A Connectionist Unification Algorithm

Steffen Hölldobler[1]

TR-90-012

March 15, 1990

Abstract

Unification plays an important role in many areas of computer science, mathematical logic, and artificial intelligence. It is also at the heart of connectionist models concerned with knowledge representation and inference. However, most of these models are severly restricted by their *propositional fixation* as they are defined over a finite set of constants and predicates. This restriction is caused by the inability to unify terms built from function symbols, constants and variables. In this paper 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.

[1]on leave from FG Intellektik, FB Informatik, TH Darmstadt, West-Germany

1 Introduction

Following Leibnitz's and Frege's idea to formalize human thought Herbrand, Gödel, and Skolem developed predicate logic by 1930. In the years to follow 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 [Kowalski, 1979] connection graphs or Bibel's [Bibel, 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 [Herbrand, 1930] and which was formally introduced by Robinson [Robinson, 1965].

Despite their success automatic theorem provers and logic programming languages are still plagued be several problems. Most of them lack a *clever* proof strategy that takes into account domain specific knowledge of an application. Most of them do not exhibit common sense. Most of them cannot deal efficiently with equational axioms. This list of problems is far from being complete, but we want to mention just one more problem which we intend to tackle in this paper. Most theorem provers do not explore the parallelism inherent in logic since they are still designed with the sequential von-Neumann computer in mind.

The sequential characteristics of a conventional 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.

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 [Smolensky, 1988] has emphasized that such a spreading of activation *cannot be adequate for complex tasks such as question answering or grammticality judgements.* And in an earlier paper Smolensky [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 the processes are structure sensitive [Fodor and 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.

This report is organized as follows. After an informal introduction of the unification problem we review the connectionist approaches taken so far to deal with unification. Most of these

approaches are not specifically concerned with unification but they embody some restricted form of it. In fact, the only other attempt to capture unification as a whole is Stolcke's [1989a] unification algorithm for feature structures.

There exist a variety of unification algorithms for conventional von Neumann computers (see e.g. [Paterson and Wegman, 1978; Martelli and Montanari, 1982; Corbin and Bidoit, 1983]) or for PRAMs ([Vitter and Simons, 1986; Ito *et al.*, 1985; Singhal and Patt, 1989]) and we want to make use of their techniques. Our connectionist unification algorithm is based on Paterson's & Wegman's observation, that the most general unifier of two terms can be characterized by the finest valid equivalence relation defined on the nodes of a directed acyclic graph (or *dag* for short). In section 3 we formally define the unification problem and identify a suitable representation for it.

In the main part of the paper, section 4, we develop the connectionist unification algorithm. In particular, we show how the unification problem can be represented by a connectionist network and how the finest valid equivalence relation for a unification proplem can be computed locally by the units this network. These units are simple threshold units in the sense of [Feldman and Ballard, 1982] and no central controller is needed. Moreover, we prove rigorously that the connectionist model solves the unification problem. The connectionist unification algorithm 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. From [Dwork *et al.*, 1984] we learn that the unification problem is logspace complete and, hence, we cannot expect that a parallel unification algorithm solves a unification problem U in the worst case in a time which is not bounded by the size of U. However, as we will show the time complexity for the average case is much better and the word and matching problem can always be solved in 2 steps.

The paper concludes by pointing out some open problems and future work.

2 Approaches Taken

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. A unification algorithm can already be found in Herbrand's [1930] thesis though the notion unification has formally been introduced by J. A. Robinson 35 years later. Since unification is central to the resolution principle, the properties of unification have been studied intensively in subsequent years. Paterson & Wegman [1978] have specified an algorithm which solves the unification problem on a conventional von-Neumann computer with time complexity $O(e+n)$, where n is the number of nodes and e is the number of edges in a directed acyclic graph representing the unification problem. Dwork et. al. [1984] have proved that the unification problem 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 unification algorithm improves the time-complexity for the average case. In fact – as we will show in sections 4.3 and 4.4 – the unification problem can be solved in 2 steps if it degenerates to a word or a matching problem, where the word problem is the

question of whether s is equal to t and the matching problem is the question of whether an instance of s is equal to t. 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.

The approaches taken so far to implement a parallel unification algorithm can be divided into the approaches aiming at a distributed computation and the approaches aiming at a connectionist computation of the most general substitution (called *unifier*) which unifies two terms. Distributed unification algorithms such as the one developed by Vitter & Simons [1986] or Hager & Moser [1989] make use of a dag-representation of the unification problem. The parallelism exploited by these algorithm has its source in the decomposability or correspondence axioms which characterize a valid equivalence relation. These axioms essentially state that two terms $f(s_1, \ldots, s_n)$ and $f(t_1, \ldots, t_n)$ are unifiable if we find for all $1 \leq i \leq n$ that s_i and t_i are unifiable. The unifiability of the terms s_i and t_i is now determined in parallel as far as this is possible.

Most of the approaches taken so far to implement a unification or matching algorithm in a connectionist setting are parts of the design of larger inference systems like production systems. In the sequel we concentrate on the unification algorithm and describe these approaches only as far as unification or matching is concerned. For further information the reader is referred to the references.

2.1 Ballard's Parallel Logical Inference

Ballard [1986] is concerned with the problem of how a restricted form of resolution can be implemented using a parallel relaxation algorithm. Clauses may be used at most once in a proof and one of two parent clauses must always be a unit clause. Terms are just constants and variables. For a given set of clauses all possible substitutions are prewired and incompatible bindings are connected via inhibitory links. The basic idea is to incrementally deactivate those parts of the network which correspond to pairs of clauses which resolve. If the entire network is eventually deactivated a resolution proof has been found.

The fact that all possible substitutions are prewired restricts the applicability of Ballard's technique considerably. If arbitrary function symbols are allowed, the set of possible substitutions becomes infinite and, hence, cannot be prewired. A similar problem arises from the restriction that clauses may be used at most once in a proof. A resolution proof generally requires several new variants of a clause. These variants are obtained by renaming the variables in the clauses. Hence, we have to deal with infinitely many variables which makes it impossible to anticipate all possible substitutions.

2.2 Touretzky's & Hinton's DCPS

Touretzky's & Hinton's [1988] DCPS is a connectionist interpreter for a restricted class of production system based on distributed representations. A finite set of constants and essentially one variable comprise the set of terms. The hypothesis of each production rule

consists of two triples of terms, which are either ground or of the form $(x\ a\ b)$ and $(x\ c\ d)$. In other words, if the hypothesis of a rule contains variables at all, it contains only one variable and this one occurs as the first argument of the first and second triple. Touretzky's & Hinton's production system is further restricted by assuming that at any time only one hypothesis of a rule successfully matches against the content of the working memory. There is no conflict.

The working memory of the DCPS is a set of binary state units. It is coarse-coded such that each unit represents a set of triples. A certain triple is stored in the working memory by activating all units which vote – among others – for this triple. To find out which hypothesis of a production rule matches the working memory so-called *clause spaces* are introduced: one for the first and one for the second triple in the hypothesis of a production rule. There is a bidirectional excitatory connection between the units of the working memory and the clause spaces. Similarly, rule units representing the triples in the hypothesis of the production rules are bidirectional excitatory connected with the respective units in the clause space. Thus, two atoms in the working memory which together form the hypothesis of a production rule activate via the clause spaces the respective rule and vice versa. Other solutions are suppressed by inhibitory connections between the units of the clause space and between the units representing different rules.

So far we have considered only ground production rules. The effect of using variables in Touretzky's & Hinton's production systems is to constrain the hypothesis of the rules such that the first constant in both triples is the same in order to fire the rule. Within the *bind space* the constants are coarse-coded and bidirectional excitatory connected to the respective units of the clause space. That is, a bind unit representing – among others – the constant a is connected to all units in the clause space who represent – among others – triples whose first element is the a.

Now, a matching rule is found if a minimum energy state is found. This can be done by constraint satisfaction if the production system is modeled by a Hopfield network and by simulated annealing if it is modeled by Boltzmann machines. For more details see [Touretzky and Hinton, 1988].

Thus, Touretzky & Hinton have developed a matching algorithm for a very special form of terms. While it seems to be possible to relax the constraints on the occurrences of variables in the hypothesis of a production rule it is by no means obvious how their technique can be applied if n-ary function symbols, $n > 0$, are allowed or if the elements of the working memory may also contain variables.

2.3 Anandan's, Mjolsness' and Gindi's Frameville

Frameville is a connectionist frame system supporting the binding of variables, the allocation of frames, and equality. It is especially concerned with the dynamic creation of concepts and relations between them. Frameville distinguishes between a static, possibly hierarchically ordered set of concepts, called the *model side*, and a dynamic set of ground formulae, called the *data side*, and determines whether the data side matches the model side. The model as

well as the data side can be regarded as an directed acyclic graph, but whereas the data side is ground, the model side may contain variables. The problem now is to find a substitution for the variables in the model side such that it matches the data side. This is done by specifying a distance metric or objective function reflecting the mismatch between the dags. The minimization of such a function is performed by an analog neural network [Anandan *et al.*, 1989; Mjolsness *et al.*, 1989]. Thus, Frameville solves a matching problem. However, it is not obvious that Frameville always finds the most general matcher if it exists and it seems to be impossible to extend Frameville in order to deal with unification problems.

2.4 Lange's and Dyer's Robin

Robin (ROle Binding and Inferencing Network) is a hybrid localist spreading-activation inference system that handles dynamic bindings via marker-passing [Lange and Dyer, 1989a; Lange and Dyer, 1989b]. The authors are especially concerned with the problem that an evidential activation of a conceptual unit in a localist model gives no clue as to where the evidence came from. This problem is treated by passing markers.

Terms are again just constants and variables. Each constant has associated a unit in the network outputting a uniquely-identifying value, called its *signature*. Dynamic bindings are created by passing these values. As an example consider the rule

$$P(x) \Leftarrow Q(x)$$

and the query

$$\Leftarrow P(a).$$

The rule is represented in a two-layered architecture as shown in the figure 1. The rule is represented in the lower layer and the the values for the variables are represented in the upper layer.

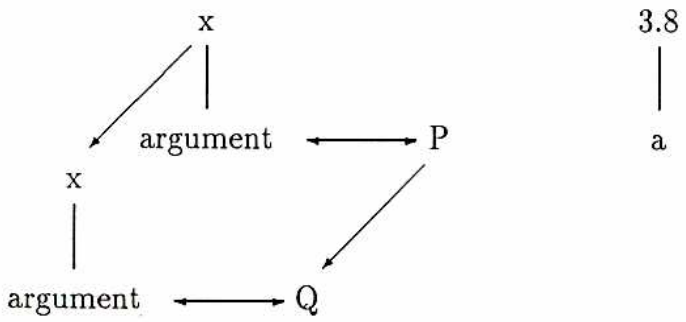

Figure 1: The representation of the rule $P(x) \Leftarrow Q(x)$ and the constant a with signature 3.8 in Robin.

Suppose that the unit corresponding to the constant a has signature 3.8. As soon as the query is posed the unit representing P becomes active and at the same time the unit x

corresponding to the argument of P receives the value 3.8. The argument of P is bound to a since it has the same value as a. As the activation spreads from P to Q, the value 3.8 is passed from the argument of P to the argument of Q resulting in the net depicted in figure 2.

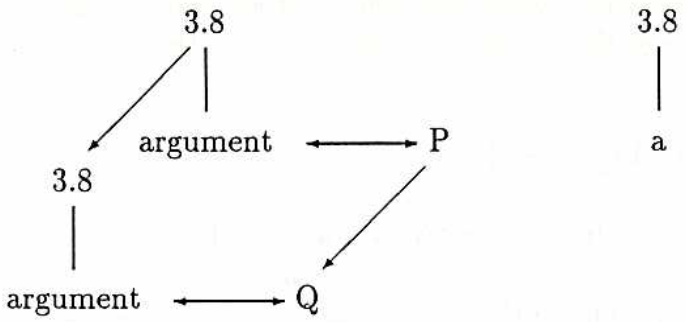

Figure 2: The representation of the rule $P(x) \Leftarrow Q(x)$ and the constant a with signature 3.8 in Robin after the query $\Leftarrow P(a)$ was processed.

Lange and Dyer show how this technique can be applied to draw inferences from a given frame instantiation and how related frames can be instantiated. Thus, they have developed a simple matching algorithm, where constants are matched against variables by passing their unique marker. However, their technique cannot be applied to develop a connectionist unification algorithm since there are generally infinitely many terms and we cannot simply assign a marker to each of them. Lange & Dyer's units are comparatively complex units which handle and compare signatures. The messages between these units may also be signatures and, thus, are also complex.

2.5 Ajjanagadde's and Shastri's Inference System

In a recent work Ajjanagadde & Shastri [1989] demonstrate how a restricted form of rules and facts involving multi-place predicates can be expressed in a connectionist system and how limited, but sound, inferences can be efficiently drawn from this knowledge.

Terms are again either constants or variables. The basic idea underlying Ajjanagadde's & Shastri's approach is similar to the one used by Dyer & Lange. Whereas the latter use markers to distinguish constants, the former use a phased clock and assign a unique phase to each constant. These phases are then used to bind variables to constants, where a variable is bound to a certain constant if the phase of the constant is assigned to it [Shastri and Ajjanagadde, 1989].

Shastri's & Ajjanagadde's inference system has several limitations. Since it seems to be biologically implausible that a *phase-sensitive* neuron has more than 10 phases (see [Shastri and Ajjanagadde, 1990]), an initial query may not contain more than 10 constants. If we are not only interested in a *yes/no* answer but also in the bindings for the variables occurring in the initial query, then the query may not contain more than 10 variables and constants.

The system can handle multiple occurrences of the same variable only if this variable is instantiated by the initial query. As far as unification is concerned this means that terms like $f(x, x)$ and $f(y, z)$ cannot be unified. Furthermore, in a proof a clause can be used only once. Finally, as with Dyer's & Lange's approach it is not easy to see how Ajjanagadde's & Shastri's unification algorithm can be extended to cope with function embeddings which arise during the unification process if n-ary function symbols, $n > 0$, are allowed.

2.6 Stolcke's Unification Algorithm for Feature Structures

Stolcke's [1989b; 1989c] interest in unification stems from unification-based grammar formalisms used within linguistics (see e.g. [Shieber, 1986]). In this field *feature structures* are used to represent the features associated with a linguistic entity. Moreover, linguistic constraints and operations on these entities can easily be expressed by unifying the respective feature structures. Since each term can be represented by a feature structure, a feature unification algorithm must solve the unification problem.

In Stolcke's approach, feature terms are represented as dags. Inspired by Paterson & Wegman [1978] the author defines a valid equivalence relation on the nodes of a dag and computes this relation via a connectionist network. Therefore, each pair of nodes in the dag representation of the unification problem is represented by a unit in the connectionist model indicating whether the dag-nodes are equivalent or not. To ensure that the relation is indeed a valid equivalence relation the axioms of transitivity and decomposability are explicitly encoded into units and links in the connectionist model. For example, let $x \sim y$ and $y \sim z$ be the units indicating that x is equal to y and y is equal to z. If these units are activated, the unit $y \sim z$ must also be activated. Similarly, suppose that v and w represent the terms $g(s)$ and $g(t)$ and that x and y represent the terms s and t. Now, if unit $v \sim w$ is activated, then unit $x \sim y$ must also be activated. Conversely, the contrapositives of the axioms of transitivity and decomposability deactivate units. For example, if unit $x \not\sim y$ is activated, then unit $v \sim w$ is deactivated, where x, y, v, and w represent the same terms as before.

The unification algorithm is triggered by activation of the unit $s \sim t$, where s and t are the root units of the terms to be unified. If this unit remains active, then the unification is succesful. Otherwise, the unification has failed. Since units must eventually be deactivated, the propagation of non-unifiability is vital for Stolcke's algorithm.

