt, Kempen, 1991), among others. Like these other formalisms, SUG allows multiple kinds of grammatical features to be specified independently of each other. Unlike these other formalisms, SUG allows multiple kinds of expectations, iteration restrictions, and structural constraints to also be specified independently of each other. In addition, SUG's derivations are only constrained by the semantics of the declarative representation, so any valid parsing strategy can be characterized in terms of valid SUG derivations.

The flexibility of SUG derivations is due to its simple mechanism for combining partial descriptions of phrase structure trees. An SUG derivation takes partial descriptions from the grammar (which is simply a set of partial descriptions), conjoins them, and equates some of their nonterminal nodes. Any order of conjoining descriptions and equating nodes is possible, so the parser can use any parsing strategy and still be following an SUG derivation. The only restrictions on derivations are that the final description be consistent and completely describe some phrase structure tree. This means that each equation done in the derivation needs to be between nodes which have consistent descriptions. The grammar can limit the possible equations by specifying inconsistent information about any two nodes which shouldn't be equated. Unlike consistency, completeness is only necessary for the final description. By not satisfying completeness requirements locally, a grammar entry can express expectations about what kinds of information other grammar entries will contribute to the final phrase structure. Because of the complete flexibility of SUG derivations, SUG grammar entries have no procedural import, and the grammar is free to group information into grammar entries in a way which expresses exactly the information interdependencies which the parser needs to know.

The language which SUG provides for specifying partial descriptions of phrase structure trees is illustrated in figure 2. As in many formalisms, the grammatical features of nodes are described with feature structures. The use of feature structures allows multiple kinds of grammatical features to be specified independently of each other. Expectations and iteration restrictions are specified with a different kind of feature, shown in figures as letter superscripts and subscripts, respectively. Expectations express what information will be specified before the parse is finished. Superscripts specify these expectations in that before a parse can be finished, any node with a superscript must equate with a node that has the same letter as a subscript. For example in figure 2, the subject node for *ate* must be equated with a node which has its head noun, thereby expressing the fact that *ate* obligatory subcategorizes for a subject. The object node has no such feature, since the object



key:

x | y — x **immediately dominates** y

x ⋮ y — x **dominates** y

x --➔ y — x **precedes** y

x^a — x **needs** an a

x_a — x **has** its a

node label features:

N = [I:-,A:-] (noun phrase)
I = [I:+,A:-] (sentence)
A = [I:-,A:+] (adjective, adverb)
P = [I:+,A:+] (prepositional phrase)

superscript and subscript features:

h = **constituent_head** (noun, inflection, adjective, preposition)
f = **functional_head** (determiner, complementizer)
v = **verb**

Figure 2: Some example grammar entries. They can be combined to derive the sentence *Who ate my white pizza* by equating the two I's, the second and third N's, and the last four N's.

of *ate* is optional. Iteration restrictions prevent grammar entries from being repeatedly attached at a node, even if the grammatical features of the grammar entries are compatible. Subscripts specify these restrictions in that any node with a subscript cannot be equated with another node which has the same subscript. For example in figure 2, *my* has a subscript to prevent other determiners from attaching to the same noun phrase, while *white* has no such subscript, thereby allowing adjectives to iterate.

Like all the above features, structural constraints can be specified partially and independently of other constraints. In addition to the immediate dominance relation for specifying parent-child relationships[9] and the linear precedence relation for specifying ordering constraints,[10] SUG allows chains of immediate dominance relationships to be partially specified using the dominance relation. A dominance constraint between two nodes specifies that there must be a chain of zero or more immediate dominance relationships between the two nodes, but it does not say anything about the chain. This relation is necessary to express long distance dependencies in a single grammar entry. For example in figure 2, the grammar entry for *who* expresses the fact that its gap is somewhere within its sentence, but does not say where. Because the final description of a derivation must specify a single tree, the N "trace" node in this grammar entry must find a "gap" node to equate with, thereby expressing the fact that the existence of a gap is obligatory.

[9] In the grammar, the solid lines in figures represent immediate dominance, but when these descriptions are interpreted by NNEP, solid lines do not specify the actual identity of the immediate parent for the dominated node. The reason is that the forgetting operation to be discussed below does not allow such identity information to be kept.

