# A Structured Connectionist Unification Algorithm

**Steffen Hölldobler***
International Computer Science Institute
1947 Center Street, Suite 600
Berkeley, CA 94704, USA
E-mail: steffen@icsi.Berkeley.edu

## Abstract

A connectionist unification algorithm is presented. It utilizes the fact that the most general unifier of two terms corresponds to a finest valid equivalence relation defined on a occurrence-label representation of the unification problem. The algorithm exploits the maximal parallelism inherent in the computation of such a finest valid equivalence relation while using only computational features of connectionism. It can easily be restricted to solve special forms of the unification problem such as the word problem, the matching problem, or the unification problem over infinite trees.

## Introduction

Following Leibnitz's and Frege's idea to formalize human thought Herbrand, Gödel, and Skolem developed predicate logic by 1930. Great efforts were taken to find efficient proof procedures which can be used to mechanize human thought. A certain break-through was achieved when J. A. Robinson invented the resolution principle in 1965. In the meantime several other refutation techniques have been developed like Kowalski's (1979) connection graphs or Bibel's (1987) connection method. At the heart of all these refutation methods is the unification procedure, a version of which can already be found in Herbrand's theses (1930) and which was formally introduced by Robinson (1965).

Despite their success automatic theorem provers and logic programming languages are still plagued be several problems like the lack of a *clever* proof strategy or the lack of common sense. The problem being tackled in this paper is that most theorem provers do not explore the parallelism inherent in logic. They are still designed with a von-Neumann computer in mind. The sequential characteristics of such a computer is fundamentally different from the characteristics of an animal – and presumably a human – brain. In the brain slow neural computing elements with a switching time of a few milliseconds are heavily interconnected. Nevertheless, the brain is capable of performing complex tasks which require millions of operations on a conventional computer and this seems to be conclusive evidence that massive parallelism must take place in the brain.

*on leave from FG Intellektik, FB Informatik, TH Darmstadt, West-Germany

It is the goal of connectionist theories to utilize the understanding of the brain for building systems with interesting behaviour. The fundamental process in a connectionist model is the activation of associated units. However, Smolensky (1988) has emphasized that such a spreading of activation *cannot be adequate for complex tasks such as question answering*. And in an earlier paper Smolensky (1987) has argued that *connectionist systems may well offer an opportunity to escape the brittleness of symbolic AI systems ... if we can find ways of naturally instantiating the sources of power of symbolic computation within fully connectionist systems*. Such symbolic systems are powerful because they provide a combinatorial syntax and semantics and processes are structure sensitive (Fodor & Pylyshyn, 1988).

In his response to (Smolensky, 1988) J. McCarthy observed that in connectionist models which he has seen *the basic predicates are all unary and are even applied to a fixed object, and a concept is a propositional function of these predicates*. It is the goal of this paper to show a way out of the *propositional fixation* of connectionist models. This is done by demonstrating how the unification computation, which is at the center of inference, can be modeled in a connectionist system.

Informally, the unification problem is the question of whether there exists a substitution for two terms $s$ and $t$ such that the respective instances of $s$ and $t$ are equal. Paterson & Wegman (1978) have shown that this problem can be solved sequentially in time linear to the size of the problem. Dwork et. al. (1984) have proved that unification is logspace-complete and, thus, we should not expect that a parallel unification algorithm has a significant better time complexity in the worst case unless $P \subseteq NC$. However, we can expect that a parallel algorithm improves the time-complexity for the average case. As we will show the unification problem can be solved in 2 steps if it degenerates to a word or a matching problem, where a word (matching) problem is the question of whether $s$ (an instance of $s$) is equal to $t$. This significantly improves results by Dwork et al., who have shown that the matching problem of size $n$ can be solved in $log^2 n$ parallel time. Such an improvement is important for many applications as, for example, a study by Citrin (1988) has shown that up to 50% of Prolog's execution time is consumed by the unification process and many of the unifications are easy.

Distributed unification algorithms such as the one developed by Vitter & Simons (1986) make use of a $dag$[1]-representation of the unification problem. The parallelism exploited by these algorithm has its source in the decomposability axioms which characterize unification. These axioms state that two terms $f(s_1, \ldots, s_n)$ and $f(t_1, \ldots, t_n)$ are unifiable if all corresponding subterms $s_i$ and $t_i$ are unifiable. The unifiability of these subterms is determined in parallel as far as this is possible.

Most of the approaches towards a connectionist unification algorithm are parts of the design of larger inference systems. Let us describe these approaches as far as unification is concerned. Ballard (1986) prewires all substitutions and selects a substitution which is consistent with a refutation proof. This is possible because terms can only be variables and constants, clauses are used at most once and, thus, the set of substitutions is finite. Touretzky's & Hinton's (1988) DCPS is capable of matching a hypothesis of the form $(xab)(xcd)$ against the content of the working memory by searching for a minimum energy state, where $x$ is a variable and the remaining symbols are constants. While it seems to be possible to relax the constraints on the occurrences of variables in the hypothesis it is by no means obvious how their technique can be applied if $n$-ary function symbols are allowed and if the elements of the working memory may also contain variables. Mjolsness et al. (1989) match a dag with variables against a dag without variables by minimizing an objective function, which specifies the mismatch (or distance) between the dags. However, it is not obvious that they always find the most general solution and it seems to be impossible to extend their approach in order to deal with unification problems. Lange & Dyer (1989) and Ajjanagadde & Shastri (1989) assign a unique signature or phase (of a phased clock) to constants. Dynamic bindings are created by passing these signatures or phases. However, their systems do not guarantee in general that multiple occurrences of the same variable are bound to the same constant. It is also not easy to see how their proposals can be extended to cope with function embeddings which arise during the unification process if $n$-ary function symbols are allowed.

