Processing Unification-based Grammars in a Connectionist Framework

Andreas Stolcke

Computer Science Division
University of California, Berkeley
and
International Computer Science Institute
Berkeley, California

ABSTRACT

We present an approach to the processing of unification-based grammars in the connectionist paradigm. The method involves two basic steps: (1) Translation of a grammar's rules into a set of structure fragments, and (2) encoding these fragments in a connectionist network such that unification and rule application can take place by spreading activation. Feature structures are used to constrain sentence generation by semantic and/or grammatical properties. The method incorporates a general model of unification in connectionist networks.

INTRODUCTION

In recent years connectionist models have achieved notable results in modeling various aspect of perception and cognition. Although natural language processing has not been among the most prominent of its applications, there are a fair number of connectionist models of both language analysis and generation (Charniak & Santos, 1987; Cottrell, 1985; Dell, 1985; Fanty, 1985; Gasser, 1988; Kalita & Shastri, 1987; McClelland & Kawamoto, 1986). However, most of these models have a very narrow coverage, and hardly any attempts to take into account current linguistic theories of grammar (other than the basic context-free framework), for the most part adopting some ad-hoc linguistic formalism.

This paper's connectionist approach to natural language is based on *unification-based grammar*, a formal framework which has gained wide acceptance within the linguistic community within the last decade through its various variants (Kay, 1984; Kaplan & Bresnan, 1982; Gazdar et al., 1985). Although it is legitimate to argue that, by their very natures, formal grammar and connectionist models have different objectives (being competence versus performance theories, respectively), this work is intended as a first step towards a reconciliation of the two paradigms.

UNIFICATION-BASED GRAMMARS

Lack of space does not allow a self-contained overview of the formal linguistic apparatus underlying this work (Shieber (1986) gives an excellent introduction). Instead, we will present the basic features of the formalism by way of a simple example, upon which further discussion can be based. The version of unification-based grammar used here is essentially the one found in the PATR-II system (Shieber et al., 1983).

Feature Structures

Unification-based grammar extends traditional context-free grammars by introducing additional structure to the language it describes. The usual tree-like phrase structure of a sentence (or sentential form) is referred to as its *c-structure*. Additionally, each node in the c-structure (i.e. each constituent) has assigned to it a matrix of feature-value pairs encoding grammatical properties, semantic content, etc., referred to as its *feature structure* or *f-structure*. Features and values can be any (mnemonically chosen) atomic labels.

For example, the propositional semantics used in the sample grammar below will be encoded as f-structures of the form shown in Figure 1a.

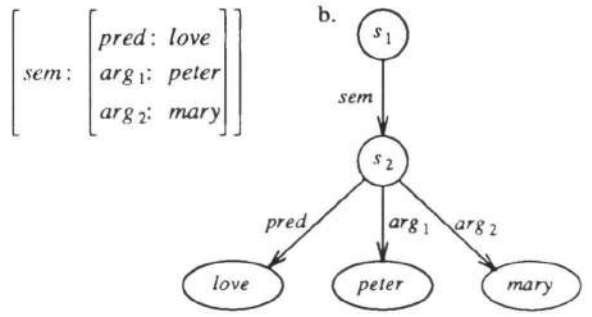

Fig. 1. F-structure representing *love(peter, mary)*.

sem is the main feature under which all semantic information is grouped. Feature *pred* contains the logical constant associated with a constituent, possibly with additional *arg*uments.

Note that the value of *sem* is a complex feature matrix, showing that f-structures may be embedded. Furthermore, feature values may be shared, i.e. the same subsidiary structure may be 'pointed to' by several features. These properties suggest that f-structures be represented as directed acyclic graphs (DAGs) with labeled edges, as in Figure 1b.

Finally, the concept of *unification* as used for terms in first-order languages can be applied to f-structures. Informally,

unifying two or more structures means merging their feature-value pairs recursively. Thus Figure 1a could be obtained as the unification of

$$\begin{bmatrix} sem: & \begin{bmatrix} pred: & love \\ arg_1: & peter \end{bmatrix} \end{bmatrix} \text{ and } \begin{bmatrix} sem: & \begin{bmatrix} arg_1: & peter \\ arg_2: & mary \end{bmatrix} \end{bmatrix}.$$

Unification might fail in case its operands contain incompatible features, such as if *peter* had been replaced with *john* in one of the structures above.

A Sample Grammar

We will now present a somewhat naive grammar generating simple active and passive constructions involving transitive verbs, such as "Peter loves Mary", "Peter is loved by Mary", etc.[1]

Consider the top-level structure of these sentences, as depicted in Figure 2.

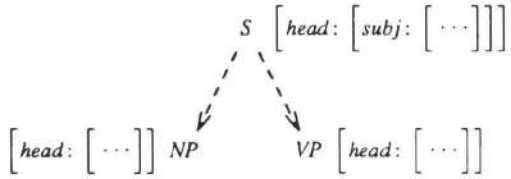

Fig. 2. C-structure and f-structure at sentence level.

The c-structure consists simply of three tree nodes, namely for the sentence (S) itself and its two subordinate constituents, noun phrase (NP) and verb phrase (VP). Attached to each of these is a piece of f-structure. By convention, the grammatical and semantic features of each c-structure node are grouped together under the *head* feature to be able to handle them as a whole. The *subj* feature in the S's *head* will be explained below.

