Neural Computation, vol. 1, no. 4, 558-566, 1989

Unification as Constraint Satisfaction in Structured Connectionist Networks

Andreas Stolcke
Computer Science Division,
University of California, Berkeley, CA 94720,
and
International Computer Science Institute,
1947 Center Street, Suite 600, Berkeley, CA 94704

Abstract

Unification is a basic concept in several traditional symbolic formalisms that should be well-suited for a connectionist implementation due to the intuitive nature of the notions it formalizes. It is shown that by approaching unification from a graph matching and constraint satisfaction perspective a natural and efficient realization in a structured connectionist network can be found.

1 Introduction

Unification is a special matching operation on recursive symbolic structures widely used—in a number of variants—in fields related to symbolic artificial intelligence, most prominently theorem proving (Martelli & Montanari, 1982) and computational linguistics (Shieber, 1986). In the connectionist literature unification is addressed in the context of resolution theorem proving (Ballard, 1986), although considering only a simple special case.

Investigating the possibilities for a connectionist approach to unification seems worthwhile for at least two reasons: The importance and ubiquity of the concept in traditional formalisms, and the fact that unification incorporates notions that, at an informal level, seem to be essential to human cognition. These include integrating and merging of information into a consistent whole, checking for compatibility, and pattern matching. Since connectionist models are generally assumed to be well-suited for problems involving these tasks, unification is a potentially useful concept in the realm of neurally-inspired processing.

2 Feature Structures and Unification

Unification is usually defined in a strictly technical sense, namely referring to a specific operation in certain types of algebras. The variant considered here operates on recursive sets of attribute-value pairs known as *feature structures* (f-structures). An f-structure is either an atomic label like, e.g., *square*, or a set of *features* with f-structures as their *values*. Complex f-structures are conventionally represented as *feature matrices* such as

$$\begin{bmatrix} \mathit{shape}: & \mathit{square} \\ \mathit{length}: & \begin{bmatrix} \mathit{value}: & 5 \\ \mathit{unit}: & \mathit{inch} \end{bmatrix}^{*} \\ \mathit{width}: & {}^{*} \end{bmatrix} \tag{1}$$

The asterisk marks a feature value that is shared among more than one feature. These possible *reentrancies* in f-structures suggest an alternative representation which maps every structure into a rooted directed labeled graph. The graph corresponding to structure (1) is depicted in Figure 1(a).

Unifying a set of f-structures intuitively means merging all the features into a single structure, preserving reentrancies. Using $\sqcup$ as the unification operator, structure (1) may be obtained as:

$$\begin{array}{c} \begin{bmatrix} \mathit{length}: & \begin{bmatrix} \mathit{value}: & 5 \end{bmatrix} \end{bmatrix} \quad \sqcup \\ \begin{bmatrix} \mathit{width}: & \begin{bmatrix} \mathit{unit}: & \mathit{inch} \end{bmatrix} \end{bmatrix} \quad \sqcup \\ \begin{bmatrix} \mathit{shape}: & \mathit{square} \\ \mathit{length}: & \begin{bmatrix} \quad \end{bmatrix}^{*} \\ \mathit{width}: & {}^{*} \end{bmatrix} \quad = \quad \begin{bmatrix} \mathit{shape}: & \mathit{square} \\ \mathit{length}: & \begin{bmatrix} \mathit{value}: & 5 \\ \mathit{unit}: & \mathit{inch} \end{bmatrix}^{*} \\ \mathit{width}: & {}^{*} \end{bmatrix} \end{array} \tag{2}$$

Hence unification can be thought of as taking the union of features at each level and unifying values of identical features recursively. This process bottoms out at atomic values where feature values have to match exactly. Consequently unification is said to *fail* (or result in Ω) if atomic values mismatch as in

$$\begin{bmatrix} \mathit{shape}: & \mathit{square} \\ \mathit{color}: & \mathit{red} \end{bmatrix} \sqcup \begin{bmatrix} \mathit{color}: & \mathit{blue} \end{bmatrix} = \Omega \tag{3}$$

It should be evident from this short discussion that unification incorporates the three intuitive concepts alluded to above: data from several sources is merged into a single structure, checking for compatibility in the process. Alternatively, each of the structures being unified can be view as a recursive pattern against which all the other structures are matched.[1]

[1]The more familiar variant of *term unification* used in logic-based formalisms can be shown to be a special case of the version presented here (for details see, e.g., Stolcke (1989a)). Also, the model described here translates to term unification in a straightforward way.