However, Stolcke's algorithm will unify the terms x and $f(x)$, since the so-called *occur check* is not performed. In other words, if a variable x is to be unified with a term t then Stolcke does not check whether x occurs in t. It is not obvious how the algorithm can be changed such that an occur check is performed. As usual in unification-based grammar formalism Stolcke is not so much interested in the most general unifier σ for two terms s and t, but in the term σs. Consequently, his algorithm generates this term and it is not obvious how to extract the most general unifier from it.

3 The Unification Problem

In this section we formally define the unification problem and identify several problems that have to be solved in order to obtain an algorithm which solves the unification problem. These problems are concerned with the representation of terms and variable bindings as well as with the operations upon this data structures.

We assume to have an *alphabet* consisting of a finite set F of graded *function symbols* and a finite set V of *variables*. 0-ary function symbols are often called *constants*. A *term* is either a constant, a variable, or an n-ary function symbol f applied to the terms $t_1, \ldots, t_n$. A term is said to be *ground* if it does not contain a variable.

A *substitution* is defined to be a mapping from the set of variables into the set of terms which is equal to the identity almost everywhere and is represented by

$$\{x_1 \leftarrow t_1, \ldots, x_n \leftarrow t_n\},$$

where the t_i's denote terms different from the x_i's. Substitutions are extended to morphisms on the set of terms in the usual way.

Throughout the paper we make use of the notation depicted in the following table in the sense that, whenever we refer to x – for example – we implicitly assume that x is a variable.

$a, b, \ldots$	constants	$s, t, \ldots$	terms	π	occurrence
$f, g, \ldots$	function symbols	$z, y, \ldots$	variables	$\sigma, \theta, \ldots$	substitutions

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 σ such that $\sigma s = \sigma t$. If such a substitution σ exists then σ is called *unifier* of $\langle s = t \rangle$. A substitution σ is said to be a *most general unifier* for $\langle s = t \rangle$ iff for each unifier θ for $\langle s = t \rangle$ we find a substitution λ such that $\theta = \lambda\sigma$. It is well-known that the unification problem is decidable and that a most general unifier of two terms exists and can effectively be computed whenever the terms are unifiable [Robinson, 1965; Martelli and Montanari, 1982; Paterson and Wegman, 1978]. Such a most general unifier is unique modulo variable renaming [Fages and Huet, 1986] and, therefore, is often called *the* most general unifier.

To give some examples consider the unification problem

$$\langle f(g(x), h(y, x)) = f(g(z), h(z, a)) \rangle.$$

It is solvable with most general unifier

$$\{x \leftarrow a, y \leftarrow a, z \leftarrow a\}.$$

On the other hand,

$$\langle x = g(x) \rangle$$

is unsolvable, since x cannot be equal to a term which contains x as a proper subterm. This problem is often referred to as *occur check problem* since we have to check whether x occurs in $g(x)$ in order to solve the unification problem.

Since the occur check is a very costly operation it has been omitted in virtually all logic programming systems. As a consequence many Prolog systems exhibit an unspecified behaviour when encountering an occur check problem (see [Marriott and Sondergaard, 1988]). Colmerauer [1982; 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, terms like x and $g(x)$ are unifiable by replacing the x by the infinite tree $g(g(g(\ldots)))$.

In analogy to the unification problem we define the *matching problem* $\langle s \geq t \rangle$. It consists of two terms s and t and is the problem whether there exists a substitution σ such that $\sigma s = t$. In other words, only the variables occurring in one term can be replaced in an attempt to make the terms equal. The notions *matcher* and *most general matcher* are defined in analogy to the notions unifier and most general unifier. Note, for the matching problem it is irrelevant whether terms are interpreted over the domain of finite or infinite trees.

Finally, the *word problem* consists of two terms s and t and is the problem of whether s and t are syntactically equal, i.e. whether $s = t$ holds.

In order to specify a connectionist unification algorithm we have to find a suitable representation for the unification problem and the substitutions encountered during the computation of the most general unifier. We have also to define the operations which compute the most general unifier if the unification problem is solvable.

3.1 Representation

As pointed out by Smolensky [1987] and many others, representation is a crucial component, *for a poor representation will often doom the model to failure, and an excessively generous representation may essentially solve the problem in advance.*

3.1.1 Terms

Our underlying alphabet consists of a finite set of function symbols and variables. As long as we are only interested in solving the unification problem this is a perfect assumption. However, if we intend to built the unification algorithm into a theorem prover then the underlying alphabet may generally contain a countable number of variables, function and predicate symbols. These sets can generally not be restricted to finite ones as, for example, the proof of the lifting lemma in a Horn clause calculus requires to have *new* constants at our disposal [Lloyd, 1984]. On the other hand, whenever a logic program and a certain query are given, then only function and predicate symbols occurring in the program and the query will be used to compute an answer. The problem to deal with potentially infinitely many variables remains. In any case the set of (ground) terms is countably infinite. As a consequence, we cannot simply assign a different marker, signature or phase with each term. How are we going to represent terms?

3.1.2 Variable Bindings

A unification algorithm should produce several results. First of all we are interested in whether two terms are unifiable at all. However, a simple *yes* or *no* will often not suffice. Since the terms to be unified are often parts of larger expressions we are also interested in the most general unifier of two terms if they are unifiable. This unifier is usually generated by the unification algorithm and returned as output. How are we going to represent these bindings?

Since the set of substitutions is infinite we cannot simply mark certain units in a network as *response units* [Shastri, 1988] for each possible unifier. However, the most general unifier for a unification problem is the identity except for (some of) the variables occurring in the unification problem. Hence, we may identify certain units as representing these variables. After the net has performed the unification the bindings for these variables should be accessed via these units.

3.1.3 Consistent Variable Bindings

If a term contains more than one occurrence of a variable and this variable is bound to a certain term, then all these occurrences have to be consistently replaced by the new term. This poses a problem to many connectionist systems (see [Barnden, 1984]). It is no solution to this problem to instantiate all expressions (e.g. production rules or program clauses) and, thus, to eliminate variables since there are infinitely many ground terms.

3.2 The Operations

Of course we want the connectionist model to decide the unification problem. To identify the necessary operations which perform this task we look at an axiomatization of unification. Such an axiomatization can essentially be found in Herbrand's [1930] thesis and has also been investigated by Paterson & Wegman [1978], Martelli & Montanari [1982], or Maher [1988]. To define this axiomatization it will be necessary to have a formal way to deal with subterms. Therefore, we introduce the notion of an occurrence.

The set of *occurrences* (or *positions*) 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 $1 \leq i \leq n$. For example, the term $f(y, g(y))$ has the occurrences Λ, 1, 2, and 2·1, where the Λ has been omitted whenever it was not the only symbol. In other words, an occurrence is a finite string over natural numbers, where $\cdot$ denotes concatenation and Λ denotes the empty string. On occurrences we define a partial ordering by $\pi_1 \geq \pi_2$ iff there exists a π_3 such that $\pi_1 \cdot \pi_3 = \pi_2$. Furthermore, $\pi_1 > \pi_2$ iff $\pi_1 \geq \pi_2$ and $\pi_1 \neq \pi_2$.

With the help of the notion of occurrences we are now in a position to define subterms. For a term t and occurrence $\pi \in O(t)$ the *subterm of t at π*, $t|\pi|$, is inductively defined as $t|\Lambda| = t$ and $f(t_1, \ldots, t_i, \ldots, t_n)|i \cdot \pi| = t_i|\pi|$. For example, the term $f(y, g(y))$ has the subterms y and $g(y)$ at occurrences 1, 2·1 and 2, respectively.

We can now define the notion of a label for each occurrence in the terms of a unification problem $\langle s = t \rangle$. Informally, each occurrence π is labeled by the variables and function symbols that occurs at π in s and t. These labels play a central role in our connectionist unification algorithm. We will show that the most general unifier for a unification problem can be represented by these labels. Let $O = O(s) \cup O(t)$ and recall that V and F denote the set of variables and function symbols, respectively. For all $\pi \in O$ we define

- $label_v(\pi) = \{x \in V \mid s|\pi| = x \vee t|\pi| = x\}$
- $label_f(\pi) = \{f \in F \mid s|\pi| = f(s_1, \ldots, s_n) \vee t|\pi| = f(t_1, \ldots, t_n)\}$ and
- $label(\pi) = label_v(\pi) \cup label_f(\pi)$.

For example, if the unification problem consists of the terms $f(x, x, y)$ and $f(g(y), g(g(z)), g(a))$, the sets of labels are as follows.

	$label_f$	$label_v$	$label$
Λ	$\{f\}$	$\emptyset$	$\{f\}$
1	$\{g\}$	$\{x\}$	$\{g, x\}$
1·1	$\emptyset$	$\{y\}$	$\{y\}$
2	$\{g\}$	$\{x\}$	$\{g, x\}$
2·1	$\{g\}$	$\emptyset$	$\{g\}$
2·1·1	$\emptyset$	$\{z\}$	$\{z\}$
3	$\{g\}$	$\{y\}$	$\{g, y\}$
3·1	$\{a\}$	$\emptyset$	$\{a\}$

With the help of these technical definitions we can now set up the operations which enable us to compute the solution of a unification problem. As Paterson & Wegman [1978] have shown the most general unifier for a unification problem can be determined via a so-called *valid* equivalence relation on the nodes of a dag representing the unification problem. We will essentially use their approach to specify a connectionist unification algorithm, but there is one major difference. We will not base our algorithm on a dag representation but on an occurrence-label representation of the unification problem. Thus we have to specify an axiom which corresponds to a shared node in a dag representation.

Formally, an *equivalence relation* $\sim$ on $O = O(s) \cup O(t)$ is a relation which is

- *reflexive*: $\forall \pi \in O : \pi \sim \pi$,
- *symmetrical*: $\forall \pi_1, \pi_2 \in O : \pi_1 \sim \pi_2 \Rightarrow \pi_2 \sim \pi_1$, and
- *transitive*: $\forall \pi, \pi_1, \pi_2 \in O : \pi_1 \sim \pi \wedge \pi \sim \pi_2 \Rightarrow \pi_1 \sim \pi_2$.

Furthermore, $\sim$ is said to be

- *decomposable* 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 $\forall \pi_1, \pi_2 \in O : \pi_1 \sim \pi_2 \Rightarrow |label_f(\pi_1) \cup label_f(\pi_2)| \leq 1$,
- *singular* iff $\forall \pi_1, \pi_2 \in O : label_v(\pi_1) \cap label_v(\pi_2) \neq \emptyset \Rightarrow \pi_1 \sim \pi_2$.

A decomposable and singular equivalence relation is called *DSE-relation*. Intuitively, decomposability states that corresponding subterms have to be equal under $\sim$. $\sim$ is homogeneous iff each occurrence is labelled with at most one function symbol, whereas singularity states that two occurrences labelled with the same variable are equal under $\sim$. For the unification

problem $\langle f(x,x,y) = f(g(y),g(g(z)),g(a))\rangle$ a homogeneous DSE-relation is defined by the equivalence classes

$$[\Lambda],[1,2],[1{\cdot}1,2{\cdot}1,3],\ [2{\cdot}1{\cdot}1,3{\cdot}1].$$

In the literature these definitions are sometimes slightly altered. The decomposability is called *correspondence* in e.g. [Dwork *et al.*, 1984]. In [Kirchner, 1984] the definition of decomposability forces the relation to be homogeneous. As mentioned Paterson & Wegman [1978] represent the unification problem as a directed acyclic graph, where common subterms and especially common variables are shared. Thus, they have not to specify singularity explicitly.

For $\sim$-equivalence classes C_1 and C_2 we define $C_1 \succ C_2$ iff $\exists \pi_1 \in C_1, \pi_2 \in C_2 : \pi_1 > \pi_2$. With this ordering we find

$$[\Lambda] \succ [1,2] \succ [1{\cdot}1,2{\cdot}1,3] \succ [2{\cdot}1{\cdot}1,3{\cdot}1]$$

However, the equivalence classes defined by a homogeneous DSE-relation on the set of occurrences of a unification problem are not always partially ordered. The interested reader is encouraged to verify that for the unification problem $\langle f(x,y) = f(g(y),g(x))\rangle$ we find that

$$[1,2{\cdot}1] \succ [1{\cdot}1,2] \succ [1,2{\cdot}1].$$

An equivalence relation $\sim$ on O is said to be *acyclic* iff the $\sim$-equivalence classes are partially ordered by $\succ$. A homogeneous and acyclic DSE-relation on O is often called *valid*. The following proposition is an immediate consequence of [Paterson and 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 $O(s) \cup O(t)$.*

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

Moreover, Paterson & Wegman have shown how the most general unifier of two terms can be constructed from the finest valid equivalence relation (i.e. the finest homogeneous and acyclid DSE-relation) on the set of occurrences of the unification problem. Their technique will be reviewed in section 4.6. For the moment we will tackle the problem of how to compute the finest valid equivalence relation.

4 A Connectionist Unification Algorithm

The stage is set to specify our connectionist unification algorithm. As mentioned we intend to represent the unification problem as the sets of labels attached to the occurrences of the

unification problem. The first question which arises is whether this is a suitable representation to solve the unification problem or, in other words, whether these labels can be used to represent a DSE-relation. A DSE-relation on the occurrences of a unification problem can be characterized by its equivalence classes and each equivalence class can be represented by the union of the labels of its members. More importantly, these unions can be computed by two simple operations derived from the axioms of decomposability and singularity. This derivation will be motivated in the following subsection. Thereafter we will show 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 so far. Finally, we show how the most general unifier can be determined by the labels attached to each DSE-equivalence class.

4.1 The Representation of a DSE-Relation

We start with the development of the representation for a DSE-relation on $O = O(s) \cup O(t)$. In the sequel let $\sim$ be such an equivalence relation. For a $\sim$-equivalence class C we define

- $label_v(C) = \bigcup_{\pi \in C} label_v(\pi)$
- $label_f(C) = \bigcup_{\pi \in C} label_f(\pi)$
- $label(C) = label_v(C) \cup label_f(C)$.

For notational convenience we denote an equivalence class C by $[\pi_1, \ldots, \pi_n]$ whenever $\{\pi_1, \ldots, \pi_n\} \subseteq C$. As an example consider again the unification problem

$$\langle f(x, x, y) = f(g(y), g(g(z)), g(a)) \rangle$$

and the homogeneous DSE-relation for this problem presented in the previous section. We have

	$label_f$	$label_v$	$label$
$[\Lambda]$	$\{f\}$	$\emptyset$	$\{f\}$
$[1, 2]$	$\{g\}$	$\{x\}$	$\{g, x\}$
$[1{\cdot}1, 2{\cdot}1, 3]$	$\{g\}$	$\{y\}$	$\{g, y\}$
$[2{\cdot}1{\cdot}1, 3{\cdot}1]$	$\{a\}$	$\{z\}$	$\{a, z\}$

Thus, each $\sim$-equivalence class is represented by the union of the labels of its members. It is important to note that this representation is not unique which can be examplified by the unification problem

$$\langle f(x, x, y, y) = f(g(g(z)), g(g(a)), g(g(b)), g(g(b))) \rangle.$$

Here we find

	$label_f$	$label_v$	$label$
$[\Lambda]$	$\{f\}$	$\emptyset$	$\{f\}$
$[1,2]$	$\{g\}$	$\{x\}$	$\{g,x\}$
$[1{\cdot}1,2{\cdot}1]$	$\{g\}$	$\emptyset$	$\{g\}$
$[1{\cdot}1{\cdot}1,2{\cdot}1{\cdot}1]$	$\{a\}$	$\{z\}$	$\{a,z\}$
$[3,4]$	$\{g\}$	$\{y\}$	$\{g,y\}$
$[3{\cdot}1,4{\cdot}1]$	$\{g\}$	$\emptyset$	$\{g\}$
$[3{\cdot}1{\cdot}1,4{\cdot}1{\cdot}1]$	$\{b\}$	$\emptyset$	$\{b\}$

and $[1{\cdot}1,2{\cdot}1] \neq [3{\cdot}1,4{\cdot}1]$ but $label([1{\cdot}1,2{\cdot}1]) = \{g\} = label([3{\cdot}1,4{\cdot}1])$. However, the following proposition is an immediate consequence of the singularity of $\sim$.