Now consider the top-level rule for sentences, characterizing the structures shown in Figure 2. Grammar rules take the form of usual context-free rewriting rules, augmented with equations specifying that certain parts of the associated f-structures have to unify (failure to do so will render the rule inapplicable).

(R_S) $S \rightarrow NP\ VP$
$S.head = VP.head$
$S.head.subj = NP.head$

$S.head$ designates the value of the *head* feature in the f-structure belonging to the S node. The dot notation may be extended to specify feature values buried deeper in the f-structure, as in $S.head.subj$.

The second rule describes how to further expand verb phrases.

(R_{VP}) $VP \rightarrow V\ NP$
$VP.head = V.head$
$VP.head.obj = NP.head$

[1] Here and in the following, note the distinction between the surface string "Mary" and its semantics, the logical constant *mary*.

The unification equations so far specify that the *head* structures of S, VP, and V all have to unify, i.e. can be merged and thus effectively shared. The *head* features of the subject and object NP, on the other hand, will be unified with the values of *subj* and *obj*, respectively, in the sentence's *head*, i.e. they are effectively 'assigned' to these features.

All that has to be added now, for the grammar to fulfill its humble purpose, are lexical rules for NP and V (verbs).

(R_{Peter}) $NP \rightarrow$ Peter
$NP.head.sem = peter$

(R_{Mary}) $NP \rightarrow$ Mary
$NP.head.sem = mary$

(R_{loves}) $V \rightarrow$ loves
$V.head.sem.pred$ = love
$V.head.sem.arg_1 = V.head.subj.sem$
$V.head.sem.arg_2 = V.head.obj.sem$

For the purpose of exposition, passives are handled in a very simple-minded way, assuming auxiliary + past participle + "by" as a single complex verb. Thus passives can be generated by a single additional lexical rule.

$(R_{\text{loved-by}})$ $V \rightarrow$ is loved by
$V.head.sem.pred$ = love
$V.head.sem.arg_1 = V.head.obj.sem$
$V.head.sem.arg_2 = V.head.subj.sem$

This completes our sample grammar. Note how the assignment of grammatical roles (*subj*, *obj*) to semantic arguments (arg_1, arg_2) is neatly handled by unification equations, depending on the verb.

For simplicity, we have only considered the semantic features shown above; in a typical grammar additional equations would have to be included, such as

$NP.head.agree.person = 3rd$,
$V.head.agree = V.head.subj.agree$, or
$V.head.tense = pres$

to account for agreement, tense, etc.

There is one fine point about the rule notation which has been omitted so far: Category labels (S, VP, NP, etc.) are not really part of the c-structure, but are encoded as another standard feature at each node: *cat*. Thus, for convenience and to relate the notation to its context-free origins, a node designation such as VP is simply a shorthand for a generic node X with a category specification of $X.cat = VP$.

Grammar Rules As Fragments of Structure

In their usual interpretation unification equations function as declarative constraints on the f-structure assigned to a sentence's c-structure. Taking an alternative view, however, the rules can themselves be regarded as pieces of structure.

Besides being a very compact representation for rules, this interpretation will allow processing of rules using just one basic mechanism, namely unification. Thus, the goal of this transformation of rules into structure fragments is to have rule application translate precisely into unification of well-defined nodes in the corresponding fragments.