[10] In order to simplify figures, linear precedence constraints will not in general be shown. Most such constraints are between words and either nonterminals or their head terminals. These can be inferred from the lateral position of the nodes relative to the words.

4.1.2 Node Closure

In addition to providing the necessary flexibility in the specification of the phrase structure of the sentence, Structure Unification Grammar localizes information in a way that allows completed nodes to be closed from further access by the syntactic parser. Closed nodes can be removed from the phrase structure representation, thereby reducing the number of nodes which NNEP needs to store information about, and allowing it to stay within its memory bounds. Because NNEP outputs all the information about the phrase structure of the sentence as it computes it, forgetting nodes does not interfere with the interpretation of the output. The mechanism for closing nodes, called the forgetting operation, does not imply any particular node closure strategy; it simply provides a sound mechanism for implementing such a strategy.

For the forgetting operation to be sound, its use cannot allow the forgotten information to be contradicted later in the parse. Because forgetting a node prevents any future parser actions at that node, soundness can be guaranteed as long as all the information about a forgotten node would only be needed to test the consistency of parser actions at that node. The only information in an SUG description which is a problem for this requirement is immediate dominance. Some parser actions need to test whether a node has an immediate parent, but if that parent has been forgotten, then this information would not be available. Since no parser actions need to know the actual identity of the immediate parent, this problem can be easily solved by representing immediate dominance in two parts, dominance (for ordering constraints), and having an immediate parent. The later information is a property of an individual node, so forgetting the parent will not interfere with accessing this information. With this change in representation, forgetting a node will never allow the parser to compute an analysis which would otherwise be impossible. It may, however, prevent the parser from finding an analysis which would otherwise have been possible. Thus the parser wants to avoid forgetting nodes which have a significant chance of being involved in a parser action. In particular, it never wants to forget nodes which must be equated with in order for the parse to be completed.

4.2 Recovering Phrase Structure Trees

The parsing model presented here (NNEP) uses SUG's phrase structure descriptions as its representation of phrase structure information, and computes SUG's derivations in recovering that phrase structure information from the words of a sentence. NNEP's parser state represents an SUG description which specifies the information that has been determined so far about the phrase structure of the sentence. NNEP's operations compute the SUG derivation steps which combine this intermediate description with descriptions from the grammar and perform node equations. NNEP outputs each of these derivation steps as they are computed, thereby outputting all the information which NNEP adds to its parser state as soon as the information is inferred. When the parse is done, NNEP checks to make sure it has produced a complete description, thereby ensuring that NNEP will only accept sentences which the grammar specifies as grammatical.

The set of SUG derivations which NNEP can compute is limited by the computational constraints discussed in section 3. Because NNEP must produce incremental output, the phrase structure information which is implied by the presence of a word must be added to the parser state (and therefore output) when the word is input. This information is precisely the grammar entry for the word, provided there is no lexical ambiguity. If there is more than one grammar entry that could be used for a word, then because no disjunction is allowed in the parser state, one of them must be chosen.[11] In some cases this forced choice can result in a mistake, thereby predicting a

[11] It is possible that predicates could be defined which represent a bounded disjunction between grammar entries,

garden path.[12] The parse shown in figure 3 gives examples of three parser operations which add the information in a grammar entry to the parser state.

In contrast to grammar entries, the equations between nodes in the grammar entry and nodes already in the parser state do not necessarily have to be specified. For example in figure 3, when the grammar entry for *who* is added, its root is equated with the sentence node which initialized the parser state, but when *the* is processed, it's grammar entry is not attached to the tree fragment from the previous portion of the sentence. Such delays in attachment decisions are necessary when there is not enough information available at that time to make a commitment to one equation, since the determinism requirement prevents the retraction of commitments. In this case, NNEP can't be sure that *the* is the start of the object of *ate*, since it might also be the start of the possessor of the object of *ate*, as in *Who ate the pizza's crust.* If an equation decision is delayed, one of the possible equations can be performed later in the parse when there is enough information available. In this example, the equation is done when the end of the sentence is reached, at which point there can be no forthcoming possessive marker. After this equation, NNEP has specified a single immediate dominance tree, and there are no remaining superscripts, so the parse has been completed successfully.