Proposition 2 *If C_1 and C_2 are different $\sim$-congruence classes then*

$$\mathrm{label}_v(C_1) \cap \mathrm{label}_v(C_2) = \emptyset.$$

We intend to represent the $\sim$-equivalence class containing π by $label([\pi])$. Therefore, the question is how can we generate $label([\pi])$? The following lemma will show us how to achieve this goal. Let A contain the axioms of reflexivity, symmetry, transitivity, decomposability, and singularity. By $A \vdash^k \pi_1 \sim \pi_2$ we denote that $\pi_1 \sim \pi_2$ was derived from A in k steps. To prove this lemma we need the notion of the *depth* of an occurrence π, $|\pi|$, which is inductively defined as $|\Lambda| = 0$ and $|i{\cdot}\pi| = |\pi| + 1$.

Lemma 3 *Let $\pi_1 \neq \pi_2$.*

$$A \vdash^k \pi_1 \sim \pi_2$$
$$\Leftrightarrow$$
$$\exists \pi, \pi_1', \pi_2' : A \vdash^{k-|\pi|} \pi_1' \sim \pi_2' \wedge \pi_1 = \pi_1'{\cdot}\pi \wedge \pi_2 = \pi_2'{\cdot}\pi \wedge \mathrm{label}_v([\pi_1', \pi_2']) \neq \emptyset$$

Proof: The if-half can easily be proved by induction on $|\pi|$ using $|\pi|$ applications of the axiom of decomposability. We turn to the only-if-half, which is proved by transfinite induction on k. Suppose the result holds for all $l < k$ and assume that $\pi_1 \neq \pi_2$ and $A \vdash^k \pi_1 \sim \pi_2$. We distinguish 4 cases with respect to the axiom used in the last step of the derivation. Note that the axiom of reflexivity was not applied since this implies $\pi_1 = \pi_2$.

1. If the axiom of symmetry was applied, then

   $$A \vdash^{k-1} \pi_2 \sim \pi_1.$$

   By the induction hypotheses we find π, π_2', π_1' such that

   $$A \vdash^{k-1-|\pi|} \pi_2' \sim \pi_1' \wedge \pi_2 = \pi_2'{\cdot}\pi \wedge \pi_1 = \pi_1'{\cdot}\pi \wedge label_v([\pi_2', \pi_1']) \neq \emptyset.$$

   Applying the axiom of symmetry we conclude

   $$A \vdash^{k-|\pi|} \pi_1' \sim \pi_2' \wedge \pi_1 = \pi_1'{\cdot}\pi \wedge \pi_2 = \pi_2'{\cdot}\pi \wedge label_v([\pi_1', \pi_2']) \neq \emptyset$$

   which proves this case.

2. If the axiom of transitivity was applied, then we find π_3 such that

$$A \vdash^{k-1} \pi_1 \sim \pi_3 \wedge A \vdash^{k-1} \pi_3 \sim \pi_2.$$

By the induction hypotheses we find π, π_1', π_3' such that

$$A \vdash^{k-1-|\pi|} \pi_1' \sim \pi_3' \wedge \pi_1 = \pi_1' \cdot \pi \wedge \pi_3 = \pi_3' \cdot \pi \wedge label_v([\pi_1', \pi_3']) \neq \emptyset$$

and $\overline{\pi}, \overline{\pi}_3', \overline{\pi}_2'$ such that

$$A \vdash^{k-1-|\overline{\pi}|} \overline{\pi}_3' \sim \overline{\pi}_2' \wedge \pi_3 = \overline{\pi}_3' \cdot \overline{\pi} \wedge \pi_2 = \overline{\pi}_2' \cdot \overline{\pi} \wedge label_v([\overline{\pi}_3', \overline{\pi}_2']) \neq \emptyset.$$

Without loss of generality we may assume that $|\pi| < |\overline{\pi}|$. But now we find a π^* such that $\overline{\pi} = \pi^* \cdot \pi$, $\pi_3' = \overline{\pi}_3' \cdot \pi^*$, and $|\pi| = |\overline{\pi}| + |\pi^*|$. With $\pi_2' = \overline{\pi}_2' \cdot \pi^*$ we find

$$A \vdash^{k-1-|\pi|} \pi_3' \sim \pi_2'$$

by $|\pi^*|$ applications of the axiom of substitutivity and with the axiom of transitivity we conclude

$$A \vdash^{k-|\pi|} \pi_1' \sim \pi_2'$$

which proves this case.

3. If the axiom of decomposability was applied, then we find i, $\overline{\pi}_1$, and $\overline{\pi}_2$ such that

$$A \vdash^{k-1} \overline{\pi}_1 \sim \overline{\pi}_2$$

and

$$\pi_1 = \overline{\pi}_1 \cdot i \wedge \pi_2 = \overline{\pi}_2 \cdot i.$$

By the induction hypothesis we find $\overline{\pi}, \overline{\pi}_1', \overline{\pi}_2'$ such that

$$A \vdash^{k-1-|\overline{\pi}|} \overline{\pi}_1' \sim \overline{\pi}_2' \wedge \overline{\pi}_1 = \overline{\pi}_1' \cdot \overline{\pi} \wedge \overline{\pi}_2 = \overline{\pi}_2' \cdot \overline{\pi} \wedge label_v([\overline{\pi}_1', \overline{\pi}_2']) \neq \emptyset.$$

Hence,

$$A \vdash^{k-1-|\overline{\pi}|} \overline{\pi}_1' \sim \overline{\pi}_2' \wedge \pi_1 = \overline{\pi}_1 \cdot \overline{\pi} \cdot i \wedge \pi_2 = \overline{\pi}_2 \cdot \overline{\pi} \cdot i \wedge label_v([\overline{\pi}_1', \overline{\pi}_2']) \neq \emptyset,$$

which proves the lemma in this case.

4. If the axiom of singularity was applied, then we find

$$label_v(\pi_1) = label_v(\pi_2).$$

Hence,

$$label_v([\pi_1, \pi_2]) \neq \emptyset,$$

and the lemma follows immediately with $\pi = \Lambda$, $\pi_1' = \pi_1$, and $\pi_2' = \pi_2$. qed

This lemma tells us that, if π_1 and π_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 π, π_1', π_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 $\pi \in O(s) \cup O(t)$ until $label(\pi) = label([\pi])$. When shall the set of labels be increased? By the singularity of $\sim$ we find that

- $label(\pi_1) \leftarrow label(\pi_1) \cup label(\pi_2)$ whenever $label_v(\pi_1) \cap label_v(\pi_2) \neq \emptyset$

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

- $label(\pi_1 \cdot \pi) \leftarrow label(\pi_1 \cdot \pi) \cup label(\pi_2 \cdot \pi)$ whenever $label_v(\pi_1) \cap label_v(\pi_2) \neq \emptyset$,

where $\leftarrow$ denotes assignment. We will call these operations op_S and op_D, respectively. In the sequel we show that these two operations can easily be performed by a connectionist network and that they generate the finest DSE-relation on $O(s) \cup O(t)$.

4.2 The Representation of the Unification Problem

The unification algorithm is based on Feldman's & Ballard's [1982] connectionist model. *Units* will be charcterized by a *potential* p, an *output value* v, and a vector of *inputs* $i_1, \ldots, i_n$. In particular a form of so-called *p-units* will be used, whose potential and output value is determined via the rules

- $p \leftarrow \sum w_k i_k$[1]
- $v \leftarrow$ if $p \geq \theta$ then 1 else 0,

where θ is a constant, called the *threshold* and w_k are weights on the input values. Such a unit will often be called *threshold unit*. Observe that the output value of a threshold unit will be 1 iff the potential is greater or equal to the threshold. The unique output value will be spread along all connections from the unit though these connections will not always be drawn from the same location. For convenience we will occasionally use a *bidirectional* link $\leftrightarrow$ with weight w between two units u_1 and u_2. This is an abbreviation for two links with weight w; one from u_1 to u_2 and another one from u_2 to u_1.

From the previous motivation it is obvious that we intend to represent a unification problem $\langle s = t \rangle$ by the labels of the occurrences $\pi \in O = O(s) \cup O(t)$. For each occurrence π and each label $j \in V \cup F$ there will be a threshold unit $M(\pi, j)$, which is active iff j is a label of π. For our running example $\langle f(x, x, y) = f(g(y), g(g(z)), g(a)) \rangle$ we obtain the units depicted in figure 3 in a tree representation and in figure 4 in a more compact matrix representation. One should observe that this representation is not unique. For example, the unification problem $\langle f(x, g(g(z)), y) = f(g(y), x, g(a)) \rangle$ has the same representation as the running example. However, this does not lead to a problem, since the finest valid equivalence relation for both examples is identical and our goal is to compute this equivalence relation.

As far as this paper is concerned we will not address the problem how the connectionist net

[1]Note that in [Feldman and Ballard, 1982] the potential is updated by $p \leftarrow p + \sum w_k i_k$. As far as our connectionist unification algorithm is concerned there is no need to memorize the potential.

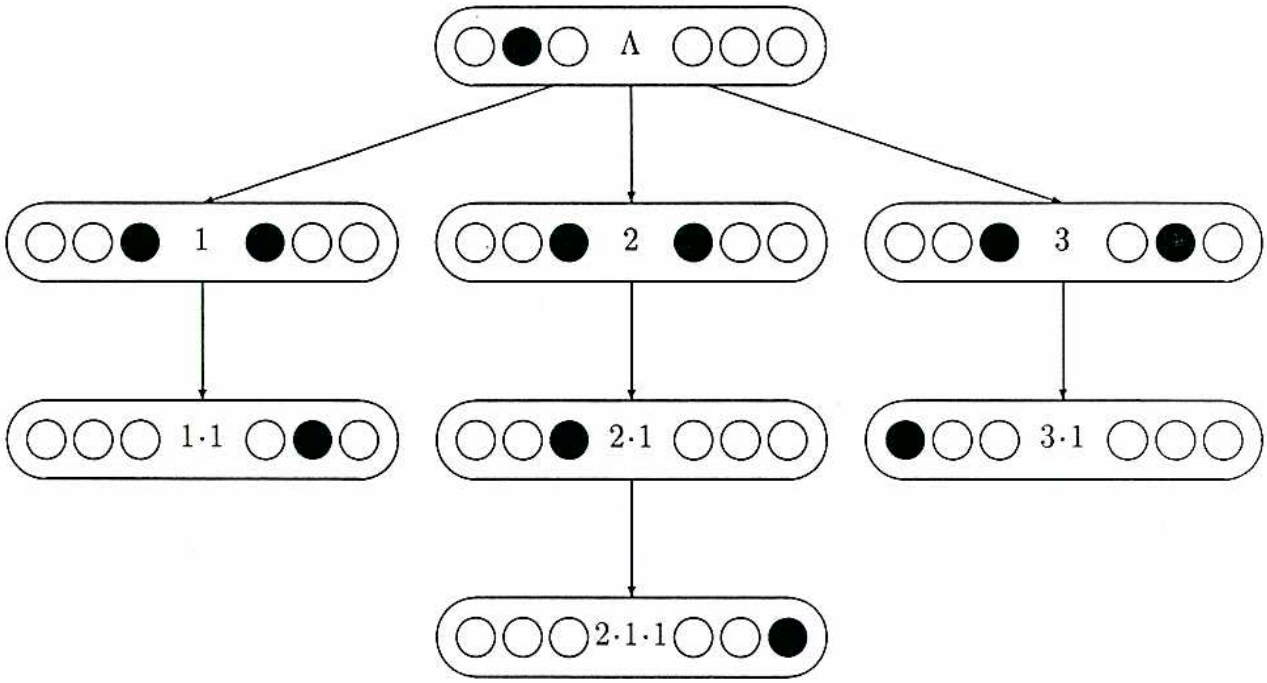

Figure 3: The tree representation of $\langle f(x,x,y) = f(g(y),g(g(z)),g(a))\rangle$.

	a	f	g	x	y	z
Λ	○	●	○	○	○	○
1	○	○	●	●	○	○
1·1	○	○	○	○	●	○
2	○	○	●	●	○	○
2·1	○	○	●	○	○	○
2·1·1	○	○	○	○	○	●
3	○	○	●	○	●	○
3·1	●	○	○	○	○	○

Figure 4: The matrix representation of $\langle f(x,x,y) = f(g(y),g(g(z)),g(a))\rangle$.

is recruited. We just assume that it is present[2]. The user can now specify the unification problem by activating the units in the matrix M, which define the problem. This external activation has to be maintained throughout the computation since our threshold units do not memorize their potential.

If we update the labels according to the operations op_S and op_D we obtain figures 5 and 6 after 1 step and figures 7 and 8 after 2 steps, where the most recent activated units are half-filled. For example, since $M(1\cdot 1, y), M(3, y)$, and $M(3, g)$ are active in figure 4, $M(1\cdot 1, g)$ has become active in figure 6 by op_S. Since $M(1, x), M(2, x)$, and $M(1\cdot 1, y)$ are active in figure 4, $M(2\cdot 1, y)$ has become active in figure 6 by op_D. The final matrix in figure 8 represents precisely the labels of the finest DSE-relation on the set of occurrences of the unification problem.

In section 4.6 we demonstrate how the most general unifier of a solvable unification problem can be read off the final matrix representation. This allows us to view figure 8 as the output of the connectionist unification algorithm. Note that there is no need to generate new terms since each term occurring in the unification problem as well as in the most general unifier is represented in M by a set of occurrence-label pairs modulo the finest DSE-relation.

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, the *term* and the *unification layer*, which we specify in the sequel. Therefore, let

- $\langle s = t\rangle$ be a unification problem,
- $n = |O(s) \cup O(t)|$,
- m be the number of function symbols and variables occurring in $\langle s = t\rangle$, and
- $w = \frac{1}{2} \times n \times m \times (m-1)$ be an integer used to set up thresholds and weights.

4.2.1 The Term Layer

The term layer contains the representation for a unification problem $\langle s = t\rangle$ as an $m \times n$-matrix M of threshold units. Each unit has a threshold of w and is connected via bidirectional links with $n-1$ units in the unification layer. Each link has weight w. The links, thresholds, and weights are determined with respect to the unification layer units and will be explained in the next paragraph. Figure 9 shows a term layer unit.

4.2.2 The Unification Layer

The unification layer contains the units necessary to establish the finest DSE-relation for $\langle s = t\rangle$. There are two basic operations in order to implement op_S and op_D. First, to determine whether two occurrences have the same variable among their labels and, second, to update the set of labels of an occurrence π_1 such that $label(\pi_1) \leftarrow label(\pi_1) \cup label(\pi_2)$ for some π_2. Both operations require that for any two occurrences π_1 and π_2 and for any symbol j, $M(\pi_1, j)$ and $M(\pi_2, j)$ are connected. Since the unification problem has n different

[2]The unification algorithm is usually build into a larger system and we expect that such a system will take care of this problem.