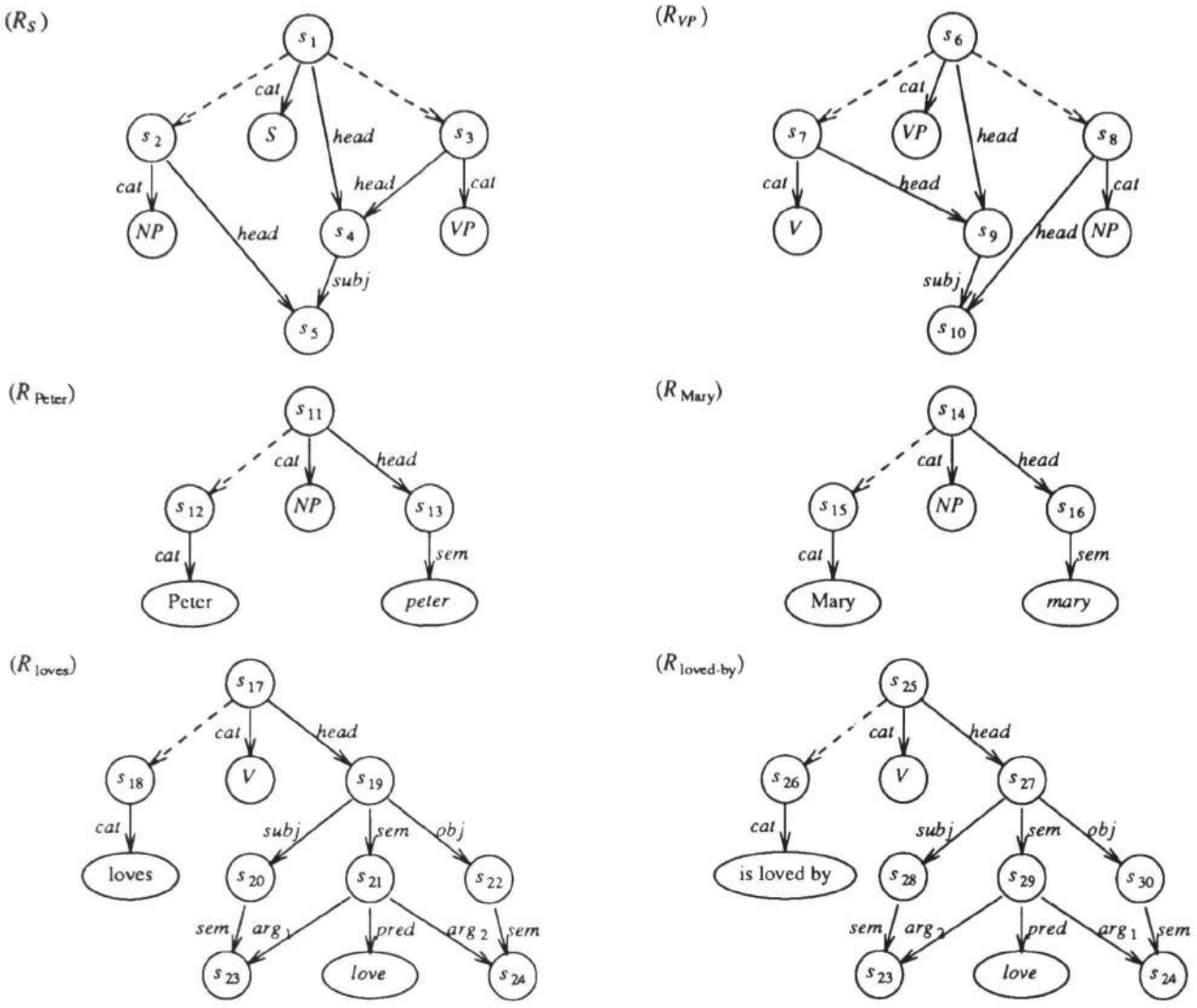

Fig. 3. Structure fragments derived from sample grammar.

The *c-structure fragment* derived from a grammar rule is rather obvious: Since the c-structure is just a tree, it can be obtained by 'pasting together' tree fragments of depth one, with a root node for the left-hand side part of the rule and one child node for each right-hand side element. For instance, rule (R_S) corresponds to a c-structure fragment consisting of an S node and two child nodes NP and VP. Likewise, rule (R_{VP}) has a VP root node and children V and NP. Applying (R_{VP}) to the VP in the right-hand side of (R_S) then corresponds to a simple 'merging' of the two VP nodes. Since c-structures are trees, we can view them as special cases of f-structures and interpret the node merging as a unification operation.[2]

The *f-structure fragment* corresponding to a rule is derived from its unification equations. We create a minimal DAG containing all the features and values mentioned in the equations, and encode equalities of values as reentrancies (shared nodes) in the DAG. We thus arrive at a DAG that can be interpreted as a somewhat generalized type of f-structure, since it contains not a single root node, but rather several 'root' or 'source' nodes, one for each element of the rule.

The result of applying this transformation to the sample grammar is depicted in Figure 3. Here c-structure fragments and f-structure fragments have been combined into a single structure by identifying a c-structure node with the root node of the f-structure assigned to it. C-structure edges are distinguished as dashed arrows, and internal nodes have been numbered for reference.

The critical point in the construction of f-structure fragments from rules is, again, that rule application maps directly to unification of corresponding nodes.[3] For example, both the c-structure and the f-structure of "Peter loves Mary" is obtained

[2] At this point it is crucial that, technically, c-structure nodes do not carry any category labels themselves. Rather, categories are encoded as *cat* values in the f-structures corresponding to c-structure nodes.

[3] We shall say that two f-structure *nodes* unify, iff the f-structures rooted in those nodes unify. In general, we use the root of a structure to designate the structure as a whole, whenever this is implied by the context.

by performing the following unifications ('~' denotes the 'unifies' relation):

$$s_2 \sim s_{11},\ s_3 \sim s_6,\ s_7 \sim s_{17},\ s_8 \sim s_{14}.$$

In our example, possible unifications are mainly restricted by category (*cat*) matching, but in a richer grammar agreement, selection restrictions, etc. would all be encoded in the f-structures and act as constraints.

F-structures As Sentence Specifications

One of the appealing features of unification-based grammars is that multiple levels of linguistic description can be accommodated within a single simple formal apparatus. Both syntax and semantics of a sentence can be encoded in f-structures and are generated by the rules if the grammar accounts for them. This is a very desirable property when the grammar is used in the context of sentence generation or parsing.

In generation, the semantics can be specified as a partially filled f-structure which automatically constrains the application of rules so as to produce sentences which conform to the specified semantics. Conversely, when parsing takes place based on the syntactic form of a sentence, its semantics are assembled as a side effect of f-structure construction by successive unifications.