The locality constraint on rules means that the rules which implement the parser's operations must each involve only one variable. This constraint limits the set of operations that NNEP can use. This has its greatest effect on the process of recovering long distance dependencies, which requires the calculation of what constituents a trace node might be equated with or extracted out of. To allow the calculation of these relationships, trace nodes are placed on a bounded stack, and the rules which do these calculations are restricted to only apply to the top node on this stack. The top node on the stack is called the public node. Because the public node is always unique, potential equation sites and potentially dominating nodes for the public node can be accessed by rules using unary predicates. In figure 3, the trace node introduced by *who* is the public node when *ate* is being processed. This allows a single operation to simultaneously equate the trace node as the subject of *ate*, and equate the sentence node of *ate* with the sentence node introduced by *who*.

The bound on the number of variables that can be stored in NNEP's memory requires the use of the forgetting operation discussed above. In figure 3, after *ate* is processed, the two NP's on the left are no longer on the right frontier of the sentence. Thus no other nodes will be equated with them, and NNEP can safely close them off from further consideration. Since this level of representation is only being used for syntactic parsing, forgetting these nodes does not interfere with processes which might involve their associated nodes at other levels of processing. The resulting parser state only requires two variables. The terminals are only shown for readability.

4.3 The Connectionist Implementation

As discussed above, NNEP is implemented using the Shastri and Ajjanagadde connectionist computational architecture. The S&A connectionist architecture was developed for modeling fast common sense reasoning (called reflexive reasoning), and here it is used to implement a special purpose module for syntactic constituent structure parsing. It is a module in that the predicates and variable

or portions of grammar entries, thus allowing lexical disambiguation to be delayed. However, this would greatly complicate the parser, since such predicates would require a very complex interpretation which is rather different from the node-local features represented by most other predicates. Thus this alternative has not been pursued, although perhaps the constrained use of such predicates would be feasible.

[12] This discussion is a slight simplification. In the complete model (Henderson, 1994), the parser can wait for information about the immediately following word in cases where it isn't sure which grammar entry to pick. Also, not all grammar entries are associated with words, so some ambiguities can be handled by delaying the addition of one of these nonlexical grammar entries.