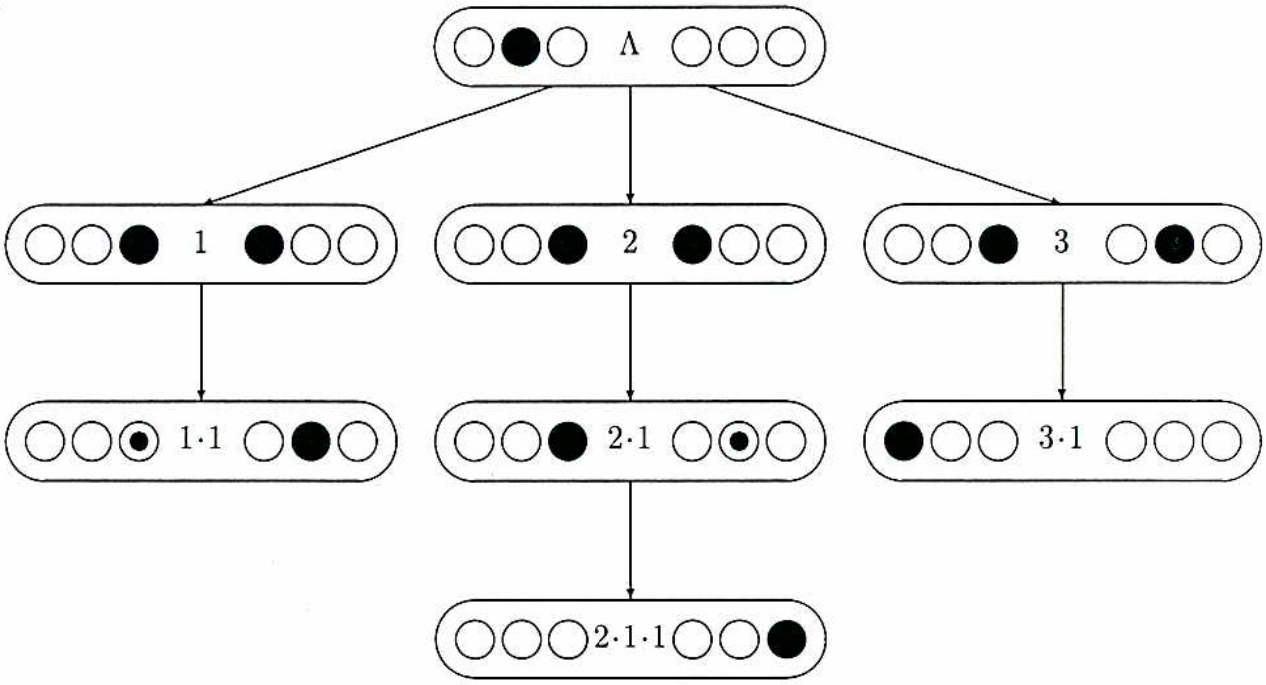

Figure 5: The tree representation of $\langle f(x,x,y) = f(g(y), g(g(z)), g(a))\rangle$ after 1 step.

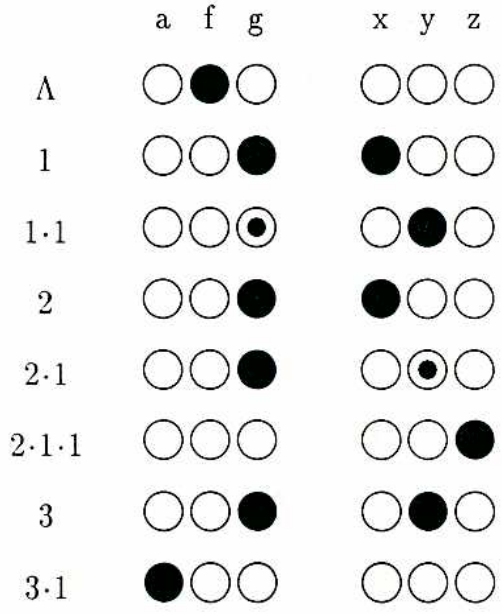

Figure 6: The matrix representation of $\langle f(x,x,y) = f(g(y), g(g(z)), g(a))\rangle$ after 1 step.

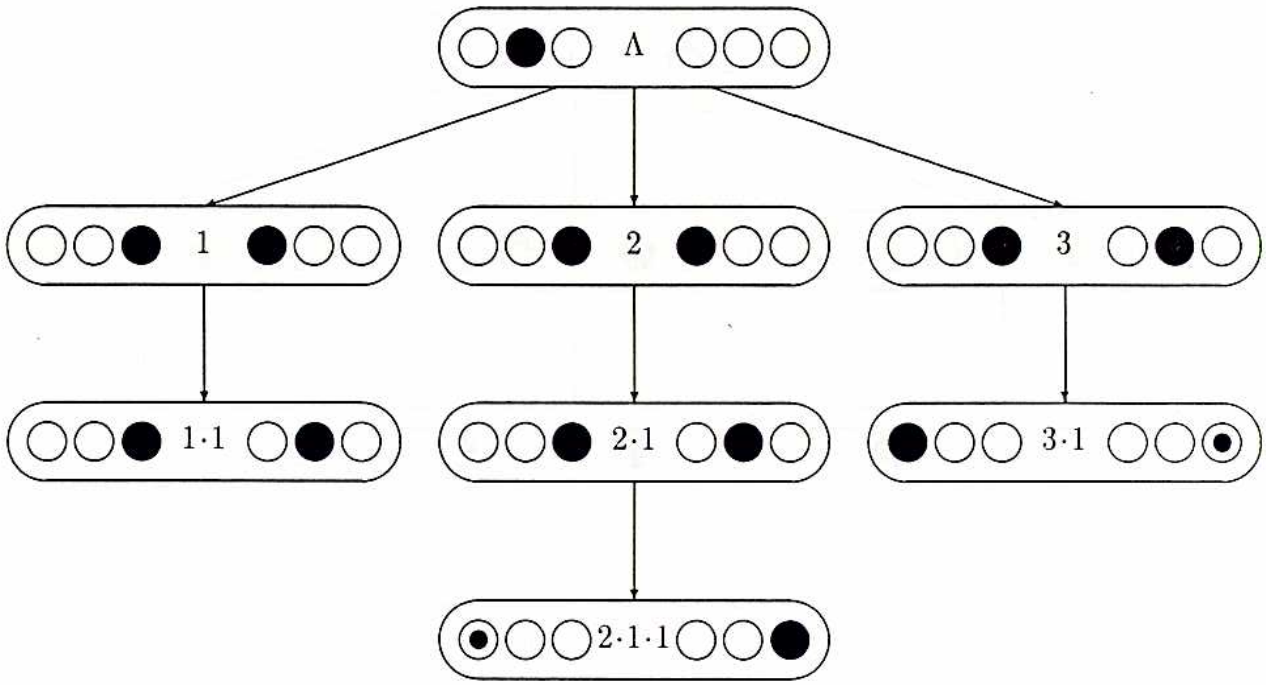

Figure 7: The final tree representation of $\langle f(x,x,y) = f(g(y), g(g(z)), g(a))\rangle$.

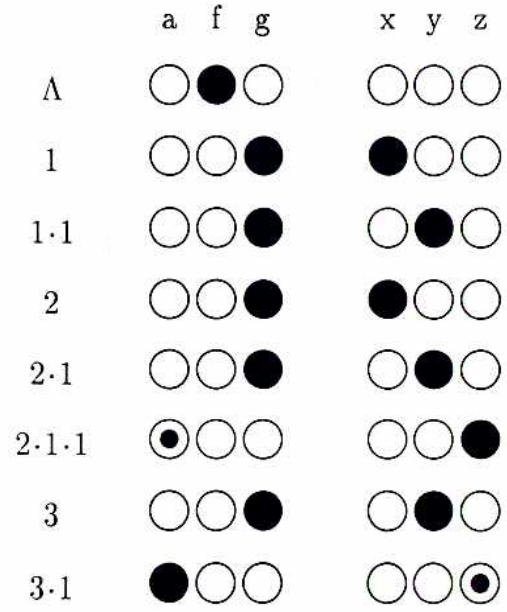

Figure 8: The final matrix representation of $\langle f(x,x,y) = f(g(y), g(g(z)), g(a))\rangle$.

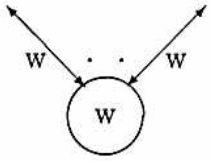

Figure 9: A term layer unit.

occurrences and m different symbols, we need $\frac{1}{2} \times n \times (n-1)$ units for each symbol, i.e. altogether $w = \frac{1}{2} \times m \times n \times (n-1)$ units. These units can easily be represented by a $(\frac{1}{2} \times n \times (n-1)) \times m$ matrix U of threshold units with threshold $w+1$. The elements of this matrix will be $U(\{\pi_1, \pi_2\}, j)$, or $U(\pi_1, \pi_2, j)$ for short[3], indicating that this unit connects $M(\pi_1, j)$ and $M(\pi_2, j)$. Each connection between the unification and the term layer has weight w. Each unification layer unit is also connected to other unification layer units with weight 1 such that there is a connection from

- $U(\pi_1, \pi_2, x)$ to $U(\pi_1, \pi_2, j)$ for all $x \in V$ and $j \in V \cup F$, and
- $U(\pi_1, \pi_2, x)$ to $U(\pi_1 \cdot \pi, \pi_2 \cdot \pi, j)$
  for all π, $x \in V$ and $j \in V \cup F$ 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. Figure 10 depicts a unification layer unit.

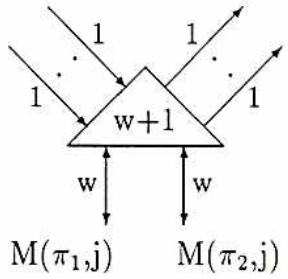

Figure 10: The unification layer unit $U(\pi_1, \pi_2, j)$.

To exemplify the term and unification layers we have depicted the term layer unit $M(2 \cdot 1, y)$ and the unification layer unit $U(1 \cdot 1, 2 \cdot 1, y)$ for the unification problem $\langle f(x, x, y) = f(g(y), g(g(z)), g(a)) \rangle$ in figures 11 and 12, respectively. Figure 13 shows the term layer

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

together with the unification layer units and the connections needed to solve the unification problem. Recall that a unification layer unit will be active iff both corresponding term layer units are active or one corresponding term layer unit is active and the unit receives also activation from another unification layer unit. A term layer unit is active iff it is externally activated or one of its corresponding unification layer units is active. In figure 13 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. More formally,

1. the units labelled 1 are (from left to right) $U(1\cdot 1, 3, y)$ and $U(1, 2, x)$. They will be activated in the first step since the corresponding term layer units $M(1\cdot 1, y)$, $M(3, y)$ as well as $M(1, x)$, $M(3, x)$ are initially active.

2. The units labelled 2 are $U(1{\cdot}1, 3, g)$ and $U(1{\cdot}1, 2{\cdot}1, y)$. The first one is activated because $U(1\cdot 1, 3, y)$ is activated in the first step and $M(3, g)$ is initially active. The second one is activated because $U(1, 2, x)$ is activated in the first step and $M(1\cdot 1, y)$ is initially active.

3. The units labelled 3 are $M(1\cdot 1, g)$ and $M(2\cdot 1, y)$. They are activated in the third step since the corresponding unification layer units $U(1\cdot 1, 2\cdot 1, y)$ and $U(1\cdot 1, 3, y)$ are activated in the second step.

4. The unit labelled 4 is $U(2\cdot 1, 3, y)$. It is activated in the forth step because the corresponding term layer units $M(2\cdot 1, y)$ and $M(3, y)$ are both active after the third step.

5. The units labelled 5 are $U(2{\cdot}1{\cdot}1, 3{\cdot}1, a)$ and $U(2{\cdot}1{\cdot}1, 3{\cdot}1, z)$. They are activated in the fifth step because $U(2{\cdot}1, 3, y)$ as well as $M(3{\cdot}1, a)$ and $M(2{\cdot}1{\cdot}1, z)$ are active after four steps.

6. The units labelled 6 are $M(2{\cdot}1{\cdot}1, a)$ and $M(3{\cdot}1, z)$. They are activated in the sixth step because the corresponding unification layer units $U(2{\cdot}1{\cdot}1, 3{\cdot}1, a)$ and $U(2{\cdot}1{\cdot}1, 3{\cdot}1, z)$ are active after 5 steps.

Note that each activated term or unification layer unit remains active as long as the external activation of the term layer units representing the unification problem is maintained.

4.3 The Computation of the finest DSE-relation

As we have seen in figure 13 the connectionist model solves our running example. This is nice but, of course, we are interested in whether the network computes the finest DSE-relation for any solvable unification problem. More precisely, does

$$\forall \pi \in O(s) \cup O(t) : \forall j \in V \cup F : M(\pi, j) \text{ is active} \Leftrightarrow j \in \mathit{label}([\pi])$$

hold whenever the spreading of activation within the net has come to an end?

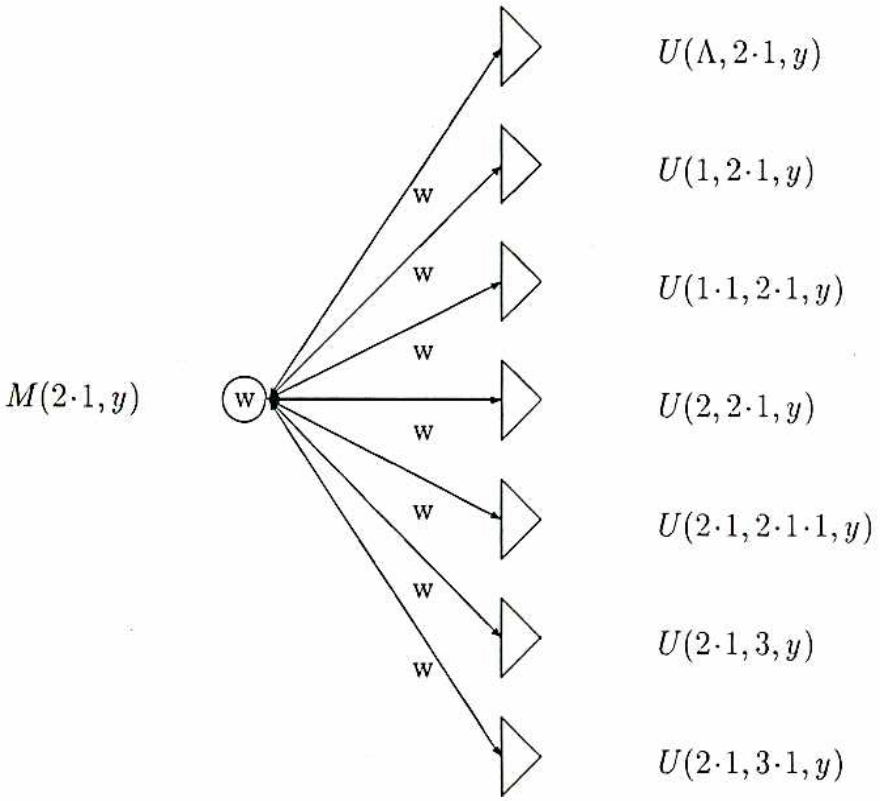

Figure 11: The term layer unit $M(2\cdot 1, y)$.