3 Unification as Graph Matching

Consider the graphs involved in unification (2), shown in Figure 1(b). There is a mapping from nodes in the operand structures onto nodes in the unified structure such that edges are consistent among both. This suggests that unification can be viewed as a specialized form of graph matching and given a connectionist treatment as a a discrete constraint satisfaction problem (the same general approach has recently been used by Mjolness et al. (1989)).

This idea can be made precise by observing that the mapping from operand nodes to result nodes defines an equivalence relation (partitioning). Thus, in Figure 1, the partitions are $\{s_3, s_5, s_7\}$, $\{s_4, s_6, s_8\}$, $\{$*square*$\}$, $\{5\}$, and $\{$*inch*$\}$, corresponding to nodes s_1, s_2, *square*, 5, and *inch*, respectively, in the unified structure. Equivalence relations thus induced by unifications satisfy certain conditions:

(V1) For any pair of edges with identical labels (features) $x.f = x'$, $y.f = y'$, $x \sim y$ only if $x' \sim y'$.

(V2) For any pair of atomic nodes (values) u, v, $u \sim v$ only if $u = v$.

(V3) For any atomic node u and non-atomic node y, $u \sim y$ only if y has no outgoing edges $y.f$.

[Here $\sim$ denotes the equivalence relation and $x.f$ refers to the value of feature f on node (structure) x.] We will call equivalence relations satisfying these constraints *valid*, in accordance with Paterson and Wegman (1976) who used the concept to devise a fast sequential algorithm for unification.

It can be shown that the unification of a set of f-structures is isomorphic to the partitions obtained from the finest valid equivalence which makes the operand structures' root nodes equivalent. Such an equivalence is guaranteed to exist if the structures are unifiable. Hence we can redefine unification entirely in terms of a set of constraints on binary relations between graph nodes.

4 The Connectionist Implementation

4.1 Representation

F-structures can be seen as an abstraction of several frame-like data-structures commonly used for knowledge representation, some of which have been addressed in the connectionist literature using various distributed and local representations (Touretzky, 1987; Shastri, 1988). Our implementation uses a localist scheme for representing both f-structures and node equivalences. All units in the model are binary threshold units choosing their activations asynchronously between 0 and 1.

F-structures are represented simply as their constituting set of edges. Assuming we draw nodes from some pool N and given a feature set F, we obtain a pool $E \subseteq N \times F \times N$ of possible edges, each of which is assigned an *e-unit*. The e-unit $\langle x.f = y \rangle$

is active precisely when the corresponding edge is present. In many cases the semantics of a specific application rule out all but some small subset of $N \times F \times N$, or predetermine the presence or absence of many of the edges, thus avoiding the full combinatorics of the representation. This is illustrated in Stolcke (1989b) where the model is applied to the processing of unification-based grammars.

Node equivalences—or, loosely speaking, node unifications— in $N \times N$ are represented by *u-units*. The u-unit $\langle x \sim y \rangle$ is active iff x and y are considered equivalent (unified). The encoding of constraints requires that non-equivalence be explicitly represented as well, hence *nu-unit* $\langle x \not\sim y \rangle$ is active whenever x and y are non-unifiable.

4.2 Enforcing Validity

Figure 2(a) shows how the first validity constraint (V1) is enforced by interactions between e-unit, u-units, nu-units, and auxiliary units implementing conjunctions. The global result is that node unifications propagate top-down from the roots of the structures towards the leaves, while non-unifiability is determined bottom-up, starting at incompatible atomic values. Initial activation for nu-units is provided as a consequence of constraint (V2) which requires nu-units of the form $\langle u \not\sim v \rangle$, where u and v are non-equal atomic values, to be clamped on. The network structure in Figure 2(b) implements constraint (V3) and causes additional nu-units to become active.