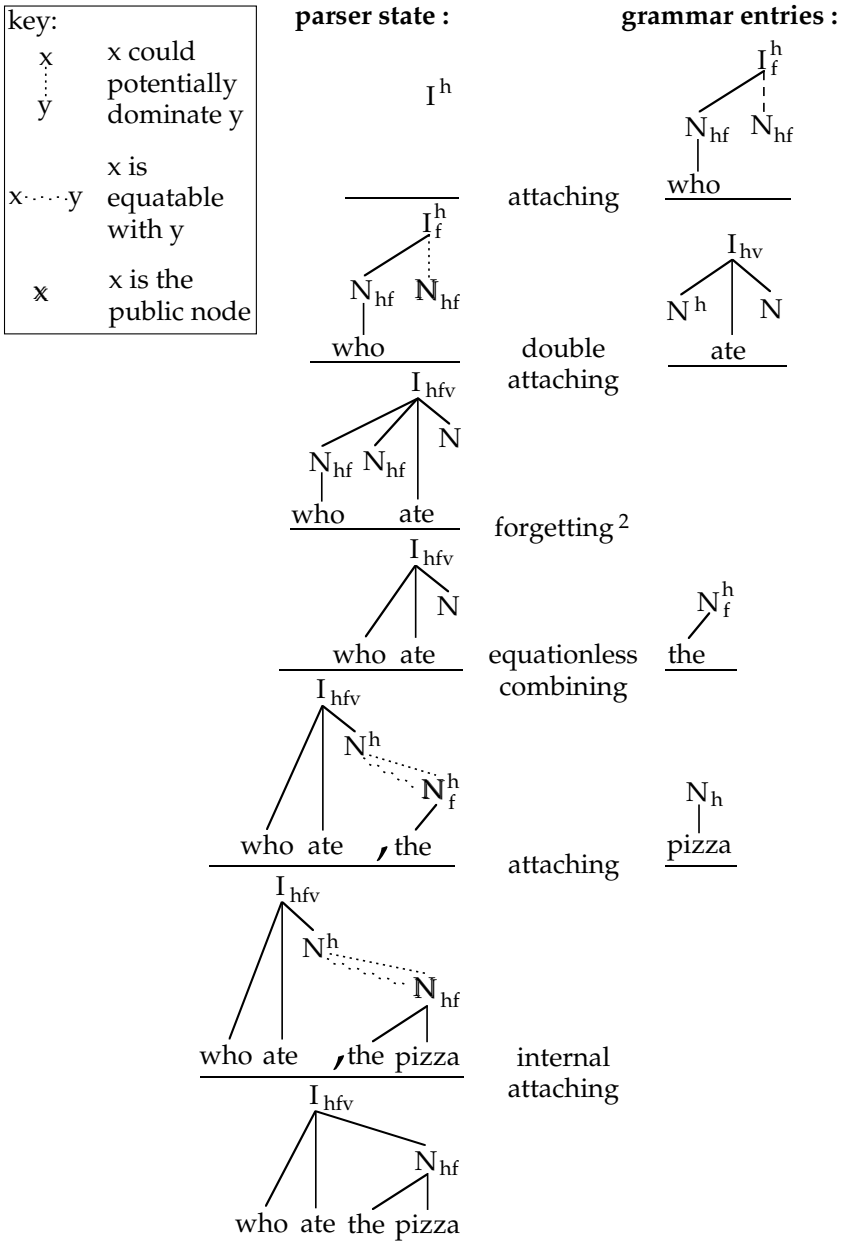


Figure 3: An example parse of *Who ate the pizza.*

bindings of the network are specific to the network's particular task. It is not currently clear whether this network is part of a larger module which includes things such as the calculation of predicate-argument structure, or whether it is a distinct module that interacts highly with other stages of language processing.

The network has three basic parts, input units, predicate units, and grammar units. The temporal pattern of activation over the predicate units represents the information the parser needs to know about its parser state. The phases in this pattern of activation represent variables that refer to nonterminal nodes in the phrase structure of the sentence, and the predicates represent properties of nodes and of the phrase structure tree as a whole. Since these predicate units represent a feature decomposition of the types of nodes, and these features are only interpreted within this module, the predicate units are analogous to hidden units in Parallel Distributed Processing (Rumelhardt, McClelland, 1986) networks.

The network's input units are just a stand-in for the word recognition component of the system. There is one input unit per word. The unit for the next word is active across all phases until a grammar entry for that word is combined with the parser state. Links from these units go to units for the grammar entries of the word. These primary links are inhibited by links from the units that represent predicates. These inhibitory links filter out all the phases for phrase structure nodes in the parser state which cannot be sites for a combination with the grammar entry. Thus primary links from the input units and inhibitory links from the predicate units implement the patterns for the pattern-action rules that calculate what action the parser should take given the parser state and the next input word. Other such links implement the patterns for parser actions that do not use a grammar entry (such as internal attaching).

The input activation provided to the grammar units by the above patterns is used to choose what parser action to perform next. The nature of the arbitration network that should be used to make this disambiguation decision has not been significantly addressed in this work, but see (Stevenson, 1994) for an investigation of these issues. Once this choice has been made, the grammar unit for the chosen parser action fires in the phase of the chosen site. This is the output of the parser, since this is the earliest indication of what the parser has decided, and the sequence of parser actions completely determines the information that the parser recovers about the phrase structure of the sentence. This output is also used to trigger the action component of the pattern-action rule that calculates the chosen parser action. This action changes the states of the predicate units to reflect the new information that is implied by the chosen parser action. An action is implemented with links from the grammar unit to the units for new predications about the site of the parser action, plus a unit that gates (using inhibitory links) links to units for the new predications about other nodes in the grammar entry. These gated links may introduce new nodes into the parser state, and may add information about uniquely identifiable nodes. The forgetting operation, which removes nodes from the parser state, is implemented with a pattern-action rule that looks for nodes which have at most a small chance of ever being involved in any future parser actions, suppresses all the predications about these nodes, and makes their phases available for future use.

In addition to the pattern-action rules for parser actions, there are some rules for calculating the indirect implications of the predications that are directly added by the above rule actions. These are implemented with links that propagate activation from the directly set predicate units to the units for the implied predications. These rules include those that calculate possible long distance dependencies, and rules that transfer information about the public node to predications about the structure as a whole.