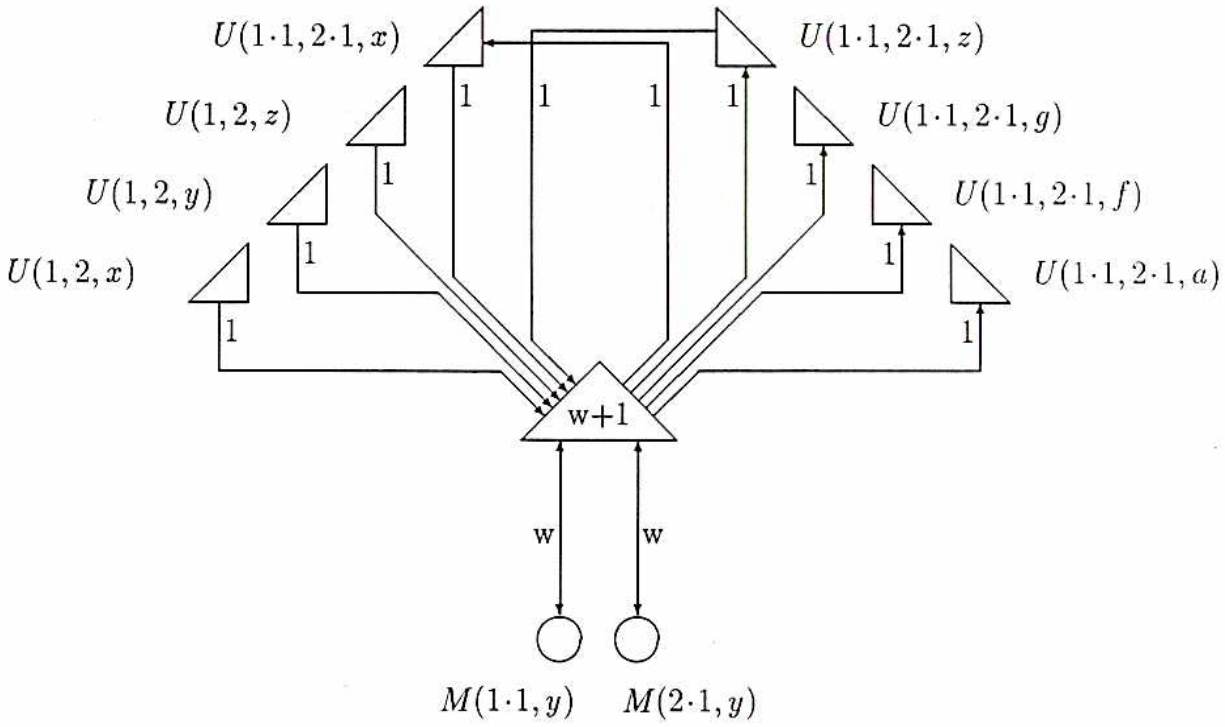

Figure 12: The unification layer unit $U(1{\cdot}1, 2{\cdot}1, y)$.

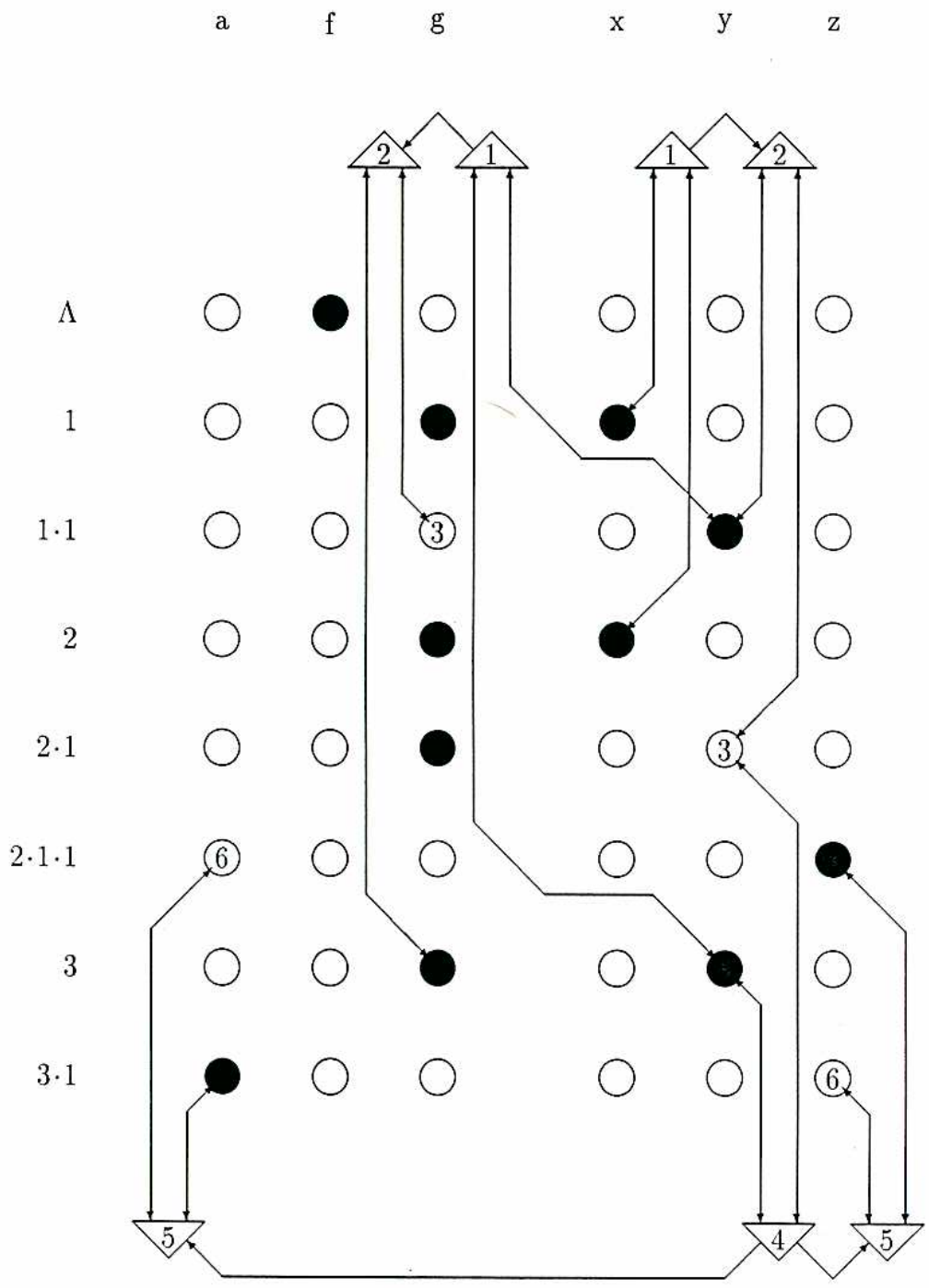

Figure 13: The term layer together with the unification layer units and the connections needed for solving the unification problem $\langle f(x,x,y) = f(g(y),g(g(z)),g(a))\rangle$. The number i in a unit indicates that this unit will be active after i steps.

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

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

Coming back to the running example let

$$\begin{array}{rcl} N & = & \{M(\Lambda, f), M(1, g), M(1, x), M(1 \cdot 1, y), M(2, g), M(2, x), \\ & & M(2 \cdot 1, g), M(2 \cdot 1 \cdot 1, z), M(3, g), M(3, y), M(3 \cdot 1, a)\} \end{array}$$

be the set of active term layer units representing the unification problem. Now we find

$$\begin{array}{rcl} T(N) & = & N \cup \{U(1, 2, x), U(1 \cdot 1, 3, y)\} \\ T^2(N) & = & T(N) \cup \{U(1 \cdot 1, 2 \cdot 1, y), U(1 \cdot 1, 3, g), U(1, 2, g), U(1, 2 \cdot 1, g), U(1, 3, g), \\ & & U(2, 2 \cdot 1), U(2, 3, g), U(2 \cdot 1, 3, g)\} \\ T^3(N) & = & T^2(N) \cup \{M(2 \cdot 1, y), M(1 \cdot 1, g)\} \\ T^4(N) & = & T^3(N) \cup \{U(2 \cdot 1, 3, y), U(1, 1 \cdot 1, g), U(1 \cdot 1, 2, g), U(1 \cdot 1, 2 \cdot 1, g)\} \\ T^5(N) & = & T^4(N) \cup \{U(2 \cdot 1 \cdot 1, 3 \cdot 1, z), U(2 \cdot 1 \cdot 1, 3 \cdot 1, a)\} \\ T^6(N) & = & T^5(N) \cup \{M(3 \cdot 1, z), M(2 \cdot 1 \cdot 1, a)\} \\ T^i(N) & = & T^6(N) \text{ for all } i > 6. \end{array}$$

Note that in figure 13 we have depicted only those unification layer units, whose activation triggers the activation of term layer units.

The following proposition is an immediate consequence of the definition of T and the fact that there are $\frac{1}{2} \times m \times n \times (n+1)$ units in the term and unification layer.

Proposition 4

1. $T(N) \supseteq N$
2. $T(N) = N \Rightarrow \forall k > 0 : T^k(N) = N$
3. $[\exists x \in V : U(\pi_1, \pi_2, x) \in N]$
   $\Rightarrow [\exists k \geq 0 : \forall j \in V \cup F : \forall \pi : M(\pi_1 \cdot \pi, j) \in T^k(N) \Leftrightarrow M(\pi_2 \cdot \pi, j) \in T^k(N)]$
4. $\exists k \leq \frac{1}{2} \times n \times m \times (m+1) : T^{k+1}(N) = T^k(N)$

We can now reformulate the question raised at the beginning of this section. Let N be the set of active term layer units which represent a unification problem and let $N^* = T^i(N)$ such that $T(N^*) = N^*$.

Theorem 5 $\forall \pi \in O(s) \cup O(t) : \forall j \in V \cup F : M(\pi, j) \in N^* \Leftrightarrow j \in \text{label}([\pi])$

Proof: To prove the only-if-half we show by transfinite induction on k that

$$M(\pi_1, j) \in T^k(N) \Rightarrow j \in \mathit{label}([\pi]).$$

Suppose the result holds for all $l < k$ and assume that

$$M(\pi_1, j) \in T^k(N).$$

By the definition of T either

$$M(\pi_1, j) \in N$$

or we find a π_2 such

$$U(\pi_1, \pi_2, j) \in T^{k-1}(N).$$

The first case being trivial ($j \in \mathit{label}(\pi_1)$) we turn to the second case. By the definition of T either

$$\{M(\pi_1, j), M(\pi_2, j)\} \subseteq T^{k-2}(N) \tag{1}$$

or we find $x \in V, \pi, \pi_1', \pi_2'$ such that $\pi_1 = \pi_1' \cdot \pi, \pi_2 = \pi_2' \cdot \pi$ and

$$(M(\pi_1, j) \in T^{k-2}(N) \vee M(\pi_2, j) \in T^{k-2}(N)) \wedge U(\pi_1', \pi_2', x) \in T^{k-2}(N). \tag{2}$$

We distinguish two cases. If $M(\pi_1, j) \in T^{k-2}(N)$ then the result follows immediately from the induction hypothesis. Otherwise, equations (1) and (2) can be reduced to

$$M(\pi_2, j) \in T^{k-2}(N) \wedge U(\pi_1', \pi_2', x) \in T^{k-2}(N).$$

By the definition of T we find

$$M(\pi_2, j) \in T^{k-1}(N) \wedge \{M(\pi_1', x), M(\pi_2', x)\} \subseteq T^{k-1}(N)$$

and by an application of the induction hypothesis we learn

$$j \in \mathit{label}([\pi_2]) \wedge x \in \mathit{label}([\pi_1', \pi_2']).$$

With the axiom of singularity follows

$$j \in \mathit{label}([\pi_2]) \wedge \pi_1' \sim \pi_2'$$

and an application of lemma 3 yields

$$j \in \mathit{label}([\pi_2]) \wedge \pi_1 \sim \pi_2.$$

Hence, $\mathit{label}([\pi_1]) = \mathit{label}([\pi_2])$ which proves the only-if-half.

Conversely, we have to show that

$$j \in \mathit{label}([\pi_1]) \Rightarrow \exists k : M(\pi_1, j) \in T^k(N).$$

The case $j \in label(\pi_1)$ being trivial ($M(\pi_1, j) \in N$) we assume that $j \notin label(\pi_1)$. But then we find a π_2 such that $\pi_1 \sim \pi_2$, $\pi_1 \neq \pi_2$, and $j \in label(\pi_2)$. We claim that

$$A \vdash^k \pi_1 \sim \pi_2 \Rightarrow [\forall j \in V \cup F : M(\pi_1, j) \in N^* \Leftrightarrow M(\pi_2, j) \in N^*]$$

holds, which implies the result. The claim is now proved by transfinite induction on k. Suppose the claim holds for all $l < k$ and assume that

$$A \vdash^k \pi_1 \sim \pi_2.$$

We distinguish four cases with respect to the axiom applied in the last step of the derivation. Note that the axiom of reflexivity was not applied, since this implies $\pi_1 = \pi_2$.

1. If the axiom of singularity was applied then

   $$label_v(\pi_1) \cap label_v(\pi_2) \neq \emptyset$$

   and we find an $x \in V$ such that

   $$\{M(\pi_1, x), M(\pi_2, x)\} \subseteq N.$$

   From the definition of T we learn that in this case

   $$U(\pi_1, \pi_2, x) \in T(N).$$

   By proposition 4 (3) we find an p such that for all $j \in V \cup F$

   $$M(\pi_1, j) \in T^p(N) \Leftrightarrow M(\pi_2, j) \in T^p(N)$$

   holds, which implies

   $$M(\pi_1, j) \in N^*.$$

2. If the axiom of symmetry was applied then

   $$A \vdash^{k-1} \pi_2 \sim \pi_1$$

   and by the induction hypothesis we find that for all $j \in V \cup F$

   $$M(\pi_2, j) \in N^* \Leftrightarrow M(\pi_1, j) \in N^*$$

   holds. The result follows immediately.

3. If the axiom of transitivity was applied then

   $$A \vdash^{k-1} \pi_1 \sim \pi_3 \wedge A \vdash^{k-1} \pi_3 \sim \pi_2$$

   and by the induction hypothesis we find that for all $j \in V \cup F$

   $$M(\pi_1, j) \in N^* \Leftrightarrow M(\pi_3, j) \in N^*$$

   and

   $$M(\pi_3, j) \in N^* \Leftrightarrow M(\pi_2, j) \in N^*$$

   holds. The result follows immediately.

4. Finally, if the axiom of decomposability was applied then we find i, π_1', π_2' such that $\pi_1 = \pi_1' \cdot i, \pi_2 = \pi_2' \cdot i$ and

$$A \vdash^{k-1} \pi_1' \sim \pi_2'.$$

By lemma 3 we find $\pi, \overline{\pi_1}, \overline{\pi_2}$ such that $\pi_1' = \overline{\pi_1} \cdot \pi, \pi_2' = \overline{\pi_2} \cdot \pi$ and

$$A \vdash^{k-1-|\pi|} \overline{\pi_2} \sim \overline{\pi_1} \wedge label_v([\overline{\pi_1}, \overline{\pi_2}]) \neq \emptyset.$$

By the induction hypothesis we find $x \in V$ such that

$$\{M(\overline{\pi_1}, x), M(\overline{\pi_2}, x)\} \subseteq N^*.$$

By the definition of T we find a q such that

$$U(\overline{\pi_1}, \overline{\pi_2}, x) \in T^q(N).$$

Since $\pi_1 = \pi_1' \cdot i = \overline{\pi_1} \cdot \pi \cdot i$ and $\pi_2 = \pi_2' \cdot i = \overline{\pi_2} \cdot \pi \cdot i$ proposition 4 (3) ensures that there exists a p such that for all $j \in V \cup F$

$$M(\pi_1, j) \in T^{q+p}(N) \Leftrightarrow M(\pi_2, j) \in T^{q+p}(N)$$

holds, which implies

$$M(\pi_1, j) \in N^*.$$

qed

This theorem ensures that the finest DSE-relation for a unification problem $\langle s = t \rangle$ is generated by the connectionist network. 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 of the algorithm is bound by the square of the size of the unification problem. The time complexity of this algorithm seems also to be bound by the square of the size the unification problem by proposition 4(4). However, the following theorem shows that the algorithm needs only parallel time linear to the size n of the unification problem.