To attempt unification of f-structures rooted in x and y, the u-unit $\langle x \sim y \rangle$ is externally activated. This is the only source of initial activation for u-units and also allows the network to return its result: if a unification exists the network will settle into a state where $\langle x \sim y \rangle$ and all other relevant u-units are turned on; otherwise $\langle x \sim y \rangle$ will be deactivated by $\langle x \not\sim y \rangle$. This happens either as a result of some stable state or as part of an oscillation in the network.

4.3 Enforcing consistency

The correctness of the solution found by the network relies on the fact that all its stable states represent valid equivalence relations. Hence consistency of the representation with regard to equivalence needs to be enforced. Reflexivity and symmetry can be made intrinsic to the representation by simply omitting reflexive u/nu-units and merging corresponding symmetric units. Transitivity is enforced by a dedicated link structure shown in Figure 3.

Consistency between u-units and nu-units is guaranteed by inhibitory links (shown in Figs. 2(a) and 3) that allow nu-units to suppress their counterparts unconditionally. This is justified by the observation that nu-units denote non-unifiability whereas u-units merely represent attempts at unification.

An exact specification of all network parameters, as well as a number of case studies of the network's dynamics can be found in Stolcke (1989a).

5 Discussion

The foremost characteristic of the connectionist implementation is that it naturally exploits opportunities for parallel processing of substructures. Therefore the network will arrive at a solution (or negative result) in time proportional to the depth of the f-structures processed if the structures are essentially tree-like, giving time complexity of the order of the log of the structure sizes. [Known serial algorithms are linear in the structure size (Paterson & Wegman, 1976).][2]

The number of units and links required for this speedup is quadratic in the number of edges and cubic in the number of nodes. Typical applications, however, often include a fair number of fixed edges. These together with nu-units that have to be clamped on due to constraints (V2) and (V3) imply that a certain portion of units have constant activations. Such units can be eliminated by a straightforward optimization.

The network places no limit on the number of separate f-structures being unified at once (other than by the total number of nodes and edges). When trying to unify sets of more than two f-structures cases arise where the complete set has no unifier, but overlapping subsets can still take part in partial unifications. The asynchronous operation of the network would randomly choose between alternative pairings of f-structures in this case. By adding controlled noise the network might be used to search stochastically through a space of mutually exclusive unifications.

Conventional algorithms usually do not allow f-structures to be cyclic, although unification is well-defined on such structures (Colmerauer, 1982). Incidentally our model naturally represents and processes cyclic structures.

It is encouraging (and maybe surprising) that a formalism prototypical for highly structured, symbolic processing lends itself to a relatively straightforward connectionist implementation, namely when approached from the point of view of constraint satisfaction. One basic deficiency of the model in its present form, however, is inherited from the underlying formalism it implements.

Unification as traditionally defined does not distinguish between different degrees of unifiability (or matching), although this might seem natural given its intuitive interpretation. Therefore it remains to be seen if unification can be usefully generalized into a graded notion of structured matching. Related to this issue, and a subject of ongoing research, is the question of how unification can deal with distributed and/or non-discrete forms of connectionist encoding of compositional structures, such as coarse-coding (Touretzky, 1986) and recursive auto-association (Pollack, 1988).

Acknowledgments

I wish to thank Steffen Hölldobler and Jerome Feldman for their helpful comments and the International Computer Science Institute for general support. The author is currently an IBM Graduate Fellow.

[2]Note, however, that in the worst case reentrancies can prevent effective parallelization causing the network to degenerate into sequential behavior (cf. Dwork et al. 1984).

References

Ballard, D. H. (1986). Parallel Logical Inference and Energy Minimization. In *Proceedings of the 5th National Conference on Artificial Intelligence*, pp. 203–208, Philadelphia, Pa.

Colmerauer, A. (1982). Prolog and infinite trees. In Clark, K. L. & Tarnlund, S.-A., eds., *Logic Programming*, pp. 231–251. Academic Press, New York.