5 Adequacy and Significance

The previous section described a model of syntactic parsing (NNEP) which is designed to comply with the constraints imposed by the S&A architecture and the nature of the parsing task. In this section, NNEP will be used to argue that the S&A architecture is computationally adequate for syntactic parsing, and that it makes linguistically significant predictions. To argue for the adequacy of an architecture, it is not sufficient to perform tests simply on a small set of examples, or on phenomena which the architecture is well suited for. The phenomena which are likely to be difficult for the architecture need to be identified, and empirical tests need to be performed on these phenomena. Because the limitations of the S&A architecture which are significant for syntactic parsing have been characterized at the same level of abstraction (symbolic computation) as has traditionally been used in the study of linguistic phenomena, it is fairly easy to identify the phenomena which are of particular concern. Most of this section discusses these phenomena and the empirical tests which have been performed on them. To argue that an architecture makes linguistically significant predictions, it is simply necessary to provide examples of such predictions. These results will be outlined in the discussion of the tests. See (Henderson, 1994) for an extensive discussion of these results.

While it is important to pay particular attention to phenomena where the limitations of the architecture are likely to be significant, it is also necessary to guard against errors in the identification of these phenomena. There is no complete proof that the relevant limitations of the S&A architecture and the relevant phenomena for these limitations have been completely identified. For this reason, NNEP is also tested on a set of randomly selected, naturally occurring sentences. This provides an essentially unbiased test of the parser's ability to handle the diversity of phenomena in natural language. While it is not necessary for the parser to handle every phenomena in this test set, specific arguments need to be made that any excluded phenomena can be handled by extensions to the parser.

Four types of phenomena are of particular concern given the limitations of the S&A architecture. As discussed in section 3, the parser has a bounded memory, must be deterministic, and must be implemented with rules that only use one variable. The bounded memory constraint requires testing on center embedded sentences, since these are the sentences which necessarily involve remembering a relatively large number of nodes. The determinism constraint requires testing with locally ambiguous sentences. If a given sentence prefix can be continued in more than one way, then at that point the parser needs a representation of the sentence's phrase structure which is compatible with both continuations, without using disjunction.[13] This requirement in turn places constraints on the representation of grammatical analyses. Thus we also need to test the parser's ability to express phrase structure analyses that accurately characterize the language. Computations that involve more than one node always involve nodes which are looking for an immediate parent. These nodes include trace nodes, so testing on long distance dependencies is necessary. They also include subject nodes and delayed attachment decisions, but these phenomena are adequately covered in the local ambiguity and phrase structure analysis data. The data structures which are used to comply with the locality constraint on rules also introduce constraints. These data structures are of bounded size, so they also require testing on center embedded sentences. From this analysis we

[13] Maintaining local ambiguities requires additional resources, but to date the interaction between maintaining ambiguity and resource bounds has not been investigated. For example, a long right branching sentence could conceivably have a modifier attached to any one of the nodes on the right frontier, and thus all of these nodes would need to be stored. However, it is well known that the set of nodes which are available for such modification is severely restricted (Church, 1980). Because of these performance constraints, the parser's bounded memory is not likely to be a problem for maintaining local ambiguities. No test in this area has been done because no suitable set of data has been found.

see that NNEP needs to be tested on center embedded sentences, local ambiguities, long distance dependencies, and phrase structure analyses.

To test NNEP on the specific phenomena identified above, papers were selected from the literature which discuss a representative sample of the data on these phenomena. NNEP was then tested on its ability to at least parse the data in the papers, if not adopt the same analyses.[14] For an approximately unbiased sample of sentences, a set of sentences randomly selected from the Brown Corpus was used. All of these tests deal only with English data, except to the extent that the analyses inherited from the papers generalize to other languages.

To test NNEP's ability to express phrase structure analyses that accurately characterize the language, the phrase structure analyses in (Kroch, 1989) were used. In (Kroch, 1989), constraints from Government Binding theory are expressed in the Tree Adjoining Grammar (TAG) framework. The similarity between SUG and TAG made this paper particularly appropriate for this task. As is the case in the examples given above, NNEP's grammars represent each lexical projection and all its associated functional projections as a single SUG nonterminal node. All the information about these projections are expressed in the feature structures which label the node. This compact representation is possible because SUG can represent multiple types of expectations and iteration restrictions on a single node. Ordering constraints within these projections can be specified with respect to the distinguished terminal nodes (constituent head, functional head, and verb).[15] Any schematized local collection of nodes can be represented in this way, so the test was successful.