The particular application experimented with in our research involved sentence generation; therefore the usage of f-structures as specifications for generation will be discussed in more detail. Suppose, e.g., we wanted to use the f-structure in Figure 1 to specify the semantics of the sentence to be generated. More formally, we want the f-structure of the root of the c-structure (i.e. the S node) to *unify* with

$$\left[head: \left[sem: \left[\begin{array}{ll} pred: & love \\ arg_1: & peter \\ arg_2: & mary \end{array} \right] \right] \right]$$

(For technical reasons, our grammar embeds *sem* features in the *head* values, hence the same embedding has to occur in the specification.)

It turns out, however, that specification by f-structures does not require any additions to our formal apparatus and its implementation. Alternatively, we can express the above constraint by adding to our grammar a new top-level category S', plus a rule of the form

$$\begin{array}{ll} (R_{S'}) & S' \rightarrow S \\ & \quad S'.head = S.head \\ & \quad S.head.sem.pred = love \\ & \quad S.head.sem.arg_1 = peter \\ & \quad S.head.sem.arg_1 = mary \end{array}$$

and generate sentences from S'.

This rule can then be transformed into a structure fragment as described before. As a result, sentence specifications can be incorporated in our framework in a natural way following the pattern shown above. This gives a set of structure fragments encoding the grammar, which is typically kept invariant within a certain context of application, plus a single varying f-structure encoding a sentence specification.

A CONNECTIONIST MODEL OF UNIFICATION

The previous section has shown that sentence generation in unification-based grammars can essentially be reduced to unification of structural fragments derived from the grammar. We will now describe how unification in turn can be efficiently implemented using the connectionist model of computation, i.e. a network of very simple processing units exchanging activation. This model of unification is by no means restricted to linguistic applications; connectionist unification is discussed in detail elsewhere (Stolcke, 1989). For the purposes of this paper an informal description will be sufficient.

Representing F-structures

Note that each f-structure, and hence the grammar as a whole can be represented as a set of edges, each characterized by a triple (node, feature, node). Each such triple is represented by a single so-called *e-unit*, with an activation corresponding to the presence or absence of the corresponding edge. Units will be designated by enclosing their 'meaning' in angle brackets. Thus we have, e.g., that the activation of e-unit $\langle s_3.cat = VP \rangle$ equals 1 whenever feature *cat* has value *VP* in structure (node) s_3, and is 0 otherwise. All the units in the implementation will be simple linear threshold units operating with activations of either 0 or 1.

This representational scheme is essentially a localized version of the encoding of S-expressions in BoltzCONS (Touretzky, 1986). E-units can be visualized as residing in a 3-dimensional space, with two 'node' dimensions and one 'feature' dimension.

Representing Unifications

Unification of f-structures can be viewed as a merging of DAG nodes. For example, consider the structures representing rules (R_S) and (R_{VP}) in Figure 3. Suppose structures s_3 and s_6 are to be unified. This means node s_3 has to be merged (unified) with s_6, and, due to the features *head* and *subj* present in both structures, s_4 has to unify with s_9 and s_5 with s_{10}.

Again, we use a localist representation to encode the 'unifies' relation on nodes. Each possible unification $s_i \sim s_j$ is represented by a *u-unit*, such that activation 1 on $\langle s_i \sim s_j \rangle$ indicates that the two nodes have been unified.

It is not sufficient, however, to just represent unifications; *non-unifiability* has to be dealt with explicitly, too. For example, suppose we wanted to unify s_7 in (R_{VP}) with s_{11} in (R_{Peter}). This will fail due to incompatible *cat* features V and NP, and will be represented in the network by turning on the *nu-unit* $\langle s_7 \not\sim s_{11} \rangle$. (Exactly how this takes place is the subject of the next section.)

Thus we have two spaces of units representing node unifications, u-units and nu-units, which can be thought of as organized along two 'node' dimensions. Obviously activations in these two sets of units have to be kept consistent. In particular, activity on corresponding u-units and nu-units should be mutually exclusive. The way the net is operated implies that u-units merely represent 'tentative' unifications

which should always be overruled by the stronger 'evidence' from nu-units. Therefore the relationship is not symmetrical, and nu-units simply deactivate their corresponding u-units by strong inhibitory links, so-called *nu-links*.

Unification As Constraint Satisfaction

Using the representational scheme described above, we now have to implement the *operation* of unification within our connectionist framework, using appropriate link patterns. Broadly speaking, the link structure must connect e-units and u/nu-units such that the unifications represented conform to the edges present in the f-structures.

The approach taken here is formally justified by a characterization of unifications as a specially constrained class of equivalence relations on f-structure nodes. This gives rise to a reformulation of unification as a constraint satisfaction problem with a straightforward connectionist implementation. It is possible, however, to describe the implementation without going into the formal details, so we will try to convey a more intuitive understanding.

Node equivalence

The equivalence relation alluded to above informally corresponds to the 'unifies' relation on nodes, as used in the discussion so far; it should be obvious that this relation actually has to be reflexive, symmetrical and transitive. Therefore, the activity patterns on n/nu-units have to be constrained to actually have these properties.

Reflexivity and symmetry can be encoded implicitly using simple techniques. To ensure reflexivity, all u-units $\langle x \sim x \rangle$ are kept active all the time, while all nu-units $\langle x \not\sim x \rangle$ are clamped to be permanently inactive.[4] Similarly, symmetry can be implicitly accounted for by the net structure by collapsing symmetric u-units $\langle x \sim y \rangle$ and $\langle y \sim x \rangle$ (and likewise for nu-units).