The connectionist approaches mentioned so far have severly shortcomings as far as the unification computation is concerned. Only Stolcke (1989) has recently investigated unification from a connectionist point of view. Stolcke represents terms as dags and – inspired by Paterson & Wegman (1978) – defines an equivalence relation on the nodes of a dag. He computes this relation via a connectionist network by explicitely coding the axioms defining the equivalence relation. However, Stolcke's algorithm will unify the terms $x$ and $f(x)$, since it does not check that the equivalence relation is acyclic or, in other words, it does not perform an occur check. Furthermore, the algorithm does not compute the most general unifier $\sigma$ for two terms $s$ and $t$, but the term $\sigma s$. It is not obvious how the algorithm can be changed such that an occur check is performed and the most general unifier is returned.

In this paper we present a connectionist unification algorithm. Terms and substitutions are represented as sets of occurrence-label pairs. Using Smolensky's (1987) terminology the occurrences are the roles that are filled by the labels. This representation allows us to represent terms and substitutions in a finite net though the set of terms as well as the set of substitutions is infinite. We define a finest valid equivalence relation on this representation, which represents a solvable unification problem, and show how this relation can be computed using only simple threshold units. Moreover, we formally prove that our connectionist model solves the unification problem. On this way we solve the variable binding problem and, moreover, ensure that multiple occurrences of the same variable are consistently bound to the same term (see e. g. (Barnden, 1984)). Due to lack of space we had to omit some details and all the proofs. They can be found in (Hölldobler, 1990). As far as this paper is concerned we do not address the problem how the connectionist net is recruited. We expect that the unification algorithm will be built into a larger system and that such a system will take care of this problem.

## The Unification Problem

We assume to have a finite *alphabet* consisting of a set $F$ of graded function symbols and a set $V$ of variables. Terms and substitutions are defined as usual. Throughout the paper $a, b, \ldots$ denote function symbols, $s, t, \ldots$ denote terms, and $z, y, \ldots$ denote variables.

A *unification problem* consists of two terms $s$ and $t$ and is denoted by $\langle s = t \rangle$. It is the problem of whether there exists a substitution $\sigma$ such that $\sigma s = \sigma t$. If such a substitution $\sigma$ exists then $\sigma$ is called *unifier* of $\langle s = t \rangle$. A substitution $\sigma$ is said to be a *most general unifier*, or *mgu*, for $\langle s = t \rangle$ iff for each unifier $\theta$ for $\langle s = t \rangle$ there exists a substitution $\lambda$ such that for each variable $x$ occurring in $\langle s = t \rangle$ we find $\theta x = \lambda \sigma x$. It is well-known that the unification problem is decidable and that an mgu of two terms can effectively be computed whenever the terms are unifiable (Robinson, 1965). Such an mgu is unique modulo variable renaming (Fages & Huet, 1986) and, therefore, is often called *the* mgu.

To develop a unification algorithm we need an axiomatization of unification which is suitable for a connectionist implementation. We will essentially use Paterson's & Wegman's (1978) approach, but our algorithm is not based on a dag but on an *occurrence-label* representation of the unification problem.

The set of *occurrences* of a term $t$, $O(t)$, is inductively defined as $\Lambda \in O(t)$, and $\pi \in O(t_i)$ implies $i \cdot \pi \in O(f(t_1, \ldots, t_i, \ldots, t_n))$ for all $1 \leq i \leq n$[2]. On occurrences a partial ordering is defined by $\pi_1 \geq \pi_2$ iff there

[1] directed acyclic graph

[2] We omit $\Lambda$ and $\cdot$ if this does not lead to confusion.

exists a $\pi_3$ such that $\pi_1 \cdot \pi_3 = \pi_2$. Furthermore, $\pi_1 > \pi_2$ iff $\pi_1 \geq \pi_2$ and $\pi_1 \neq \pi_2$.