The analyses in (Kroch, 1989) were also used to test NNEP's ability to recover long distance dependencies. Again the use of TAG in this paper was useful, because NNEP's rules for calculating long distance dependencies factor the local component of that dependency from the recursive component, just as is done in TAG. The local rule calculates what constituents could be the gap for a filler, and the recursive rule calculates what constituents could have the gap somewhere within them. Because the relevant filler is always uniquely identifiable as the public node, these rules only need a variable to range over the candidate nodes, and thus do not have to violate the locality constraint on rules. Most of the constraints on long distance dependencies are simply compiled into features on nodes in grammar entries (i.e., *extractable* and *not_barrier*), and enforced by the long distance dependency rules. However, some of the constraints are enforced by the computational constraints on these rules. In particular, these rules can only access the most recently introduced trace node (i.e., the top node on the public node stack). This constraint is used to explain the that-trace effect, the cases of subject islands that precede inflection, and the limited possible extractions out of wh- islands. The later phenomena are particularly interesting, because accounting for this data required Kroch to go outside the power of TAG. Thus by accounting for this phenomena with a computational constraint, the competence theory of long distance dependencies can be simplified. This explanation for wh- island constraints is also interesting in that it subsumes Pesetsky's path containment condition (Pesetsky, 1982). In summary, all the data in (Kroch, 1989) was correctly categorized, mostly by adopting the same analyses, and some of the phenomena were predicted by the computational constraints imposed by the S&A architecture.

To test NNEP's ability to handle local ambiguities, the data from the chapters on ambiguity resolution in (Gibson, 1991) were used. (Gibson, 1991) is particularly well suited for this purpose because Gibson surveys the literature on ambiguity resolution and discusses the relevant data. To

[14] While each of these areas deserve a more detailed analysis, a broad and shallow test is appropriate for this stage of the investigation. A demonstration of the feasibility of addressing all of these issues is necessary to justify the detailed investigation of any one of them.

[15] Ordering constraints involving subjects are enforced by limiting the parser operations that can be used with the grammar entry, since the presence of a subject in the parser state changes how the subject's sentence can be attached to.

test whether the S&A architecture's determinism constraint prevents a parser from being adequate, only NNEP's ability to represent the ambiguities in this data set needs to be addressed. NNEP needs representations which allow it to delay the resolution of a local ambiguity long enough for disambiguating information to be found. The issue of whether the parser can make the right choice given disambiguating information is not of particular concern here, given the general success of connectionist networks in disambiguation tasks. Because NNEP can delay attachment decisions, ambiguities in the way two grammar entries are connected can be handled. Because SUG's partial descriptions allow NNEP to avoid specifying information which it doesn't yet know, NNEP can use grammar entries which are compatible with all the possible continuations of a locally ambiguous sentence. This may involve leaving some structure unspecified. For example, the ambiguity between a sentential complement and a relative clause requires that the relative clause's modification relationship and trace node not be specified until the gap is found. This can be handled with grammar entries which specify the delayed structure information, but which are not associated with a word. The addition of such nonlexical grammar entries can be delayed until there is disambiguating information. Because SUG's partial descriptions allow NNEP to specify the information it does know independently of the information it needs to leave unspecified, the information which is necessary to resolve an ambiguity is available for decision making. In a few cases the information which is available for decision making also needs to include the word immediately following the current word. If the parser has to make a decision and it can't decide based on the left context and the next two words, then a garden path is predicted in one of the alternatives. There is one pair of sentences for which this prediction may be a problem, given below. Since *found* is obligatorily transitive, looking at the immediately following word to see if it could be the start of an object would ordinarily allows this reduced relative/main verb ambiguity to be resolved, but because of the heavy NP shift, this is not possible for these sentences. Gibson (personal communication) agrees that experiments are needed to determine whether one of the sentences is a garden path, as this model predicts. With this one caveat, all the acceptable data is parsable. NNEP accounts for the unacceptability of *The horse raced past the barn fell*, but otherwise no attempt was made to account for the unacceptable data. Some of the unacceptable data is probably due to the disambiguation mechanism's efforts to conserve resources, but this possibility has not been investigated.