Theorem 6 $T^{3\times n+1}(N) = T^{3\times n}(N)$.

Proof: The proof is based on the observation that each spreading of activation in the term and unification layer has its origin in active term layer units which represent a variable. Observe that whenever two rows in the term layer are equal by virtue of the singularity axiom, then it takes 3 steps in the worst case to exchange the activation in these rows and, similarly, to exchange the activation of rows which have to be equal by the decomposability axioms.

Formally, let $\overset{*}{\approx}_i$ be the equivalence closure of $\approx_i$, where $\pi_1 \approx_i \pi_2$ iff $\exists x : \{M(\pi_1, x), M(\pi_2, x)\} \subseteq T^i(N)$. Furthermore, let $O|_{\overset{*}{\approx}_i}$ be the quotient of the set O of occurrences of the unification

problem and $\overset{*}{\approx}_i$. Since n is the cardinality of O and activated term layer units remain active throughout the unification process we conclude that for all i

$$card(O|_{\overset{*}{\approx}_i}) \leq n \tag{3}$$

and

$$card(O|_{\overset{*}{\approx}_{i+1}}) \leq card(O|_{\overset{*}{\approx}_i}) \tag{4}$$

hold. Now, for each N and sequence $T^j(N)$ "delay" the activation of term layer units representing variables to obtain a new sequence $\overline{T}^i(N)$ such that

$$card(O|_{\overset{*}{\approx}_0}) = n \tag{5}$$

$$card(O|_{\overset{*}{\approx}_{i+3}}) \geq card(O|_{\overset{*}{\approx}_i}) - 1 \tag{6}$$

$$card(O|_{\overset{*}{\approx}_{i+3}}) = card(O|_{\overset{*}{\approx}_i}) \Rightarrow \forall k : card(O|_{\overset{*}{\approx}_{i+k}}) = card(O|_{\overset{*}{\approx}_i}) \tag{7}$$

Obviously, $\overline{T}^i(N) \subseteq T^i(N)$ for all i and for each sequence $\overline{T}^i(N)$ we find that

$$card(O|_{\overset{*}{\approx}_{i+3}}) = card(O|_{\overset{*}{\approx}_i}) \Rightarrow \overline{T}^{i+4}(N) = \overline{T}^{i+3}(N). \tag{8}$$

By (4) the cardinality of $O|_{\overset{*}{\approx}_0}$ can be decreased only n times and by (6) and (7) a decrease of this cardinality by 1 requires precisely 3 steps. Hence, with (8) we conclude that $T^{3\times n+1}(N) = T^{3\times n}(N)$. qed

It is worth to look at an example for the construction of $\overline{T}^i(N)$ from $T^j(N)$ used in the previous proof. Consider the unification problem $\langle f(x,x) = f(y,z)\rangle$. Then,

$$\begin{array}{lcl} N & = & T^0(N) = \{M(\Lambda, f), M(1,x), M(2,x), M(1,y), M(2,z)\} \\ T^1(N) & = & T^0(N) \cup \{U(1,2,x)\} \\ T^2(N) & = & T^1(N) \cup \{U(1,2,x), U(1,2,z)\} \\ T^3(N) & = & T^2(N) \cup \{M(2,y), M(1,z)\} \\ T^{3+k}(N) & = & T^3(N) \text{ for all } k. \end{array}$$

To construct $\overline{T}$ we "delay" the activation of $M(2,x)$ until the third step and obtain

$$\begin{array}{lcl} \overline{T}^0(N) & = & \{M(\Lambda, f), M(1,x), M(1,y), M(2,z)\} \\ \overline{T}^1(N) & = & \overline{T}^0(N) \\ \overline{T}^2(N) & = & \overline{T}^1(N) \\ \overline{T}^3(N) & = & \overline{T}^2(N) \cup \{M(2,x)\} \\ \overline{T}^4(N) & = & \overline{T}^3(N) \cup \{U(1,2,x)\} \\ \overline{T}^5(N) & = & \overline{T}^4(N) \cup \{U(1,2,x), U(1,2,z)\} \\ \overline{T}^6(N) & = & \overline{T}^7(N) \cup \{M(2,y), M(1,z)\} \\ \overline{T}^{6+k}(N) & = & \overline{T}^6(N) \text{ for all } k. \end{array}$$

The previous theorem describes a worst case scenario. We should not be surprised by this result. As Dwork, etal. [1984] have shown, unification is logspace complete and, thus, a

better worst case performance can only be achieved if $P \subseteq NC$, which is very unlikely. In fact, our running example is used by Dwork etal. to examplify their result. Recall, that

$$\langle f(x,x,y) = f(g(y), g(g(z)), g(a)) \rangle$$

can be solved in 6 steps with $n = 8$. The interested reader is encouraged to verify that the unification problem

$$\langle f(x,x,y_1,y_2) = f(g(y_1), g(g(y_2)), g(g(z)), g(a)) \rangle$$

can be solved in 9 steps with $n = 11$. However, even in this worst case scenario the number of steps of the connectionist unification algorithm is smaller than the size of the unification problem. In fact, we are unaware of an example where the least fixpoint of T was not computed after $n + 1$ steps. In most examples this fixpoint was computed in a number of steps much smaller than n (see section 4.7).

Let's have a look at a best case scenario. This is – for example – the case when the unification problem degenerates to the word problem, i.e. to the question of whether two terms are syntactically equal. If two terms s and t are syntactically equal then each occurrence π in $O(s) \cup O(t)$ has precisely one label. Furthermore, whenever the label of π is a variable then there is no i such that $\pi \cdot i \in O(s) \cup O(t)$. Consequently, whenever a unit of the form $U(\pi_1, \pi_2, j)$ is activated in the first step, then j occurs at π_1 and π_2 in s as well as in t and no further activation will take place. Thus, the finest DSE-relation is found after 1 step, i.e. $N^* = T(N)$, where N is the set of active term layer units representing the word problem. For more examples see section 4.7.

There is another interesting restriction of the unification problem, namely the matching problem. Recall, that a matching problem $\langle s \geq t \rangle$ has a solution iff we find a substitution σ such that $\sigma s = t$. If we replace each variable in t with a new constant[4], then the matching problem is equal to the unification problem. Hence, in the sequel we may assume without loss of generality that t is a ground term. Now the interesting question is of how long it will take to compute the finest DSE-relation for a matching problem.

Theorem 7 *If N is the representation of a solvable matching problem then $T^2(N) = T(N)$.*

Proof: Consider a solvable matching problem $\langle s \geq t \rangle$. Since t is a ground term we find for all $\pi_1 \in O(s) \cup O(t)$ that either

$$|label(\pi_1)| = 1 = |\{f\}| \text{ and } f \in F \tag{9}$$

or

$$|label(\pi_1)| = 2 = |\{x, f\}| \text{ and } x \in V \text{ and } f \in F. \tag{10}$$

Now let N be the set of active term layer units representing the matching problem. By the definition of T we find that

$$U(\pi_1, \pi_2, j) \in T(N)$$

[4]i.e. a constant not occurring in s and t

whenever $\{M(\pi_1, j), M(\pi_2, j)\} \subseteq N$. Hence, in (9) as well as in (10) a unit $U(\pi_1, \pi_2, f)$ is in $T(N)$ iff $f \in \mathit{label}(\pi_2)$. However, since $f \in F$ such a unit $U(\pi_1, \pi_2, f)$ does not trigger the activation of any other unit.

Whenever (10) holds we find ground terms $t_1, \ldots, t_p$, $n \geq 0$, such that

$$s|\pi_1| = x \text{ and } t|\pi_1| = f(t_1, \ldots, t_p).$$

A unit of the form $U(\pi_1, \pi_2, x)$ is in $T(N)$ iff $M(\pi_2, x) \in N$ iff $x \in \mathit{label}(\pi_2)$. Now assume that we find a π_2 such that $M(\pi_2, x) \in N$ and let $t|\pi_2| = g(s_1, \ldots, s_q)$. For the matching problem to be solvable we conclude that $g = f$, $p = q$, and $s_i = t_i$ for all $1 \leq i \leq p$. In other words, for all π such that $\{\pi_1 \cdot \pi, \pi_2 \cdot \pi\} \subseteq O(s) \cup O(t)$ we find $t|\pi_1 \cdot \pi| = t|\pi_2 \cdot \pi|$ and, by the definition of T, we find a function symbol j such that $U(\pi_1 \cdot \pi, \pi_2 \cdot \pi, j) \in T(N)$. However, the only units that can be activated by $U(\pi_1, \pi_2, x)$ are units of the form $U(\pi_1 \cdot \pi, \pi_2 \cdot \pi, j)$.

In any case we find $T^2(N) = T(N)$. qed

This theorem improves a result by Dwork et al. [1984], who have shown that the matching problem can be solved on a PRAM in $log^2 n$ parallel time, where n is the size of the matching problem. As an example consider

$$\langle f(x, g(x)) \geq f(h(a), g(h(a))) \rangle.$$

Then

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

and

$$T(N) = N \cup \{U(1, 2 \cdot 1, h), U(1 \cdot 1, 2 \cdot 1 \cdot 1, a), U(1, 2 \cdot 1, x)\} = T^2(N).$$

The most general matcher for this example is $\{x \leftarrow h(a)\}$ and – as shown in section 4.6 – can easily be extracted from $T(N)$.

It should be noted that word or matching problems are solved without changing the design of the algorithm. There is also no need of informing the algorithm that a restricted unification problem is to be solved.

4.4 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 in previous section and we find that $2 \cdot 1 \cdot 1 \sim 3 \cdot 1$, but $\mathit{label}_f(2 \cdot 1 \cdot 1) \cup \mathit{label}_f(3 \cdot 1) = \{a, b\}$. Fortunately, the fact that a DSE-relation is not homogeneous can easily be detected.

Since a DSE-relation $\sim$ is characterized by $\mathit{label}([\pi])$ we find that $\sim$ is homogeneous iff

$$\forall \pi \in O(s) \cup O(t) : |\mathit{label}_f([\pi])| \leq 1$$

In other words, $\sim$ is not homogenous iff there is a row π 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 π of the term layer we insert an additional threshold unit $H(\pi)$ which receives activation from each unit $M(\pi, f)$, where f is a function symbol. If each connection from a term layer unit to $H(\pi)$ has weight 1 and the threshold of $H(\pi)$ is 2, then we find a π such that $H(\pi)$ is active iff $\sim$ is not homogeneous. Figure 14 shows the required additional network for the unification problem $\langle f(x,x,y) = f(g(y), g(g(b)), g(a))\rangle$. The unit H is a threshold unit with threshold 1 and will become active if at least one of the $H(\pi)$-units is active. Thus, H will become active iff the finest DSE-relation is not homogeneous. In the example, the units $M(2{\cdot}1{\cdot}1, a)$ and $M(3{\cdot}1, b)$ will become active after 6 steps. Hence, $H(2{\cdot}1{\cdot}1)$ and $H(3{\cdot}1)$ will be active after 7 steps and, finally, H will be active after 8 steps indicating the non-unifiability of $f(x,x,y)$ and $f(g(y), g(g(b)), g(a))$. Thus, to check whether the finest DSE-relation is homogemeous requires $n+1$ additional units and takes 2 steps.

Proposition 1(1) ensures that the algorithm developed so far decides a unification problem $\langle s = t\rangle$ over infinite trees [Colmerauer, 1982; Colmerauer, 1984]. One should observe, that the word as well as the matching problem is the same regardless whether it is interpreted over the domain of finite or infinite trees. Thus, by the contraposition theorem 7 we find that whenever $T^2(N) \supset T(N)$ the matching problem is unsolvable and, hence, a word or a matching problem can be decided in 2 steps. As an example consider the matching problem

$$\langle f(x, g(x)) \geq f(a, g(b))\rangle.$$

Here we find

$$N = \{M(\Lambda, x), M(1, x), M(2, g), M(2{\cdot}1, x), M(1, a), M(2{\cdot}1, b)\}$$

and

$$\begin{aligned} T(N) &= N \cup \{U(1, 2{\cdot}1, x)\} \\ T^2(N) &= T(N) \cup \{U(1, 2{\cdot}1, a), U(1, 2{\cdot}1, b)\} \\ T^3(N) &= T^2(N) \cup \{M(2{\cdot}1, a), M(1, b)\} \\ T^4(N) &= T^3(N). \end{aligned}$$

After 2 additional steps it can be recognized that the DSE-relation is not homogeneous since $\{M(1,a), M(1,b)\} \subset T^3(N)$ and $\{M(2\cdot 1, a), M(2\cdot 1, b)\} \subset T^3(N)$. But recall, that the matching problem is unsolvable follows already from the fact that $T^2(N) \supset T(N)$.

4.5 Valid Equivalence Relations

It remains to be checked whether the finest homogeneous DSE-relation is acyclic or, in other words, valid. Recall that an equivalence relation $\sim$ on the occurrences of a unification problem is acyclic, iff the $\sim$-equivalence classes are partially ordered by $\succ$, where $C_1 \succ C_2$ iff we find $\pi_1 \in C_1$ and $\pi_2 \in C_2$ such that $\pi_1 > \pi_2$. Unfortunately, there is no unit in our connectionist model which represents a $\sim$-equivalence class. However, we know that two different occurrences π_1 and π_1' are equal under $\sim$ iff there are a variable x and occurrences π_2, π_2' and π such that $U(\pi_2, \pi_2', x)$ is active, $\pi_1 = \pi_2{\cdot}\pi$, and $\pi_1' = \pi_2'{\cdot}\pi$. This means that we find a unit in our model which tells us whether two different occurrences are equal or not. This fact will be exploited in order to determine whether $\sim$ is acyclic. Formally, let

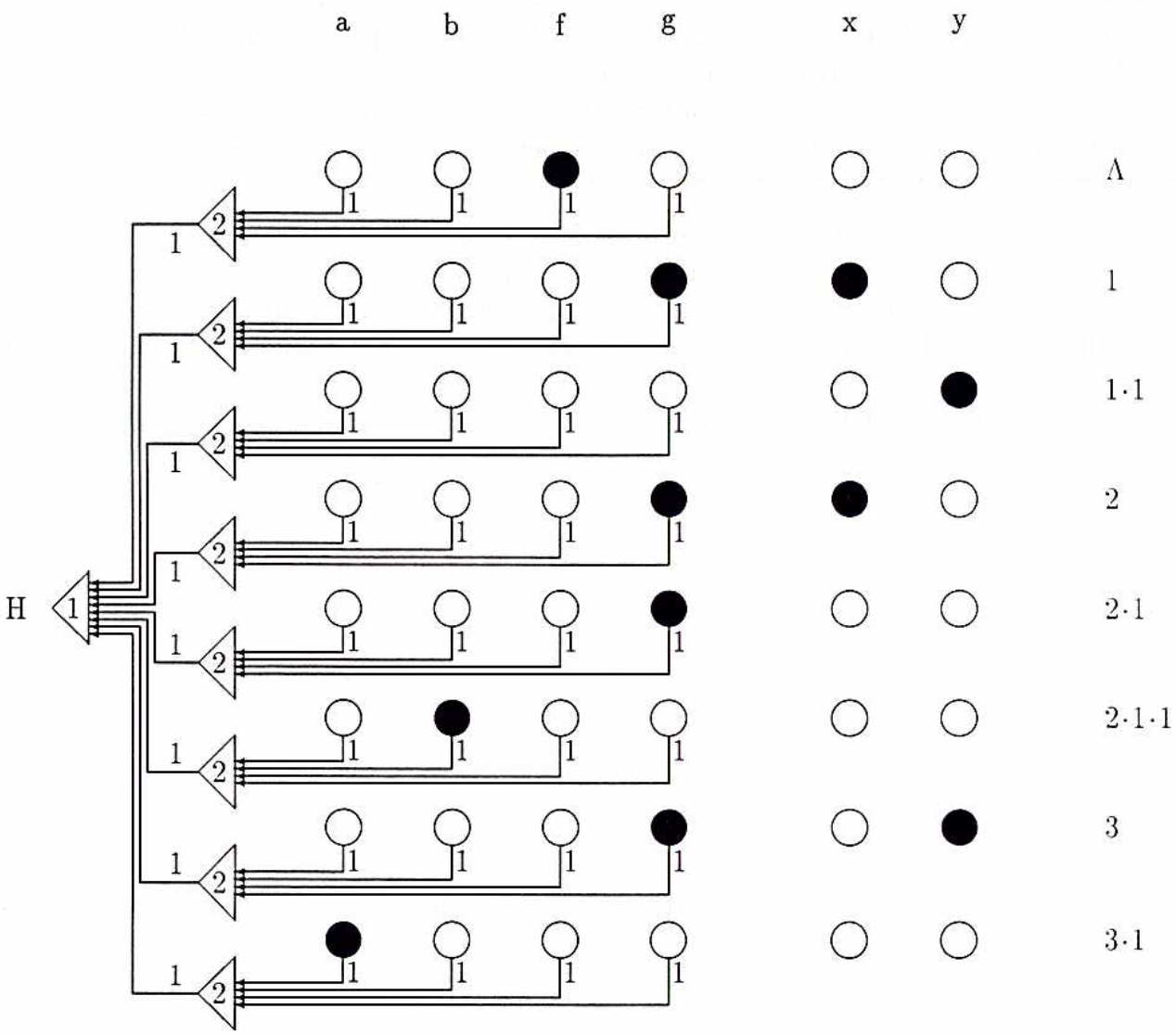

Figure 14: The term layer for the unification problem $\langle f(x,x,y) = f(g(y),g(g(b)),g(a))\rangle$ together with the units and connections needed to test whether the finest DSE-relation is homogeneous. All connections are labeled with weight 1. The number in the units indicates the threshold.