The set of *labels*, $l(\pi)$, for each occurrence $\pi$ in the terms of a unification problem $\langle s = t \rangle$ is the set of symbols occurring at $\pi$ in $s$ and $t$. The set $l(\pi)$ can be split into the disjoint subsets $l_f(\pi)$ and $l_v(\pi)$, which contain the function symbols and variables, respectively. For the unification problem $\langle f(x,x,y) = f(g(y),g(g(z)),g(a)) \rangle$ we obtain

| | $l_f$ | $l_v$ | $l$ |
|---|---|---|---|
| $\Lambda$ | $\{f\}$ | $\emptyset$ | $\{f\}$ |
| 1 | $\{g\}$ | $\{x\}$ | $\{g,x\}$ |
| 11 | $\emptyset$ | $\{y\}$ | $\{y\}$ |
| 2 | $\{g\}$ | $\{x\}$ | $\{g,x\}$ |
| 21 | $\{g\}$ | $\emptyset$ | $\{g\}$ |
| 211 | $\emptyset$ | $\{z\}$ | $\{z\}$ |
| 3 | $\{g\}$ | $\{y\}$ | $\{g,y\}$ |
| 31 | $\{a\}$ | $\emptyset$ | $\{a\}$ |

In the sequel let $\langle s = t \rangle$ be a unification problem and $O = O(s) \cup O(t)$. An *equivalence relation* $\sim$ on $O$ is said to be

*decomposable*[3] iff $\forall \pi_1, \pi_2 \in O : \pi_1 \sim \pi_2 \in O$
$\wedge \exists i : \{\pi_1 \cdot i, \pi_2 \cdot i\} \subseteq O \Rightarrow \pi_1 \cdot i \sim \pi_2 \cdot i$,

*homogeneous* iff $\forall \pi_1, \pi_2 \in O : \pi_1 \sim \pi_2$
$\Rightarrow |l_f(\pi_1) \cup l_f(\pi_2)| \leq 1$,

*singular* iff $\forall \pi_1, \pi_2 \in O : l_v(\pi_1) \cap l_v(\pi_2) \neq \emptyset \Rightarrow \pi_1 \sim \pi_2$.

A decomposable and singular equivalence relation is called *DSE-relation*. We denote a $\sim$-equivalence class $C$ by $[\pi_1, \ldots, \pi_n]$ whenever $\{\pi_1, \ldots, \pi_n\} \subseteq C$. For $\sim$-equivalence classes $C_1$ and $C_2$ we define $C_1 \succ C_2$ iff there exist $\pi_1 \in C_1$ and $\pi_2 \in C_2$ such that $\pi_1 > \pi_2$. An equivalence relation $\sim$ is said to be *acyclic* iff the $\sim$-equivalence classes are partially ordered by $\succ$. A homogeneous and acyclic DSE-relation is said to be *valid*.

For $\langle f(x,x,y) = f(g(y),g(g(z)),g(a)) \rangle$ a valid equivalence relation is defined by the equivalence classes $[\Lambda] \succ [1,2] \succ [11,21,3] \succ [211,31]$. For $\langle f(x,y) = f(g(y),g(x)) \rangle$ there exists a homogeneous DSE-relation with equivalence classes $[1,21]$ and $[11,2]$. But this relation is cyclic. To gain efficiency the check whether a DSE-relation is acyclic – also called *occur check* – has been omitted in virtually all logic programming systems. Colmerauer (1984) *corrected* this bug by interpreting logic programs no longer over the Herbrand universe or finite trees, but over the domain of infinite trees. There, the latter problem is solvable by replacing $x$ as well as $y$ by the infinite tree $g(g(g(\ldots)))$. Proposition 1 is an immediate consequence of (Paterson & Wegman, 1978) and (MacQueen et al., 1984).

### Proposition 1

1. *$\langle s = t \rangle$ has a solution over the domain of infinite trees iff there is a homogeneous DSE-relation on $O$.*
2. *$\langle s = t \rangle$ has a solution iff there is a homogeneous and acyclic DSE-relation on $O$.*

Moreover, Paterson's & Wegman's have shown that the mgu of two terms can be constructed from the finest valid equivalence relation on the set of occurrences of the unification problem. For our running example we will briefly recall this technique. $[\Lambda]$ is the largest class[4] (wrt $\succ$). But since $[\Lambda]$ is not labelled by a variable we can discard it. For the next class, $[1,2]$, we find – by inspecting the labels as well as the ordering $\succ$ – that $x$ is bound to $g(y)$ and we generate the binding $\{x \leftarrow g(y)\}$. Similarly, for $[11,21,3]$ and $[211,31]$ we obtain the bindings $\{y \leftarrow g(z)\}$ and $\{z \leftarrow a\}$, respectively. Combining these bindings yields

$$\{z \leftarrow a, y \leftarrow g(a), x \leftarrow g(g(a))\},$$

which is the mgu of the unification problem. Because of this technique and since we regard our connectionist unification algorithm as part of a larger inference system utilizing the same representation as the unification algorithm itself, we are satisfied if the connectionist unification algorithm generates the finest valid equivalence relation for a unification problem.

## A Connectionist Unification Algorithm

In the first step towards a connectionist unification algorithm we use the fact that a DSE-relation can be characterized by its equivalence classes, which in turn can be represented by the union of the labels of its members. As we show in the following subsection, these unions can be computed by two simple operations derived from the axioms of decomposability and singularity. Thereafter we demonstrate how the unification problem can be represented and how the finest DSE-relation can be computed by a connectionist model. It remains to be checked that the finest DSE-relation is homogeneous and acyclic. This can be done by a simple extension of the connectionist model developed until then.