(247a) The bird found in the room was dead.
(249a) The bird found in the room enough debris to build a nest.

The test of NNEP's ability to handle center embedded sentences used the data from the chapters on processing overload in (Gibson, 1991). Again, (Gibson, 1991) is particularly well suited for this purpose because it surveys the literature. In addition, some example sentences involving nested ditransitive verbs were constructed and used. Since the interaction between ambiguity and resource requirements is not being tested here, nodes were closed as soon as possible, using the forgetting operation. None of the acceptable sentences required more than ten nonterminals to be stored at any one time, so the architecture's bounded memory was not a problem. In fact, the maximum number of nonterminals required was nine, given the compact phrase structure representation used here. This is interesting because nine is the maximum of the robust bound on human short term memory of seven plus or minus two (Miller, 1956). The data structures which are used to handle the locality constraint on rules also result in some bounds. The public node stack can be at most two deep, there can be at most three unattached tree fragments in the parser state, and there can be at most one first posthead argument node for a ditransitive verb. None of these constraints need to be violated to parse any of the acceptable sentences in this data set. In addition, much of the unacceptable data is ruled out, mostly due to the bound on the depth of the public node stack and a particular (independently motivated) strategy for when to specify a tree fragment root as the

public node. Not all the unacceptable data, however, is predicted by these constraints. Some of this data is probably due to interactions between the resources necessary for maintaining ambiguities and these resource bounds, but this possibility has not been investigated.

To make sure the above phenomena-specific tests did not miss anything which would be difficult for the S&A architecture, NNEP was also tested on an essentially unbiased sample of sentences. This set of fifty thirteen word sentences were randomly selected from the Brown corpus by Ezra Black in 1991. Since the issue being addressed is the adequacy of the S&A architecture, and not the adequacy of NNEP as it is currently designed, incompleteness in NNEP's coverage of the phenomena in this data set is only a problem to the extent that extensions to NNEP aren't likely to be able to handle the phenomena. Indeed there are some phenomena which NNEP is not yet equipped to handle, but none of these are expected to be any more difficult for this architecture than they are in general. In particular, NNEP cannot parse coordinations, or gapping in comparatives, and it cannot make many disambiguation decisions. As argued above, this architecture is well suited for doing disambiguation. I expect that the relationship between SUG and Combinatory Categorial Grammar (Steedman, 1987) will make the analyses of coordination and gapping easier for this parsing model than for most phrase structure based parsers. Due to the scope of these topics, they will have to be left for future work.

6 Conclusion

This article has discussed syntactic parsing using a model of symbolic computation in a connectionist network recently proposed by Shastri and Ajjanagadde (Shastri, Ajjanagadde, 1993). This connectionist model of computation extends previous connectionist architectures by using temporal synchrony variable binding to represent the identities of entities in a way that allows rules to generalize over entities. Because of this added ability, the architecture can take advantage of the compositional nature of natural language, while keeping the properties of connectionist networks which have made them important tools for cognitive modeling. However, the S&A architecture has some limitations, which impose computational constraints on syntactic parsing. Most of these constraints have previously been proposed on the basis of linguistic and psychological evidence, and the other constraint makes some significant linguistic predictions. None of these constraints prevent the architecture from being computationally adequate for syntactic parsing.

To demonstrate the computational adequacy and linguistic significance of the S&A architecture for syntactic parsing, a specific parsing model has been implemented in the architecture which is designed to address the architecture's computational constraints. The central characteristic of this parser which allows it to comply with these constraints is its extensive use of partial descriptions of phrase structure trees. This parser has been tested on all the phenomena which are of particular concern given the limitations of the architecture (phrase structure analyses, long distance dependencies, local ambiguities, and center embedding). The results of these tests and a test on a random sample of sentences indicate that the S&A architecture is powerful enough for recovering the syntactic structure of natural language sentences, and that the computational constraints imposed by the architecture make some significant linguistic predictions. These predictions are mostly in the areas of long distance dependencies and center embedding.