Dwork, C., Kanellakis, P. C., & Mitchell, J. C. (1984). On the sequential nature of unification. *Journal of Logic Programming*, 1:35–50.

Martelli, A. & Montanari, U. (1982). An efficient unification algorithm. *ACM Transactions on Programming Languages and Systems*, 4:258–282.

Mjolness, E., Gindi, G., & Anandan, P. (1989). Optimization in model matching and perceptual organization. *Neural Computation*, 1(2):258–282.

Paterson, M. S. & Wegman, M. N. (1976). Linear Unification. Report RC 5904 (#25518), IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y.

Pollack, J. B. (1988). Recursive Auto-Associative Memory: Devising Compositional Distributed Representations. In *Proceedings of the 10th Annual Conference of the Cognitive Science Society*, pp. 33–39, Montreal, Quebec, Canada.

Shastri, L. (1988). A Connectionist Approach to Knowledge Representation and Limited Inference. *Cognitive Science*, 12:331–392.

Shieber, S. M. (1986). *An Introduction to Unification-Based Approaches to Grammar*. No. 4 in CSLI Lecture Note Series. Center for the Study of Language and Information, Stanford, Ca.

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

Stolcke, A. (1989b). Processing Unification-based Grammars in a Connectionist Framework. In *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, pp. 908–915, University of Michigan, Ann Arbor, Mich.

Touretzky, D. 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*, pp. 522–530, Amherst, Mass.

Touretzky, D. S. (1987). Representing Conceptual Structures in a Neural Network. In Caudill, M. & Butler, C., eds., *Proceedings of the IEEE 1st International Conference on Neural Networks*, pp. 279–286, San Diego, Calif.

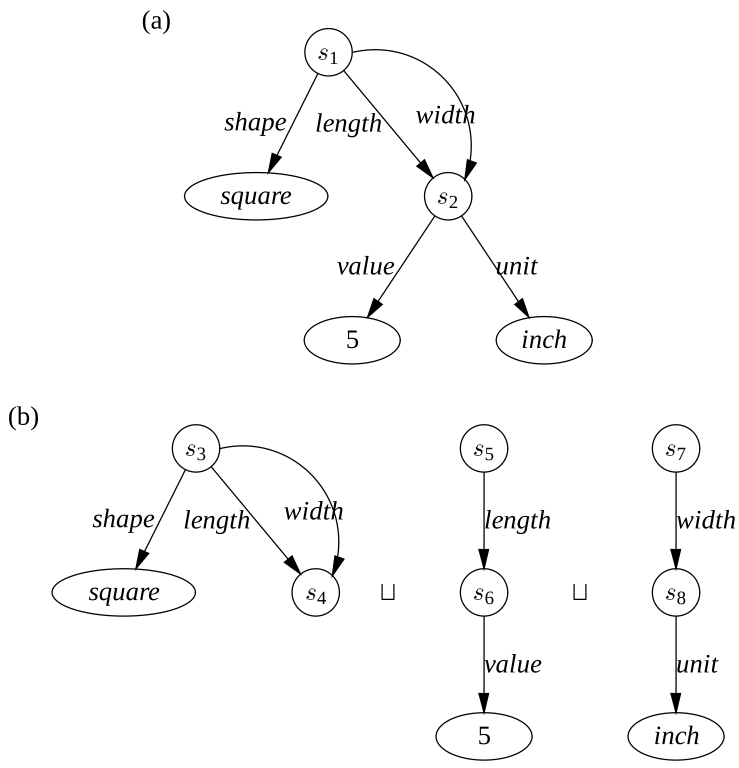

Figure 1: (a) Labeled graph representing structure (1). Features map into directed edges, atomic values translate into terminal nodes. Internal nodes correspond to substructures and are numbered for reference. (b) Three f-structures whose unification results in the structure given in (a).