### Representing a DSE-Relation

Let $\sim$ be a DSE-relation for a unification problem $\langle s = t \rangle$. For a $\sim$-equivalence class $C$ the set of labels, $l(C)$, is the union of the set of labels of its elements. As before, the set of labels of $C$ can be split into the disjoint subsets $l_f(C)$ and $l_v(C)$ containing the function symbols and variables, respectively. As example consider $\langle f(x,x,y) = f(g(y),g(g(z)),g(a)) \rangle$ and we find

| | $l_f$ | $l_v$ | $l$ |
|---|---|---|---|
| $[\Lambda]$ | $\{f\}$ | $\emptyset$ | $\{f\}$ |
| $[1,2]$ | $\{g\}$ | $\{x\}$ | $\{g,x\}$ |
| $[11,21,3]$ | $\{g\}$ | $\{y\}$ | $\{g,y\}$ |
| $[211,31]$ | $\{a\}$ | $\{z\}$ | $\{a,z\}$ |

Since we intend to represent the $\sim$-equivalence classes by $l([\pi])$ we have to generate $l([\pi])$. Let $A$ contain the

[3] Decomposability is often called *correspondence* (e.g. (Dwork et al., 1984)) and sometimes implies homogeneity (e.g. (Kirchner, 1984)).

[4] called *root class* in (Paterson & Wegman, 1978)

axioms of reflexivity, symmetry, transitivity, decomposability, and singularity. By $A \vdash \pi_1 \sim \pi_2$ we denote that $\pi_1 \sim \pi_2$ can be derived from $A$. It can now be shown that for $\pi_1 \neq \pi_2$

$$A \vdash \pi_1 \sim \pi_2 \Leftrightarrow \exists \pi, \pi_1', \pi_2' : A \vdash \pi_1' \sim \pi_2' \wedge \pi_1 = \pi_1' \cdot \pi \wedge \pi_2 = \pi_2' \cdot \pi \wedge l_v([\pi_1', \pi_2']) \neq \emptyset$$

holds. This tells us that, if $\pi_1$ and $\pi_2$ are in the same DSE-equivalence class, then either there is a variable among the labels of $[\pi_1, \pi_2]$ or we find occurrences $\pi, \pi_1', \pi_2'$ such that $\pi \neq \Lambda$, $\pi_1 = \pi_1' \cdot \pi$, $\pi_2 = \pi_2' \cdot \pi$, and there is a variable among the labels of $[\pi_1', \pi_2']$. In other words, the finest DSE-relation can be constructed entirely from occurrences which are labelled with the same variable. The key idea of the connectionist unification algorithm is to increase the set of labels of each occurrence $\pi$ until $l(\pi) = l([\pi])$. When shall the set of labels be increased? By the singularity of $\sim$ we find that

$$l(\pi_1) \leftarrow l(\pi_1) \cup l(\pi_2) \quad \text{whenever } l_v(\pi_1) \cap l_v(\pi_2) \neq \emptyset \quad (op_S)$$

and by the decomposability of $\sim$ and the previous result we find that

$$l(\pi_1 \cdot \pi) \leftarrow l(\pi_1 \cdot \pi) \cup l(\pi_2 \cdot \pi) \quad \text{whenever } l_v(\pi_1) \cap l_v(\pi_2) \neq \emptyset \quad (op_D)$$

where $\leftarrow$ denotes assignment and $\pi \neq \Lambda$. In the sequel we show that these two operations can easily be performed by a connectionist model and that they generate indeed the finest DSE-relation.

## Representating a Unification Problem

The unification algorithm is based on Feldman's & Ballard's (1982) connectionist model. *Units* are charcterized by a *potential* $p$, an *output value* $v$, and a vector of *inputs* $i_1, \ldots, i_n$. In particular we use so-called *threshold units*, whose potential and output values are determined via the rules

$$p \leftarrow \sum w_k i_k,$$
$$v \leftarrow \text{if } p \geq \theta \text{ then 1 else 0},$$

where $\theta$ is a constant called *threshold* and $w_k$ are weights on the input values. The unique output value is spread along all connections from the unit though these connections are not always drawn from the same location. For convenience we occasionally use a *bidirectional* link $\leftrightarrow$ with weight $w$ between two units $u_1$ and $u_2$ as an abbreviation for two links with weight $w$; one from $u_1$ to $u_2$ and another one from $u_2$ to $u_1$.

The sets of labels for a unification problem can be represented by threshold units $M(\pi, j)$ for each occurrence $\pi$ and each symbol $j$ such that $M(\pi, j)$ is active iff $j$ is a label of $\pi$. A unification problem is specified by externally activating the units which define the problem. This external activation has to be maintained throughout the computation since threshold units do not memorize their potential. For our running example we obtain the units depicted in figure 1(a) as a matrix. One should observe that this representation is not unique. The unification problem $\langle f(x, g(g(z)), y) = f(g(y), x, g(a)) \rangle$ has

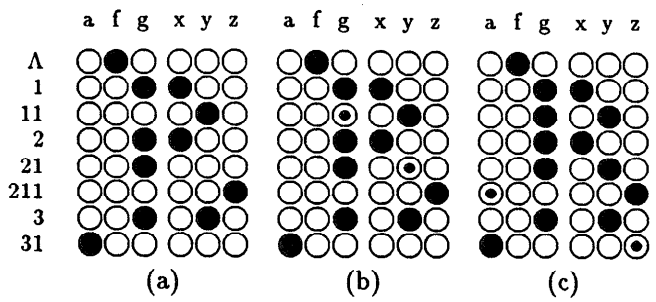


Figure 1: The representation of $\langle f(x, x, y) = f(g(y), g(g(z)), g(a)) \rangle$; (a) initially, (b) after 1 step, and (c) after 2 steps, where the most recently activated units are half-filled.

the same representation as the running example. But this does not lead to a problem since the finest valid equivalence relation is identical for both examples.