Transitivity, on the other hand, has to be encoded explicitly in the link structure. For every group of three u-units $\langle x \sim y \rangle$, $\langle y \sim z \rangle$ and $\langle x \sim z \rangle$, activity of any two of them should cause activation of the third.

There is a similar constraint involving nu-units, which is not completely symmetrical: For every pair of nu-units $\langle x \not\sim y \rangle$ and $\langle y \not\sim z \rangle$, and each u-unit $\langle x \sim z \rangle$, activity of one of the nu-units plus the u-unit should activate the other nu-unit.

A link setup that implements this behavior is shown in Figure 4. Also shown are the inhibitory nu-links mentioned earlier.[5] The type of conjunctive activation needed for transitivity has been realized by a set of intermediate units (*t-units*) labeled A through E in the figure. These units behave conjunctively, i.e. their threshold is set up so that *all* their inputs have to be active for the unit itself to become active (hence their distinctive square shape). Other units (represented as ellipses) generally work disjunctively, meaning that one

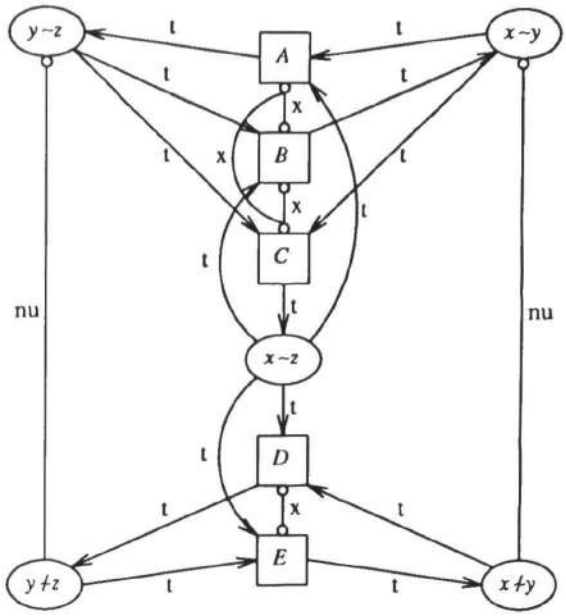

Fig. 4. Link structure enforcing transitivity.

active input is sufficient to exceed the unit's threshold turning it on.[6] Connections linking t-units to u-units and nu-units are called *t-links*.

For technical reasons, inhibitory *x-links* cause activation in t-units to be mutually exclusive, thus preventing stable coalitions (Feldman & Ballard, 1982) of u/nu-units.

Unification by spreading activation

We will now describe the link structure which forces node equivalence (represented by u/nu-units) to conform to the given set of f-structure edges (represented by e-units) and the definition of unification. Unification proceeds recursively, following the recursive composition of the f-structures being processed.

Specifically, when unifying two structures x and y containing the same feature f, say $x.f = x'$ and $y.f = y'$, we have to unify the values x' and y' recursively. In terms of node equivalence, this means that simultaneous presence of the edges $x.f = x'$ and $y.f = y'$ and the equivalence $x \sim y$ should induce the equivalence $x' \sim y'$. Again, this is implemented by a conjunctive pattern of activation, shown in the left half of Figure 5a. As for transitivity, an intermediate unit, A, is needed to realize the conjunctive behavior. A will become active only if all three of its inputs are active. The scheme causes activation representing equivalence to spread along co-occurring features top-down, i.e. from the root of the f-structures towards the leaves. Accordingly, the links and intermediate units transmitting this activation have been termed *td-links* and *td-units*, respectively.

A similar flow of activation occurs among the nu-units, since co-occuring features also transmit non-unifiability.

[4] As described later, these and other constant units can be eliminated by a simple space optimization. However, this optimization will be ignored here for clarity of exposition.

[5] Following connectionist convention, excitatory links are drawn as arrows, while inhibitory links carry small circles at their ends.

[6] Precise values for unit thresholds and link weights are ommitted here, focussing instead on the functionality of the net structure. Stolcke (1989) gives a complete specification.

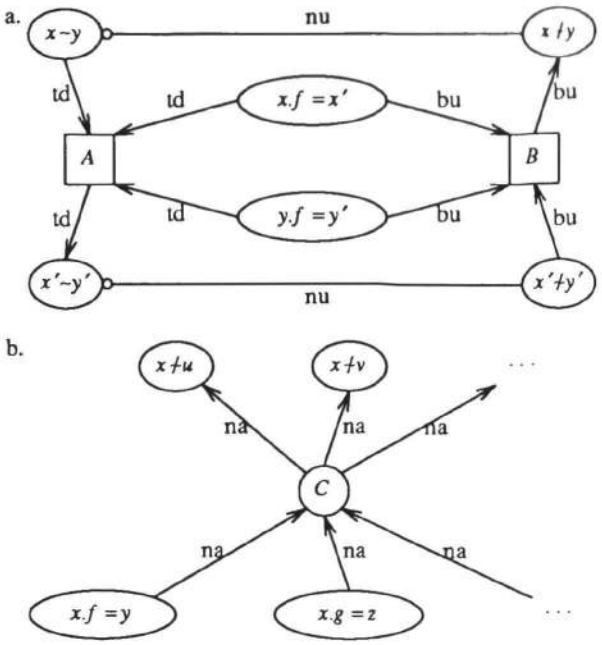

Fig. 5. Link structure performing unification.