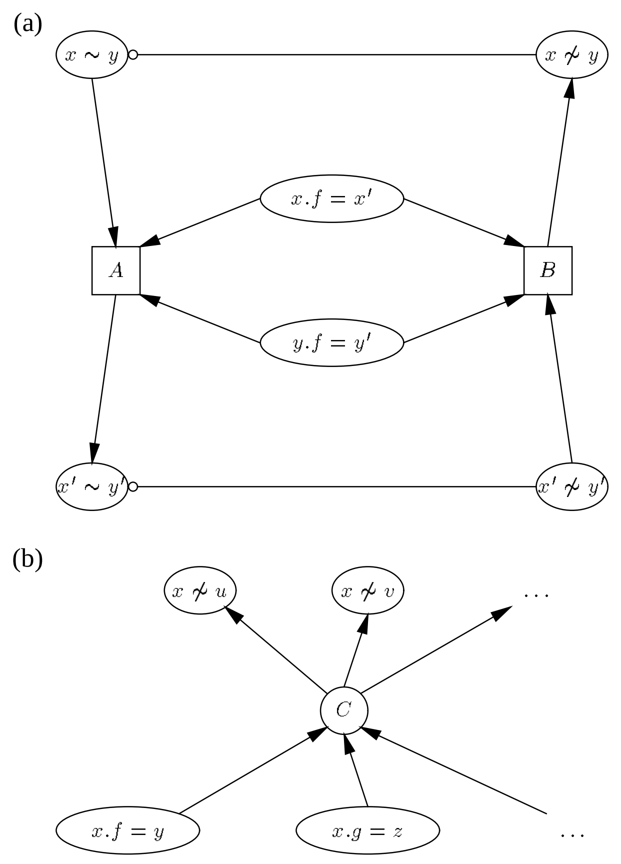

Figure 2: Link structure enforcing validity of node equivalences. (a) Auxiliary units A and B implement constraint (V1). They operate conjunctively, i.e. become active only when receiving activation from all three of their inputs. E-units with matching features f effectively enable paths allowing equivalences and non-equivalences to propagate top-down and bottom-up, respectively. Unification may fail, however, therefore nu-units are allowed to suppress corresponding u-units via strong inhibitory *nu-links*. (b) Constraint (V3) requires non-equivalence of a node x with any atomic nodes $u, v, \dots$ if x has any outgoing edges. This is accomplished by auxiliary unit C which behaves disjunctively.

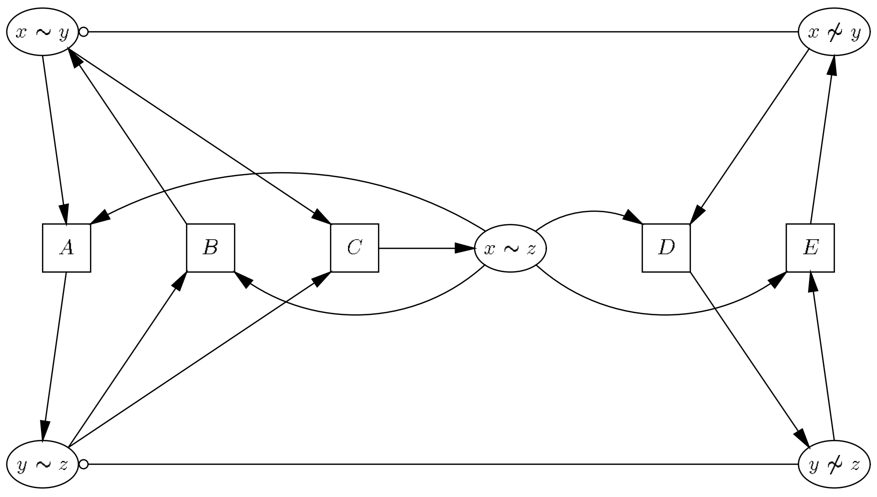

Figure 3: Network structure guaranteeing transitivity of node equivalences. Units A through E implement the implications $x \sim y \wedge y \sim z \Rightarrow x \sim z$ and $x \not\sim y \wedge y \sim z \Rightarrow x \not\sim z$. Omitted here are additional inhibitory links that render A, B, C and D, E mutually exclusive. These links are necessary to prevent stable coalitions of u-units and nu-units.