If we update the labels according to the operations $op_S$ and $op_D$ we obtain figures 1(b) and 1(c) after 1 and 2 steps, respectively. For example, since $M(11, y)$, $M(3, y)$, and $M(3, g)$ are active in (a), $M(11, g)$ has become active in (b) by $op_S$. Since $M(1, x)$, $M(2, x)$, and $M(11, y)$ are active in (a), $M(21, y)$ has become active in (b) by $op_D$. The final matrix in (c) represents precisely the labels of the finest DSE-relation and is considered as the output of our unification algorithm.

How must a connectionist network look like in order to implement the operations $op_S$ and $op_D$? We propose a network consisting of two layers called term and unification layer. Let $n = |O(s) \cup O(t)|$ be the size of $\langle s = t \rangle$, $m$ be the cardinality of the alphabet, and $w = \frac{1}{2}mn(n-1)$ be an integer used to set up thresholds and weights.

The *term layer* contains the representation for a unification problem as an $n \times m$ matrix $M$ of threshold units with threshold $w$. Each unit is connected via bidirectional links with $n-1$ units in the unification layer and has weight $w$. The *unification layer* contains the units necessary to implement $op_S$ and $op_D$. For both operations the algorithm has to determine whether two occurrences share a common variable and to update sets of labels accordingly. This requires that for any two occurrences $\pi_1$ and $\pi_2$ and for any symbol $j$, $M(\pi_1, j)$ and $M(\pi_2, j)$ are connected. Since the unification problem has $n$ different occurrences and $m$ different symbols, we need $\frac{1}{2}n(n-1)$ units for each symbol, i.e. altogether $w$ units. These units can be represented by a $\frac{1}{2}n(n-1) \times m$ matrix $U$ of threshold units with threshold $w+1$. An element of this matrix is denoted by $U(\{\pi_1, \pi_2\}, j)$, or $U(\pi_1, \pi_2, j)$ for short[5], indicating that this unit is connected with $M(\pi_1, j)$ and $M(\pi_2, j)$. Each unification layer unit is also connected to other unification layer units with weight 1 such that there is a connection from

[5] Note, $U(\pi_1, \pi_2, j)$ and $U(\pi_2, \pi_1, j)$ denote the same unit.

$U(\pi_1, \pi_2, x)$ to $U(\pi_1, \pi_2, j)$
for all $x \in V$, $j \in V \cup F$, $j \neq x$, and
$U(\pi_1, \pi_2, x)$ to $U(\pi_1 \cdot \pi, \pi_2 \cdot \pi, j)$ for all $\pi \neq \Lambda$, $x \in V$, $j \in V \cup F$ such that $\{\pi_1 \cdot \pi, \pi_2 \cdot \pi\} \subseteq O(s) \cup O(t)$.

The threshold of a unification layer unit is chosen such that active unification layer units can only activate another unification layer unit if this unit receives also activation from the term layer. Conversely, a term layer unit is activated as soon as a corresponding unification layer unit is active. To exemplify the network figure 2 shows the term layer together with the unification layer units and the connections needed to solve our running example. All externally activated term layer units are represented as full circles. The interested reader is encouraged to verify that the number $i$ in a unit indicates that this unit will be activated after $i$ steps.

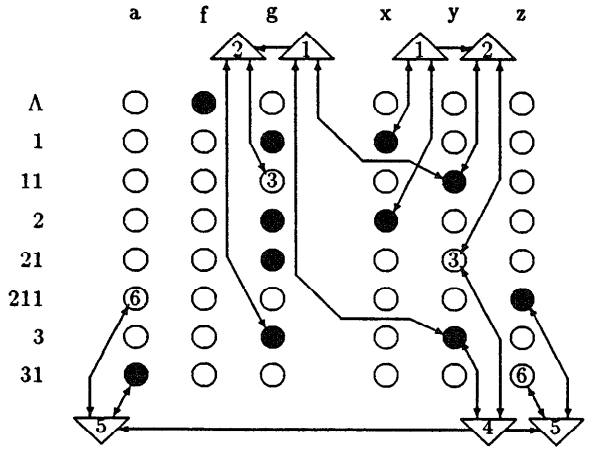


Figure 2: The cyclic term layer together with the triangular unification layer units and the connections needed for solving $\langle f(x, x, y) = f(g(y), g(g(z)), g(a))\rangle$. Recall that a unification layer unit $U(\pi_1, \pi_2, j)$ is uniquely determined by *its* term layer units $M(\pi_1, j)$ and $M(\pi_2, j)$. Each connection between unification layer units has weight 1 and each connections between a unification and a term layer unit has weight $w$.