Specifically, if the two values x' and y' are known to be non-unifiable, so are their parent nodes x and y. This gives rise to the link structure in the right half of Figure 5a, consisting of *bu-links* and the conjunctive *bu-unit* B. Thus activation representing non-unifiability flows in a bottom-up direction, possibly suppressing top-down activation via nu-links.

A question that remains is: Where does top-down and bottom-up activation originate? Top-down activation corresponds to an attempt to unify given f-structures, i.e. comes from some outside source, presumably some other network or input that uses the unification network as a 'subroutine'. In our case this initial activation is provided by a set of links that connect grammar rules to trigger the generation process, which will be described in the next section.

Bottom-up activation, on the other hand, originates in the f-structures themselves, generated by feature value mismatches at the leaves of the structures involved. A basic definitional property of unification is that non-identical atomic values can never unify. This implies that units like $\langle V \neq NP \rangle$ are clamped to remain constantly active; bottom-up activation will spread non-unifiability to any pair of nodes x and y which has these unequal values in the same feature f, activating $\langle x \neq y \rangle$, and so on further up the f-structure.

Another source of bottom-up activation comes from the fact that an atomic value can never unify with a complex one. Hence nodes with outgoing edges can never be equivalent to atomic nodes. This property is enforced by the link structure shown in Figure 5b. For every non-atomic node x, there is a disjunctive *na-unit* C which transmits activation from e-units to all nu-units $\langle x \neq u \rangle$, u atomic, using *na-links*. The intermediate unit C merely avoids full connectivity between e-units and nu-units, thus saving links.

Connectionist Unification: Summary

The preceding section presents a general mechanism for f-structure unification by connectionist means. The overall procedure is as follows: Two or more f-structures are 'input' to the net by activating the corresponding e-units. Following this, any number of unifications can be attempted by activating the u-units representing equivalence of the roots of the respective f-structures. After a time proportional to the depth of the structures these u-units will either remain active, indicating successful unification, or be turned of by the corresponding nu-unit, thus indicating failure.

Processing is extremely fast, since it explores subordinate f-structures in parallel, while requiring a reasonable amount of network resources: The most expensive aspects of the network are the links realizing transitivity (cubic in the number of nodes) and td/bu-links/units (quadratic in the number of edges).

A significant optimization is possible in case certain f-structures are fixed. This is true for our application, since all the grammar rules are typically fixed with the exception of the inital rule encoding a sentence specification. This implies that all corresponding e-units and a portion of the u/nu-units have constant activation. All of the category mismatches in a grammar can be precomputed this way. For example, $\langle V \neq NP \rangle$ is constantly 1 and implies the same for $\langle s_2 \neq s_{17} \rangle$, $\langle s_2 \neq s_{25} \rangle$, $\langle s_8 \neq s_{17} \rangle$, and $\langle s_8 \neq s_{25} \rangle$. In a second step, all units with constant activation (and the links incident upon them) can be eliminated from the network, resulting in savings of network resources and computation time.[7]

SENTENCE GENERATION

A traditional approach to sentence generation would use some top-level control structure to deal with rule selection, rule application, etc. Connectionist models lack the means to naturally implement global controlling instances and restrict themselves to purely local interactions of processing elements. This section describes how the model presented earlier can be extended to accomplish sentence generation.

Controlling Rule Application

Our approach to generation relies on parallel application of all the rules in the grammar whenever considering expansion of a non-terminal c-structure node. This is possible since the network holds all the rules of the grammar and allows parallel unification attempts to take place. The unification process itself will then single out those rules that are actually applicable, and incorporate the corresponding structure fragment into the structure generated so far.

To arrive at the link structure for this task, it is convenient to distinguish two special classes of nodes within the set of structures derived from the grammar rules. Those root nodes corresponding to left-hand side elements in rules are referred to as *L-nodes*, roots derived from right-hand side elements are

[7] Inactive units can be simply dropped. Active units can be eliminated after adjusting the thresholds of their neighbors according to the weights of the connections that are deleted in the process.

R-nodes. L-nodes in Figure 3 are s_1, s_6, s_{11}, s_{14}, s_{17}, and s_{25}; R-nodes are s_2, s_3, s_7, s_8, s_{12}, s_{15}, s_{18}, and s_{26}. Rule application corresponds to unification of R-nodes with L-nodes (L/R-unifications for short). Hence the approach of parallel rule application described above can be implemented by a link structure which attempts all possible L/R-unifications involving the R-nodes of a rule, once the L-node of that rule has been unified.