$C(\{\pi, \pi'\})$ be an $\frac{1}{2} \times n \times (n-1)$ vector, where π and π' are different elements of $O(s) \cup O(t)$. For notational convenience the elements of C will be denoted by $C(\pi, \pi')$[5]. The elements of C are ordered by

- $C(\pi_1, \pi_1') > C(\pi_2, \pi_2')$ iff $\exists \pi \in \{\pi_1, \pi_1'\}, \overline{\pi} \in \{\pi_2, \pi_2'\} : \pi > \overline{\pi}$.

The idea is that each element $C(\pi, \pi')$ of the vector C is represented by a unit which is activated iff $\pi \sim \pi'$. Furthermore, a unit $C(\pi_1, \pi_1')$ will excite a unit $C(\pi_2, \pi_2')$ iff $C(\pi_1, \pi_1') > C(\pi_2, \pi_2')$ such that the units representing a cycle will form a stable coalition, whereas the units which are not part of a cycle will be deactivated after some time.

Let $\sim$ be the finest homogeneous DSE-relation for the unification problem $\langle s = t \rangle$. The following proposition ensures that a cycle can be detected using the vector C.

Proposition 8
$\sim$ is cyclic iff $\exists C(\pi_1, \pi_1') \ldots C(\pi_k, \pi_k') : C(\pi_1, \pi_1') > \ldots > C(\pi_k, \pi_k') > C(\pi_1, \pi_1')$.

Proof: $\sim$ is cyclic iff we find $\sim$-equivalence classes $C_1, \ldots C_l$ such that

$$C_1 \succ \ldots \succ C_l \succ C_1$$

iff we find $\pi_i, \pi_i' \in C_i, 1 \leq i \leq l$, such that

$$\pi_1' > \pi_2 \sim \pi_2' > \ldots > \pi_l \sim \pi_l' > \pi_1. \tag{11}$$

Since $>$ is a partial ordering on the set of occurrences we conclude that there must be a $j \in \{1, \ldots, l\}$ such that $\pi_j \neq \pi_j'$. Now let

$$\pi_{j_1}' > \pi_{j_2} \sim \pi_{j_2}' > \ldots > \pi_{j_k} \sim \pi_{j_k}' > \pi_{j_1}$$

be obtained from (11) by deleting all pairs π_j, π_j' whenever $\pi_j = \pi_j'$. The proposition follows immediately with

$$C(\pi_{j_1}, \pi_{j_1}') > \ldots > C(\pi_{j_k}, \pi_{j_k}') > C(\pi_{j_1}, \pi_{j_1}').$$

qed

How can cycles be detected by a connectionist network? Each element $C(\pi, \pi')$ will be represented by a p-unit with threshold 1 in the so-called *occur-check layer*. There is a connection 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 iff $\pi \sim \pi'$, whereas $C(\pi, \pi')$ is externally inhibited if $\pi \not\sim \pi'$. The external inhibition must be strong enough to prevent an activation of such a unit by other occur check layer units. To get an impression of the behaviour of the occur check layer consider the unification problem $\langle f(x, y) = f(g(y), g(x)) \rangle$. For this problem a homogeneous DSE-relation exists which is represented in figure 15. Obviously, $1 \sim 2{\cdot}1$ and $2 \sim 1{\cdot}1$ and, consequently, the units $C(1, 2{\cdot}1)$ and $C(2, 1{\cdot}1)$ will initially be

[5]Note, $C(\pi, \pi')$ and $C(\pi', \pi)$ denote the same element.

active. Figure 16 shows the occur check layer for this example, where all connections from a self-excitatory unit are not shown. Obviously, a self-excitatory unit will remain active whenever activated. In the example, the units $C(1, 2{\cdot}1)$ and $C(2, 1{\cdot}1)$ are mutually excitatory and, thus, form a stable coalition. Hence, they signal that the DSE-relation is cyclic. Note that unit $C(2, 1{\cdot}1)$ does not activate $C(2, 2{\cdot}1)$, since $C(2, 2{\cdot}1)$ is externally inhibited.

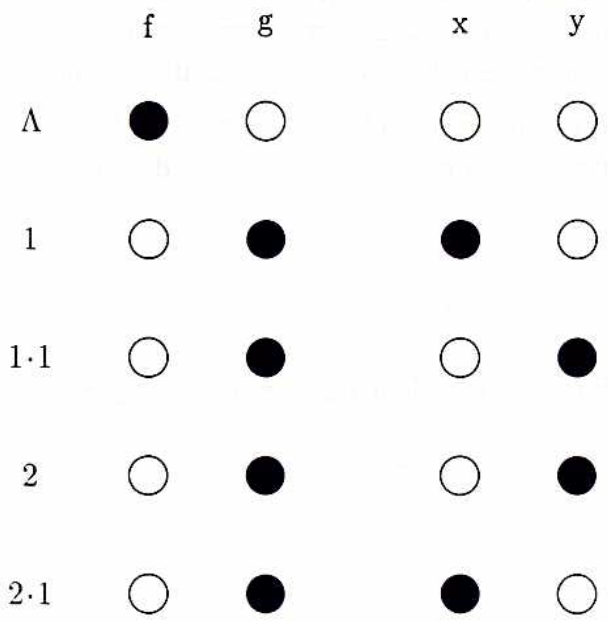

Figure 15: The term layer for the unification problem $\langle f(x, y) = f(g(y), g(x))\rangle$ after computing the finest DSE-relation.

A second example is concerned with the problem $\langle f(x, x, y) = f(g(y), g(g(z)), g(a))\rangle$. Figure 17 shows the initial activation of the occur check layer, where we have omitted all links that are not affected by the example. Since unit $C(1, 2)$ is not excited by any other unit it will be deactivated after 1 step. Consequently, $C(1{\cdot}1, 2{\cdot}1), C(1{\cdot}1, 3)$, and $C(2{\cdot}1, 3)$ will be deactivated after 2 steps and, finally, $C(2{\cdot}1{\cdot}1, 3{\cdot}1)$ will go off. This signals that the DSE-relation is acyclic.

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 function 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].

The following proposition is an immediate consequence of the definition of S.

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

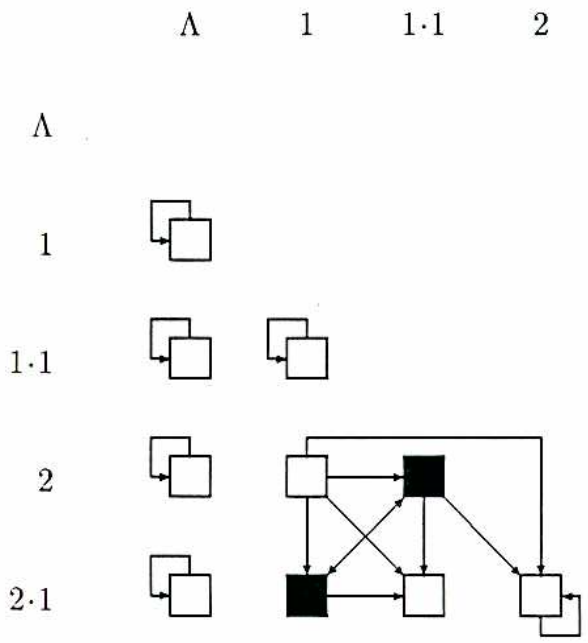

Figure 16: The occur check layer for the unification problem $\langle f(x,y) = f(g(y),g(x))\rangle$. Each unit has a threshold of 1 and each connection has weight 1. A filled square indicates that the unit is activated.

Proposition 9

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

In the sequel let $C = \{C(\pi,\pi') \mid \pi \sim \pi'\}$ and $C^* = S^k(C)$ such that $S^{k+1}(C) = S^k(C)$. In other words, C contains all occur check layer units which are initially active.

Theorem 10 *$C^* = \emptyset$ iff $\sim$ is acyclic.*

Proof: $\sim$ is cyclic

iff we find $\sim$-equivalence classes $C_1,\ldots,C_n$ such that

$$C_1 \prec C_2 \prec \ldots \prec C_n \prec C_1$$

iff we find (by proposition 8) $C(\pi_1,\pi_1'),\ldots,C(\pi_k,\pi_k')$ such that

$$C(\pi_1,\pi_1') > \ldots > C(\pi_k,\pi_k') > C(\pi_1,\pi_1')$$

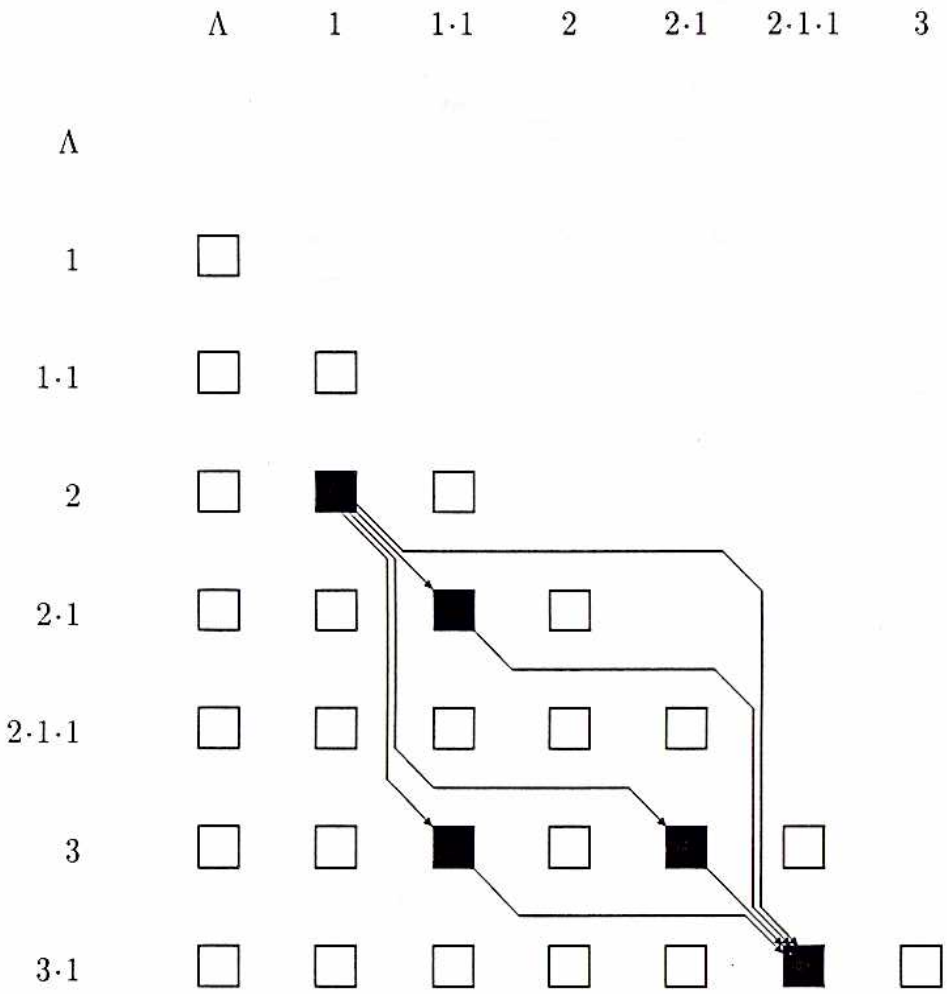

Figure 17: The occur check layer for the unification problem $\langle f(x, x, y) = f(g(y), g(g(b)), g(a))\rangle$.

iff $\{C(\pi_1, \pi_1'), \ldots, C(\pi_k, \pi_k')\} \subseteq C$ and there is a link from $C(\pi_i, \pi_i')$ to $C(\pi_{i+1}, \pi_{i+1}')$ for all $1 \leq i < k$ and a link from $C(\pi_k, \pi_k')$ to $C(\pi_1, \pi_1')$

iff for all $n \geq 0$ we find (by a simple induction on n)

$$\{C(\pi_1, \pi_1'), \ldots, C(\pi_k, \pi_k')\} \subseteq S^n(C)$$

iff $C^* \neq \emptyset$. qed

Combing back to our examples we find for the unification problem $\langle f(x,y) = f(g(y), g(x))\rangle$ that

$$C = \{C(1, 2{\cdot}1), C(2, 1{\cdot}1)\} = S(C)$$

and for $\langle f(x,x,y) = f(g(y), g(g(b)), g(a))\rangle$ that

$$\begin{array}{lcll} C & = & \{C(1,2), C(1{\cdot}1, 2{\cdot}1), C(1{\cdot}1, 3), C(2{\cdot}1, 3), C(2{\cdot}1{\cdot}1, 3{\cdot}1)\} & \\ S(C) & = & C \setminus \{C(1,2)\} & \\ S^2(C) & = & S(C) \setminus \{C(1{\cdot}1, 2{\cdot}1), C(1{\cdot}1, 3), C(2{\cdot}1, 3)\} & = \{C(2{\cdot}1{\cdot}1, 3{\cdot}1)\} \\ S^3(C) & = & S^2(C) \setminus \{C(2{\cdot}1{\cdot}1, 3{\cdot}1)\} & = \emptyset. \end{array}$$

The remaining open problem is how the occur check layer units are initially activated and externally inhibited. It is perfectly clear, where the activation has to come from, namely from the unification layer units of the form $U(\pi_1, \pi_1', x)$, where x is a variable. But there are two more facts to consider. The occur check can only be performed after the spreading of activation in the term and unification layer has stopped. Furthermore, the occur check layer units have to be activated by the respective unification layer units only once and then the occur check layer has to settle down without further activation from the unification layer. To accommodate these needs we assume that there is an *occur check request unit* CR. As far as this paper is concerned we assume that this unit will be activated by the user if he wants to perform an occur check and the spreading of activation in the term and unification layer has stopped[7]. Furthermore, we assume that for each unit $C(\pi_1, \pi_1')$ there are connections with weight 1 from

- $U(\pi_1, \pi_1', x)$ to $C(\pi_1, \pi_1')$ for all $x \in V$
- $U(\pi_1, \pi_1', x)$ to $C(\pi_1{\cdot}\pi, \pi_1'{\cdot}\pi)$ for all $x \in V$.