## Computing the finest DSE-relation

We are of course interested in whether the connectionist model computes the finest DSE-relation for any solvable unification problem. To answer this question we define a function $T$ on the units of the term and unification layer. Let $N$ be a set of term and unification layer units in $T(N) = N \cup T_1(N) \cup T_2(N)$, where

$$T_1(N) = \{M(\pi, j) \mid \exists \pi' : U(\pi, \pi', j) \in N\}$$
$$T_2(N) = \{U(\pi_1, \pi_2, j) \mid \{M(\pi_1, j), M(\pi_2, j)\} \subseteq N \vee [(M(\pi_1, j) \in N \vee M(\pi_2, j) \in N) \wedge D(\pi_1, \pi_2, N)]\}$$
$$D(\pi_1, \pi_2, N) = \exists x \in V : \exists \pi, \pi_1', \pi_2' : \pi_1 = \pi_1' \cdot \pi \wedge \pi_2 = \pi_2' \cdot \pi \wedge U(\pi_1', \pi_2', x) \in N.$$

By the definition of $T$ and the fact that there are only finitely many units we find that

$T(N) \supseteq N$,
$T(N) = N$ implies $T^i(N) = N$ for all $i > 0$, and
$\exists k : T^{k+1}(N) = T^k(N)$.

In fact, we can show that $k$ is bounded by the number $n$ of occurrences in a unification problem. Now let $N$ be the set of active term layer units which represent a unification problem and let $N^* = T^k(N)$ such that $T(N^*) = N^*$. The interested reader may check that for our running example $N^* = T^6(N)$. We can now answer the question raised at the beginning of this subsection.

**Theorem 2**

$\forall \pi \in O : \forall j \in V \cup F : M(\pi, j) \in N^* \Leftrightarrow j \in l([\pi])$.

This result ensures that a finest DSE-relation for a unification problem is generated by the connectionist model. That it is indeed the *finest* DSE-relation follows immediately from the fact that each activation was forced by either the axiom of singularity or the axiom of decomposability. The space complexity is bound by the square of the size $n$ of the unification problem, whereas the time complexity is bound by $n$. We should not be disappointed by these results as Dwork et al. (1984) have proved that unification is logspace complete. But look at a best case scenario. This is the case if the unification problem degenerates to a *word problem*, i.e. to the question of whether two terms $s$ and $t$ are syntactically equal. If the word problem is solvable, then the finest DSE-relation is found after 1 step. Similarly, we can show that even for a *matching problem*, i.e. the problem of whether there exists a substitution $\sigma$ for $s$ and $t$ such that $\sigma s = t$, the finest DSE-relation is also found after 1 step. As an example consider $s = f(x, g(x))$ and $t = f(h(a), g(h(a)))$. Then,

$$N = \{M(\Lambda, f), M(1, x), M(1, h), M(11, a), M(2, g), M(21, x), M(21, h), M(211, a)\}$$

and $T(N)$ as well as $T^2(N)$ are equal to

$$N \cup \{U(1, 21, h), U(11, 211, a), U(1, 21, x)\}.$$

Hence, the matching problem is solvable with most general solution $\{x \leftarrow h(a)\}$.

## Homogeneous DSE-relations

A DSE-relation need not to be homogeneous. This can be seen if we alter our running example by replacing the $z$ by a new constant $b$. The finest DSE-relation for $\langle f(x, x, y) = f(g(y), g(g(b)), g(a))\rangle$ can be computed as before and we find that $211 \sim 31$ and $l_f([211, 31]) = \{a, b\}$. However, since $\sim$ is homogeneous iff we find for all $\pi$ that $|l_f([\pi])| \leq 1$, we conclude that $\sim$ is not homogenous iff there is a row $\pi$ in the term layer and at least two function symbols $f_1$ and $f_2$ such that $M(\pi, f_1)$ and $M(\pi, f_2)$ are active. This condition can be directly translated into a connectionist network. For each row $\pi$ of the term layer we insert an additional threshold

unit which receives activation from each unit $M(\pi, f)$, where $f$ is a function symbol, and becomes active as soon as it is excited by two term layer units.

Proposition 1(1) ensures that the algorithm developed so far decides a unification problem over infinite trees. One should observe, that the matching problem is the same regardless whether it is interpreted over the domain of finite or infinite trees. Thus, we find that whenever $T^2(N) \supset T(N)$ the matching problem is unsolvable and, hence, a matching problem can be decided in 2 steps. This improves a result by Dwork et al. (1984), who have shown that the matching problem of size $n$ can be solved on a PRAM in $log^2 n$ parallel time.