As an example, consider rule (R_{VP}), whose context-free component is $VP \rightarrow V\ NP$ with L-node s_6 and R-nodes s_7 and s_8. The link structure shown in Figure 6 triggers parallel rule application as follows:

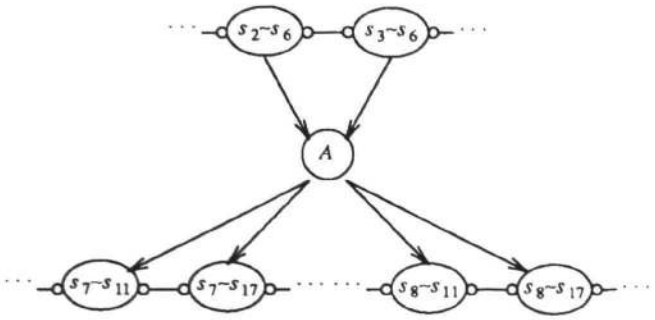

Fig 6. Link structure triggering rule application.

As soon as L-node s_6 unifies with any R-node (from any other rule), the intermediate disjunctive unit A transmits activation to the u-units responsible for unification of R-nodes s_{14} and s_{17} to other L-nodes.

Note that L/R-unifications involving the same L-node or R-node are mutually exclusive, since each right-hand side element can only be attached to exactly one left-hand side (and vice-versa). The mutally inhibitory links between u-units in Figure 6 implement this property.

Parallel rule application would in principle attempt any combination of L-nodes and R-nodes and include them in the schema shown in Figure 6. In practice those combinations resulting in category mismatches can be eliminated beforehand (e.g. $\langle s_7 \sim s_{11} \rangle$). Since categories are just *cat* feature values, however, this optimization would fall out as a by-product of the constant unit elimination process suggested earlier.

Implementing Specifications

Initial specification rules such as ($R_{S'}$) can be enoded as a degenerate structure fragment containing just a single R-node but no L-node (the R-node will be simply the root of the specifying f-structure). Accordingly, the schema from Figure 6 will be reduced to its bottom half. Assuming the root node of the specifying f-structure is s_0, this will give the links shown in Figure 7. Here the auxiliary unit A will serve the purpose of triggering the generation process as a whole.

Experience and Shortcomings

We used simlation tools to implement our model and investigate its dynamic behaviour empirically, although the underlying software posed significant limits on the size of networks

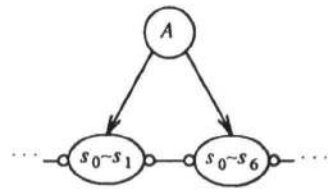

Fig 7. Links for initial rule application.

and hence grammars. The sample grammar and others of slighly higher complexity were encoded and used for generation to verify our method.[8]

One important property of the network is random selection of alternative formulations for a given semantics.[9] For example, given a *sem* structure as in Figure 1 as specification, the sample grammar will either generate an active ("Peter loves Mary") or a passive ("Mary is loved by Peter") construction. 'Priming' of either of these alternatives is possible by prespecifying what the *subj* of the sentence should be, or by specifying an explicit *aspect* feature (and accounting for it in the grammar).

The most serious drawback encountered was the apparent inability of the generation process to 'backtrack'. In cases where the unsuitability of a rule becomes evident only after several steps of intermediate rules applied successfully, the network has no means to undo the unifications and try alternatives. To guarantee successful generation, then, the specification must be specific enough to avoid such dead-end paths; this requirement, however, is clearly not acceptable for many purposes.

A different perspective on this problem might indicate a way to its solution. Rule applications from which the system has to backtrack eventually, roughly correspond to local minima in the energy function generally used to describe and analyze the dynamics of connectionst systems (Hopfield, 1982), whereas the set of unifications resulting in a complete sentence should constitute a global energy minimum. There exist standard techniques to 'escape' such local minima (Hinton & Sejnowski, 1986), but they do not apply to networks with asymmetric links such as ours.

A less serious shortcoming of the current model involves recursive rule applications. The model handles recursiveness insofar as the depth of the structures generated is limited only by the size of the relevant unit spaces. Each rule application, however, 'consumes' that rule in the sense that the corresponding fragment is incorporated in the overall structure, thus becoming unavailable for reuse. To allow multiple rule application, say of the VP rule, a corresponding number of duplicates of that rule has to be included in the grammar. A mechanism that accomplished such rule duplication 'on the spot' would run into well-known problems of connectionist models, in particular the variable binding problem.

[8] Credit is due to Kai Zimmermann at TU Munich for designing and implementing the LOOPS-based interactive network simulator that was the basis for our experimental work.

[9] This randomness is rooted in the asynchronous model of operation of the units.

EXTENSIONS AND DIRECTIONS FOR FUTURE RESEARCH

One of the most obvious extensions to the model presented here is its application to sentence analysis (parsing). The same transformation of grammar rules into structure fragments could be used, with the same connectionist approach to f-structure representation and unification. A sentence to be parsed would be represented by the set of fragments corresponding to the lexical items constituting it. Parsing would proceed combining those lexical items with rule fragments, using essentially a link structure inverse to the one in Figure 6 (links point bottom-up and the intermediate unit operates conjunctively rather than disjunctively).

A fundamental feature of natural language neglected in our model is time-sequentiality. Although considerable parallelism is probably essential for natural language processing, sequentiality is still inherent in priming effects on sequencing and other psycholinguistic phenomena (Bock, 1982). As a first step in this direction, activation flow (progressing unifications) could be modified so as to follow a left-to-right pattern. Early segments would be generated first, thereafter constraining what follows in the sentence.