The significance of this work goes beyond the specific issues addressed here. Because the primary concern here is the adequacy of the S&A architecture, this work has concentrated on problems which connectionist networks have previously had difficulty with. By demonstrating the feasibility of syntactic parsing in this architecture, this work justifies using it to investigate issues for which connectionist networks are particularly well suited. For example, previous connectionist investi-

gations of grammar learning and ambiguity resolution have been hampered by representational inadequacies. With the above work, the success connectionist models have had on similar problems in other areas is likely to be repeated for natural language. Another strength of connectionist architectures such as this one is their relationship to the biological substrate of human language processing. The biological motivations for the S&A architecture have already been important in this investigation, in that they provide independent motivations for the computational constraints, and therefore give them explanatory power. This relationship also gives work in this architecture predictive power, since the link between an abstract model and real time and space data can be made on the basis of independent biological evidence. The model presented above is compatible with the real time constraints on language processing given this relationship, but work taking full advantage of this predictive power has only begun. This predictive power and the other advantages of connectionist networks are likely to make future work on cognitive modeling in this architecture very fruitful.

References

Berwick, R. and Weinberg, A. (1984). *The Grammatical Basis of Linguistic Performance*. MIT Press, Cambridge, MA.

Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, 2:137–167.

Church, K. (1980). On memory limitations in natural language processing. Master's thesis, Massachusetts Institute of Technology. MIT LCS Technical Report 245.

de Smedt, K. and Kempen, G. (1991). Segment grammar: A formalism for incremental sentence generation. In Paris, C., Swartout, W., and Mann, W., editors, *Natural Language Generation in Artificial Intelligence and Computational Linguistics*, pages 329–349. Kluwer Academic Publishers, Boston/Dordrecht/London.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, 7:195–225.

Fillmore, C., Kay, P., and O'Connor, M. (1988). Regularity and idiomaticity in grammatical constructions: The case of let alone. *Language*, 64:501–538.

Gibson, E. (1991). *A Computational Theory of Human Linguistic Processing: Memory Limitations and Processing Breakdown*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

Henderson, J. (1990). Structure unification grammar: A unifying framework for investigating natural language. Technical Report MS-CIS-90-94, University of Pennsylvania, Philadelphia, PA.

Henderson, J. (1994). *Description Based Parsing in a Connectionist Network*. PhD thesis, University of Pennsylvania, Philadelphia, PA. Technical Report MS-CIS-94-46.

John, M. S. and McClelland, J. (1992). Parallel constraint satisfaction as a comprehension mechanism. In Reilly, R. and Sharkey, N., editors, *Connectionist Approaches to Natural Language Processing*, pages 97–136. Lawrence Erlbaum Associates, Hove, U.K.

Kroch, A. (1989). Assymetries in long distance extraction in a tree adjoining grammar. In Baltin, M. and Kroch, A., editors, *Alternative Conceptions of Phrase Structure*. University of Chicago Press.

Marcus, M. (1980). *A Theory of Syntactic Recognition for Natural Language*. MIT Press, Cambridge, MA.

Marcus, M., Hindle, D., and Fleck, M. (1983). D-theory: Talking about talking about trees. In *Proceedings of the 21st Annual Meeting of the ACL*, Cambridge, MA.

Miller, G. A. (1956). The magical number seven plus or minus two. *Psychological Review*, 63:81–96.

Pesetsky, D. (1982). *Paths and Categories*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA. [distributed by MIT Working Papers in Linguistics, Dept of Linguistics and Philosophy, MIT].

Pollard, C. and Sag, I. A. (1987). *Information-Based Syntax and Semantics. Vol 1: Fundamentals*. Center for the Study of Language and Information, Stanord, CA.

Rumelhardt, D. E., McClelland, J. L., and the PDP Reseach group (1986). *Parallel Distributed Processing: Explorations in the microstructure of cognition, Vol 1*. MIT Press, Cambridge, MA.

Shastri, L. and Ajjanagadde, V. (1993). From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. *Behavioral and Brain Sciences*, 16:417–451.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216.

Steedman, M. (1987). Combinatory grammars and parasitic gaps. *Natural Language and Linguistic Theory*, 5:403–439.

Stevenson, S. (1994). Competition and disambiguation in a hybrid network model of human parsing. *Journal of Psycholinguistic Research*, 23(6).