## Valid Equivalence Relations

It remains to be checked whether the finest homogeneous DSE-relation is acyclic or, in other words, valid. Unfortunately, there is no unit in our connectionist model which represents a $\sim$-equivalence class. However, we know that two different occurrences $\pi_1$ and $\pi_1'$ are equal under $\sim$ iff there are a variable $x$ and occurrences $\pi_2$, $\pi_2'$, and $\pi$ such that $U(\pi_2, \pi_2', x)$ is active, $\pi_1 = \pi_2 \cdot \pi$, and $\pi_1' = \pi_2' \cdot \pi$. This fact will be exploited in order to determine whether $\sim$ is acyclic in an additional layer of our algorithm called *occur check layer*. This layer consists of $\frac{1}{2}n(n-1)$ threshold units $C(\{\pi, \pi'\})$ – or $C(\pi, \pi')$ for short – with threshold 1, where $\pi \neq \pi'$. Let $C(\pi_1, \pi_1') > C(\pi_2, \pi_2')$ iff there exist $\pi \in \{\pi_1, \pi_1'\}$ and $\overline{\pi} \in \{\pi_2, \pi_2'\}$ such that $\pi > \overline{\pi}$. The units in the occur check layer are connected by links with weight 1 from $C(\pi_1, \pi_1')$ to $C(\pi_2, \pi_2')$ iff $C(\pi_1, \pi_1') > C(\pi_2, \pi_2')$. Now assume that each element $C(\pi, \pi')$ is initially activated if $\pi \sim \pi'$, whereas $C(\pi, \pi')$ is externally inhibited if $\pi \not\sim \pi'$. This inhibition must be strong enough such that an occur check layer unit cannot be activated by other occur check layer units. As a result units representing a cycle will form a stable coalition, whereas units which are not part of a cycle will be deactivated after some time.

As an example consider $\langle f(x, y) = f(g(y), g(x)) \rangle$. A homogeneous DSE-relation exists and is depicted in figure 3(a). Since $1 \sim 21$ and $2 \sim 11$ the units $C(1, 21)$ and $C(2, 11)$ will initially be active. Figure 3(b) shows the occur check layer, where all connections to and from a self-excitatory unit are not drawn. The units $C(1, 21)$ and $C(2, 11)$ are mutually excitatory and form a stable coalition. Notice that $C(21, 11)$ is externally inhibited and, thus, cannot be activated by $C(1, 21)$ and $C(2, 11)$. For our running example $\langle f(x, x, y) = f(g(y), g(g(z)), g(a)) \rangle$ the units $C(1, 2), C(11, 21), C(11, 3), C(21, 3)$, and $C(211, 31)$ are initially active, but are all deactivated after 3 steps.

The behaviour of the occur check layer can be formally verified. Let $C$ be a set of active occur check layer units and $S$ be a transformation on $C$ such that

$$S(C) = C \setminus \{C(\pi_1, \pi_1') \mid \neg\exists C(\pi_2, \pi_2') \in C : C(\pi_1, \pi_1') < C(\pi_2, \pi_2')\}^6.$$

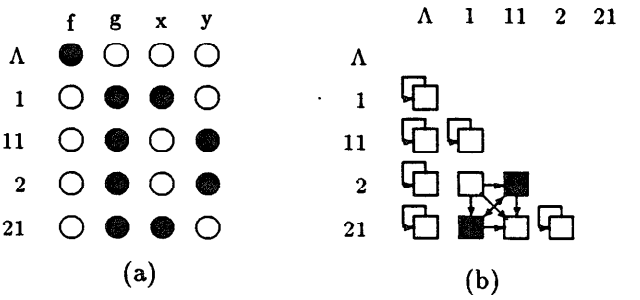


Figure 3: (a) the term layer representing the finest DSE-relation and (b) the occur check layer for $\langle f(x, y) = f(g(y), g(x)) \rangle$.

With this definition we find

$$S(C) \subseteq C$$
$$S(C) = C \Rightarrow \forall k \geq 0 : S^k(C) = C \text{ and}$$
$$\exists k \leq n : S^{k+1}(C) = S^k(C)$$

Now let $C = \{C(\pi, \pi') \mid \pi \sim \pi'\}$ and $C^* = S^k(C)$ such that $S(C^*) = C^*$. In other words, $C$ contains all occur check layer units which are initially active.

**Theorem 3** *$C^* = \emptyset$ iff $\sim$ is acyclic.*

It remains to show how the occur check layer units are initially activated. Therefore, assume that all occur check layer units are externally inhibited. Such an inhibition of a unit $C(\pi_1, \pi_1')$ is blocked and the unit is excited iff there is an active unification layer unit of the form $U(\pi_2, \pi_2', x)$ and an occurrence $\pi$ such that $\pi_1 = \pi_2 \cdot \pi$, and $\pi_1' = \pi_2' \cdot \pi$, where $x$ is a variable. Notice that a unit has to be excited only once whereas the inhibition has to be blocked for the time the occur check layer needs to settle down.

## Discussion

We have presented a connectionist unification algorithm and shown that it is correct and complete. The algorithm does not propagate potential non-unifiability as Stolcke's (1989) algorithm does. This propagation is vital for Stolcke's approach since his algorithm is initialized by activating the unit which represents the fact that the unification problem is solvable. We have tried to add the propagation of potential non-unifiability to our unification algorithm but all examples suggested that there will be no considerable speedup.