Another issue is how current theories of grammar can account for the wealth of ungrammaticality found in actual utterances, such as agreement violations and blending of constructions. These phenomena suggest that grammaticality is really a matter of degree, realized in actual speech according to limitations of processing resources and other constraints. By their nature, connectionist models seem to be better suited to model graded grammaticality, yet it is not clear how to integrate such a notion with our or other models of processing.

Finally, it would be nice if a model of processing also gave some perspective on how its language-specific structures can be efficiently acquired, i.e. learned. With respect to our model, it remains an open question whether any of the known connectionist learning procedures can be applied to accomplish this task.

CONCLUSION

We have shown that connectionist models of natural language processing can efficiently incorporate state-of-the-art linguistic formalisms. In particular, we view our model of sentence generation as a first step towards an integration of unification-based grammar with connectionist principles. Also, the work reported here seems to suggest several possible directions in which traditional linguistic theories may be extended and modified to accommodate performance models of natural language.

ACKNOWLEDGEMENTS

Many thanks are due to the members of the AI research group at TU Munich, who were of great help during the preparation of the Diploma thesis on which this work is based.

I would also like to thank Joachim Diederich, Jerry Feldman, Jim Hendler and Charles Rosenberg for their valuable comments of earlier versions of this paper.

REFERENCES

Bock, J. Kathryn (1982), "Toward a Cognitive Psychology of Syntax: Information Processing Contributions to Sentence Formulation". *Psychological Review* 89(1): 1–47.

Charniak, Eugene & Santos, Eugene (1987), "A Connectionist Context-Free Parser Which is not Context-Free, But Then It is not Really Connectionist Either". In *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, Seattle, Wash., pp. 70–77.

Cottrell, Garrison W. (1985), "A Connectionist Approach to Word Sense Disambiguation". Technical Report TR 154. Computer Science Department, University of Rochester, Rochester, N.Y..

Dell, Gary S. (1985), "Positive Feedback in Hierarchical Connectionist Models: Applications to Language Production". *Cognitive Science* 9: 3–23.

Fanty, Mark (1985), "Context-Free Parsing in Connectionist Networks". Technical Report TR 174. University of Rochester, Rochester, N.Y..

Feldman, Jerome A. & Ballard, Dana H. (1982), "Connectionist Models and Their Properties". *Cognitive Science* 6: 205–254.

Gasser, Michael E. (1988), "A Connectionist Model of Sentence Generation in a First and Second Language". Technical Report UCLA-AI-88-13. Artificial Intelligence Laboratory, University of California, Los Angeles, Calif..

Gazdar, Gerald, Klein, E., Pullum, G. K., and Sag, I. A. (1985), *Generalized Phrase Structure Grammar*. Cambridge, Mass.: Harvard University Press.

Hinton, Geoffrey E. & Sejnowski, Terrence J. (1986), "Learning and Relearning in Boltzmann Machines". In David E. Rumelhart, James L. McClelland (Ed.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. Cambridge, Mass.: MIT Press, pp. 282–317.

Hopfield, J. J. (1982), "Neural networks and physical systems with emergent collective computational abilities". *Proceedings of the National Academy of Sciences USA* 79: 2554–2558.

Kalita, Jugal & Shastri, Lokendra (1987), "Generation of Simple Sentences in English Using the Connectionist Model of Computation". In *Proceedings of the 9th Annual Conference of the Cognitive Science Society*, Seattle, Wash., pp. 555–565.

Kaplan, Ronald M. & Bresnan, Joan (1982), "Lexical Functional Grammar: A Formal System for Grammatical Representation". In Joan Bresnan (Ed.), *The Mental Representation of Grammatical Relations*. Cambridge, Mass.: MIT Press, pp. 173–281.

Kay, Martin (1984), "Functional Unification Grammar: A formalism for machine translation". In *Proceedings of the 10th International Conference on Computational Linguistics*, Stanford, Calif., pp. 75–78.

McClelland, James L. & Kawamoto, Alan H. (1986), "Mechanisms of Sentence Processing: Assigning Roles to Constituents of Sentences". In David E. Rumelhart, James L. McClelland (Ed.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 2: Psychological and Biological Models*. Cambridge, Mass.: MIT Press, pp. 272–325.

Shieber, S. M., Uszkoreit, H., Pereira, F. C. N., and Robinson, J. J. (1983), "The Formalism and Implementation of PATR-II". In B. Grosz, M. Stickel (Ed.), *Research on Interactive Acquisition and Use of Knowledge*. SRI Final Report 1894. Artificial Intelligence Center, SRI International, Menlo Park, Calif..

Shieber, Stuart M. (1986), "An Introduction to Unification-Based Approaches to Grammar". CSLI Lecture Note Series. Center for Study of Language and Information, Stanford, Calif..

Stolcke, A. (1989), "A Connectionist Model of Unification". Technical Report TR 89-032. International Computer Science Institute, Berkeley, Calif..

Touretzky, David S. (1986), "BoltzCONS: Reconciling Connectionism with the Recursive Nature of Stacks and Trees". In *Proceedings of the 8th Annual Conference of the Cognitive Science Society*, Amherst, Mass., pp. 522–530.