Recall that an equivalence relation on a set of occurrences is homogeneous iff each occurrence is labelled with at most one function symbol. Consequently, if the equivalence relation is homogeneous, then in each row of the term layer there is at most one of the units active which represent the function symbols. In this special case a coarse coded representation of the function symbols is possible even without any cross-talk.

[6] $C \setminus C'$ denote the set $C$ minus the set $C'$.

What is this unification algorithm good for besides showing that unification can be implemented in a connectionist system? We would like to apply it within term rewriting, logic programming, or theorem proving in order to tackle the problems mentioned in the introduction. But there are a lot of open problems. We have assumed that all units of our layered architecture are prewired. This is all right as long as the application requires only a bounded number of clauses. If, however, an unbounded number of variants of clauses is needed, then the network has to be recruited dynamically. The unification problem admits a single solution. Unfortunately, as soon as we are interested in unification under certain equational theories, we have to deal with complete sets of unifiers. These sets may be even infinite. Similarly, a problem posed to a theorem prover may admit several solutions. How can we represent multiple solutions and how can we deal with potentially infinite computations in a connectionist theory?

**Acknowledgement:** I would like to thank Jerry Feldman for his guidance and support as well as Andreas Stolcke and Heinz Schmidt, whose comments on earlier versions of the unification algorithm helped to improve it considerably.

## References

Ajjanagadde, V. & Shastri, L. (1989). Efficient Inference with Multi-Place Predicates and Variables in a Connectionist System. In *Proceedings of the Annual Conference of the Cognitive Science Society*, pp. 396–403.

Ballard, D. H. (1986). Parallel Logic Inference and Energy Minimization. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pp. 203 – 208.

Barnden, J. A. (1984). On Short Term Information Processing in Connectionist Theories. *Cognition and Brain Theory*, 7:25–59.

Bibel, W. (1987). *Automated Theorem Proving.* Vieweg Verlag, Braunschweig, second edition.

Citrin, W. V. (1988). Parallel Unification Scheduling in Prolog. Technical Report UCB/CSD 88/415, University of California, Berkeley.

Colmerauer, A. (1984). Equations and Inequations on Finite and Infinite Trees. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pp. 85–99.

Dwork, C., Kannelakis, P. C., & Mitchell, J. C. (1984). On the Sequential Nature of Unification. *Journal of Logic Programming*, 1:35–50.

Fages, F. & Huet, G. (1986). Complete Sets of Unifiers and Matchers in Equational Theories. *Journal of Theoretical Computer Science*, 43:189–200.

Feldman, J. A. & Ballard, D. H. (1982). Connectionist Models and Their Properties. *Cognitive Science*, 6(3):205–254.

Fodor, J. A. & Pylyshyn, Z. W. (1988). Connectionism and Cognitive Architecture: A Critical Analysis. In Pinker & Mehler, eds., *Connections and Symbols*, pp. 3–71. MIT Press.

Herbrand, J. (1930). Sur la Theorie de la Demonstration. In Goldfarb, ed., *Logical Writings (1971).* Cambridge.

Hölldobler, S. (1990). A Connectionist Unification Algorithm. Technical Report TR-90-012, International Computer Science Institute, Berkeley, California.

Kirchner, C. (1984). A New Equational Unification Method: A Generalisation of Martelli-Montanari's Algorithm. In *Proceedings of the Conference on Automated Deduction*, pp. 224–247.

Kowalski, R. (1979). *Logic for Problem Solving*, vol. 7 of *Artificial Intelligence.* North Holland, New York/Oxford.

Lange, T. E. & Dyer, M. G. (1989). Frame Selection in a Connectionist Model of High-Level Inferencing. In *Proceedings of the Annual Conference of the Cognitive Science Society*, pp. 706–713.

MacQueen, D., Plotkin, G. D., & Sethi, R. (1984). An Ideal Model for Recursive Polymorphic Types. In *Proceedings of the ACM Symposium on Principles of Programming Languages.*

Mjolsness, E., Gindi, G., & Anandan, P. (1989). Optimization in Model Matching and Perceptual Organization. *Neural Computation*, 1:218–229.

Paterson, M. S. & Wegman, M. N. (1978). Linear Unification. *Journal of Computer and System Sciences*, 16:158–167.

Robinson, J. A. (1965). A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12:23–41.

Smolensky, P. (1987). On Variable Binding and the Representation of Symbolic Structures in Connectionist Systems. Technical Report CU-CS-355-87, Department of Computer Science & Institute of Cognitive Science, University of Colorado.

Smolensky, P. (1988). On the Proper Treatment of Connectionism. *Behavioral and Brain Sciences*, 11:1–74.

Stolcke, A. (1989). Unification as Constraint Satisfaction in Structured Connectionist Networks. *Neural Computation*, 1(4):559 – 567.

Touretzky, D. S. & Hinton, G. E. (1988). A Distributed Connectionist Production System. *Cognitive Science*, 12:423 – 466.

Vitter, J. S. & Simons, R. A. (1986). New Classes for Parallel Complexity: A Study of Unification and other complete Problems for P. *IEEE Transactions on Computers*, pp. 403–418.