These connections are gated by CR such that activation is only spread from the unification layer to the occur check layer if CR is active. Note that after the activation of the occur check layer the CR-unit should be passive for the time the occur check layer needs to settle down. By proposition 9(3) we know that this will take no longer than n steps.

It remains to set up units which inhibit the occur check layer units. For each unit $C(\pi, \pi')$ we need an additional threshold unit $C'(\pi, \pi')$. This unit inhibits $C(\pi, \pi')$ and is activated by CR if there is no spreading of activation from the unification layer to $C(\pi, \pi')$. In figure 18 we have depicted the occur check request unit CR together with an occur check layer unit

[7]However, our connectionist model can easily be extended such that the activation of the unit CR is triggered by the fact, that no further unit is activated in the term and unification layer.

$C(\pi, \pi')$ and $C'(\pi, \pi')$. For simplicity we have drawn only one connection from the unification layer. The numbers in the units are the thresholds and the numbers at the connections are the weights, where $\alpha = \frac{1}{2} \times n \times (n-1)$. As soon as CR is activated activation may spread from the unification to the occur check layer since it is no longer blocked. Note that the two gates behave differently. An activation of CR opens the gate g_1, whereas an activation from the unification layer closes gate g_2.

Altogether, to test whether a DSE-relation is acyclic requires $\frac{2}{2} \times n \times (n-1)+1 = n^2 - n + 1$ additional units and the test is performed in at most $n + 1$ steps.

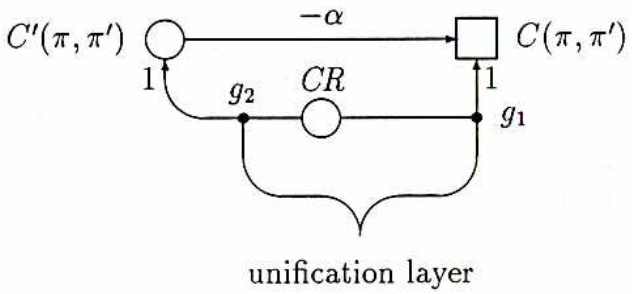

Figure 18: The occur check request unit CR with the occur check layer unit $C(\pi, \pi')$ and the unit $U'(\pi, \pi')$.

4.6 Determining the Most General Unifier

In section 4.2 we have argued that the final activation of the term layer units can be viewed as the output of the connectionist unification algorithm. This has several reasons. The unification algorithm is usually part of a larger inference system and the user is in most cases not interested in the solution of a particular unification but in the final answer substitution for a initial goal. Such a final answer substitution usually contains only few variable bindings compared to the set of bindings computed during the deduction. More importantly, we expect that a connectionist inference system using our unification algorithm will be based on the same representation for terms as the unification algorithm itself and, thus, uses the most general unifier of two expressions as represented by the final activation of the term layer. Nevertheless, in this paragraph we will show how the most general unifier can be determined by the active units of the term layer. Our approach follows the technique described by Paterson & Wegman [1978] and explores the partial ordering among the $\sim$-equivalence classes.

Since the terms are represented as occurrence-label pairs we need a function which generates expressions from the representation. This function is called *term* and will be applied to a $\sim$-equivalence class $[\pi]$. If this class is labelled by one or more variables, then *term* selects

one of this variables, say x. I.e.

$$term([\pi]) = x.$$

Otherwise, this class is labelled by a function symbol f and let us assume that f is n-ary. In this case *term* has to be applied recursively to compute the arguments of f. I.e.

$$term([\pi]) = f(term([\pi]\cdot 1), \ldots, term([\pi]\cdot n)),$$

where $[\pi]\cdot i$ denotes the $\sim$-equivalence which contains an occurrence $\pi'\cdot i$ such that $\pi' \in [\pi]$. Using the function *term* we can now determine the most general unifier. Therefore, let E be the set of those $\sim$-equivalence classes whose labels contain a variable and let σ be the empty substition. Do while $E \neq \emptyset$.

1. Select a $\succ$-maximal element $[\pi]$ from E[8].
2. If $label([\pi])$ contains only variables then select a variable x from the labels and let
$$\sigma \leftarrow \{x \leftarrow y \mid y \in label([\pi]) \setminus \{x\}\}\,\sigma.$$
3. Otherwise, $label([\pi])$ contains exactly one function symbol f and assume that f is n-ary. Let
$$\sigma \leftarrow \{x \leftarrow f(term([\pi]\cdot 1), \ldots, term([\pi]\cdot n)) \mid x \text{ is a variable in } label([\pi])\}\,\sigma.$$
4. $E \leftarrow E \setminus \{[\pi]\}$.

All selections in the algorithm have to be interpreted as don't care non-deterministic features, i.e. we may make an arbitrary selection and have not to worry about the other alternatives. The various possibilities account for the fact that a most general unifier is unique modulo variable renaming [Fages and Huet, 1986]. Note that the labels for each occurrence can be obtained from the term layer by inspection.

For our running example $\langle f(x,x,y) = f(g(y), g(g(z)), g(a))\rangle$ we find the $\sim$-equivalence classes $[\Lambda], [1,2], [1\cdot 1, 2\cdot 1, 3]$ and $[2\cdot 1\cdot 1, 3\cdot 1]$. Since Λ is only labelled by the function symbol f, E will initially contain only the remaining 3 classes. The $\succ$-maximal element of E is $[1,2]$ with labels $\{g, x\}$. Hence, case 3 applies and we obtain

$$\begin{aligned}\sigma &= \{x \leftarrow g(term([1,2]\cdot 1))\}\\ &= \{x \leftarrow g(term([1\cdot 1, 2\cdot 1, 3]))\}\\ &= \{x \leftarrow g(y)\}.\end{aligned}$$

From the remaining two equivalence classes $[1\cdot 1, 2\cdot 1, 3]$ is the $\succ$-maximal element and in analogy to the previous case we obtain

$$\sigma = \{y \leftarrow g(z)\}\{x \leftarrow g(y)\}.$$

Similarly, the final equivalence class yields

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

which is equal to

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

and is indeed the most general unifier.

[8] $[\pi]$ is called *root class* in [Paterson and Wegman, 1978].

4.7 Some Examples

The following table shows several unification problems $\langle s = t \rangle$ together with

- the size $n = |O(s) \cup O(t)|$,
- the number k of steps needed to compute the finest DSE-relation,
- the number l of steps needed to check that the finest DSE-relation is acyclic,
- and the most general unifier σ.

Note that the check whether a DSE-relation $\sim$ is acyclic requires at least 1 step, whereas the homogeneity check requires 2 steps. Both checks can be performed independently. Hence, the time required to solve a unification problem is $k + max\{2, l\}$.

	$\langle s = t \rangle$	n	k	l	σ
1	$\langle x = y \rangle$	1	0	1	$\{y \leftarrow x\}$
2	$\langle x = f(y, g(y)) \rangle$	4	1	2	$\{x \leftarrow f(y, g(y))\}$
3	$\langle f(x, g(x, a, a)) = f(x, g(x, a, a)) \rangle$	6	1	2	ε
4	$\langle f(x, x) = f(g(a), y) \rangle$	5	3	2	$\{x \leftarrow g(a), y \leftarrow g(a)\}$
5	$\langle f(g(v), h(u, v)) = f(g(w), h(w, i(x, y))) \rangle$	8	7	2	$\{u \leftarrow i(x, y), v \leftarrow i(x, y), w \leftarrow i(x, y)\}$
6	$\langle f(x, h(y)) = f(g(a), h(g(a))) \rangle$	6	1	1	$\{x \leftarrow g(a), y \leftarrow g(a)\}$
7	$\langle f(x, g(x)) = f(a, y) \rangle$	4	3	2	$\{x \leftarrow a, y \leftarrow g(a)\}$
8	$\langle f(x, g(x)) = f(h(h(a)), g(h(a), h(a))) \rangle$	8	1	2	$\{x \leftarrow h(h(a))$
9	$\langle g(x_2, x_3) = g(f(x_1, x_1), f(x_2, x_2)) \rangle$	7	4	4	$\{x_2 \leftarrow f(x_1, x_1)\}\{x_3 \leftarrow f(x_2, x_2)\}$
10	$\langle g(x_2, x_3, x_4) = g(f(x_1, x_1), f(x_2, x_2), f(x_3, x_3)) \rangle$	10	4	4	$\{x_2 \leftarrow f(x_1, x_1)\}\{x_3 \leftarrow f(x_2, x_2)\}\{x_4 \leftarrow f(x_3, x_3)\}$
11	$\langle f(x, g(h(y))) = f(g(h(a)), g(h(a))) \rangle$	7	1	1	$\{x \leftarrow g(h(a)), y \leftarrow a\}$
12	$\langle f(x, x) = f(g(g(y)), g(g(z))) \rangle$	7	3	3	$\{x \leftarrow g(g(y)), z \leftarrow y\}$
13	$\langle f(x, g(x)) = f(g(y), g(z)) \rangle$	5	4	2	$\{x \leftarrow g(y), z \leftarrow g(y)\}$
14	$\langle f(x, x, y) = f(g(y), g(g(z)), g(a)) \rangle$	8	6	4	$\{z \leftarrow a, y \leftarrow g(a), x \leftarrow g(g(a))\}$
15	$\langle f(x, x, y_1, y_2) = f(g(y_1), g(g(y_2)), g(g(z)), g(a)) \rangle$	11	9	5	$\{z \leftarrow a, y_2 \leftarrow g(a), y_1 \leftarrow g(g(a)), x \leftarrow g(g(g(a)))\}$
16	$\langle f(x, x, a) = f(a, y, y) \rangle$	4	4	1	$\{x \leftarrow a, y \leftarrow a\}$
17	$\langle f(x_1, x_1, x_2, x_2, a) = f(a, y_1, y_1, y_2, y_2) \rangle$	6	7	2	$\{x_1 \leftarrow a, x_2 \leftarrow a, y_1 \leftarrow a, y_2 \leftarrow a\}$
18	$\langle f(x_1, x_1, x_2, x_2, x_3, x_3, a) = f(a, y_1, y_1, y_2, y_2, y_3, y_3) \rangle$	8	8	2	$\{x_1 \leftarrow a, x_2 \leftarrow a, x_3 \leftarrow a, y_1 \leftarrow a, y_2 \leftarrow a, y_3 \leftarrow a\}$

Example (1) shows that two variables can be unified in 1 step. Observe that $\{x \leftarrow y\}$ is also a most general unifier for this problem. In fact, as example (2) shows a variable can be bound to any term in 1 step if this term does not contain the variable. This comes not surprisingly since we have shown in theorem 7 that for a solvable maching problem $T^2(N) = T(N)$ holds. (1)-(3), (6), (8), and (11) are matching problems and (3) is also a word problem. Recall

that a matching problem over infinite trees is identical to the respective problem over finite trees. Hence, it is not necessary to test whether the finest DSE-relation is acyclic and $l = 0$ if these examples are treated as matching problems. Example (5) has been used by Paterson & Wegman [1978] to illustrate their sequential unification procedure. (8) is the matching problem from section 4.3. (9) and (10) have been used by Paterson & Wegman to show that a careless unification algorithm may need exponential time to compute the most general unifier. These examples can easily be generalized to contain p variables x_p. But in any case k will always be equal to 4. Example (14) is our running example and has been taken from Dwork et al. [1984]. The example can be generalized to contain p variables y_p (see (15)) in which case $k = 3 \times (p + 1)$ and $l = p + 1$. This example has been used by Dwork et al. to show that the parallel time for solving a unification problem is in the worst case proportional to the size of the unification problem. The reason for this behaviour is the fact that in these examples the axioms of decomposability and transitivity have to be applied alternatively for at least p times. Examples (4) and (12) show that a closure with respect to the axiom of decomposibility can be computed in 1 step, wheras examples (16) - (18) reflect the fact that a closure with respect to the axiom of transitivity can only be computed in $log(n)$ time.

4.8 Remarks

Though we have only considered the problem of unifying two terms it is easy to simultaneously unify several pairs of terms s_1, t_1, ..., s_n, t_n by unifying $f(s_1, ..., s_n)$ and $f(t_1, ..., t_n)$. Similarly, we can simultaneously unify more than two terms $t_1, \ldots, t_n$ by unifying $f(t_1, ..., t_n)$ and $f(x, ..., x)$, where x is a new variable.

The unification algorithm presented herein does not propagate potential non-unifiability as Stolcke's [1989c] algorithm does. As mentioned in section 2.6 this propagation is vital for Stolcke's approach since his algorithm is initialized by activating the node 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. As a consequence, if we consider only the part of the term layer which contains the units for the function symbols, then there will be at most one unit active in each row if the equivalence relation is homogeneous. In this special case a coarse coded representation of the function symbols is possible even without any cross-talk. For example, 20 function symbols can be represented by 6 units such that each unit represents 10 symbols and each symbol is uniquely represented by 3 units. In this case, the equivalence relation will not be homogeneous as soon as more than 3 units are active in a row.

5 Open Problems and Future Work

What is this unification algorithm good for besides showing that unification can be implemented in a connectionist system? Of course we would like to apply it within term rewriting, logic programming, or a theorem prover in order to tackle the problems mentioned in the introduction. But there are a lot of open problems that have to be solved.

So far we assume that all units of our layered architecture are prewired. In most applications, however, unification or matching problems arise dynamically and we have to find ways such that the unification algorithm is able to set up its connections itself. The terms to be unified are often subterms of larger structures and we have to update these structures if the unification algorithm terminates successfully. We expect that some form of recruitment learning ([Valiant, 1988; Diederich, 1988]) will do the job.

Each resolution step in a logic programming language requires the use of a new variant of a clause. These variants are obtained by renaming the variables in the program clauses. This amounts to using an unbounded set of variables.

As far as unification is concerned the set of possible occurrences is determined by the terms which define the unification problem. In an automated theorem prover, however, terms are created dynamically and we cannot predict the set of occurrences in advance.

The unification problem is a *nice* problem in that it admits always a most general unifier iff it is solvable. Unfortunately, as soon as we are interested in unification under certain equational theories, we have to deal with complete sets of unifiers (see [Siekmann, 1989] for an overview). These sets may be even infinite. Similarly, derivations in theorem provers need not be finite. How are we going to deal with potentially infinite computations in a connectionist theory?

References

[Ajjanagadde and Shastri, 1989] V. Ajjanagadde and L. Shastri. Efficient inference with multi-place predicates and variables in a connectionist system. In *Proceedings of the Annual Conference of the Cognitive Science Society*, pages 396–403, 1989.

[Anandan *et al.*, 1989] P. Anandan, S. Letovsky, and E. Mjolsness. Connectionist variable-binding by optimmization. In *Proceedings of the Annual Conference of the Cognitive Science Society*, pages 388–395, 1989.

[Ballard, 1986] D. H. Ballard. Parallel logic inference and energy minimization. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 203 – 208, 1986.

[Barnden, 1984] J. A. Barnden. On short term information processing in connectionist theories. *Cognition and Brain Theory*, 7:25–59, 1984.

[Bibel, 1987] W. Bibel. *Automated Theorem Proving.* Vieweg Verlag, Braunschweig, second edition, 1987.

[Citrin, 1988] W. V. Citrin. Parallel unification scheduling in Prolog. Technical Report UCB/CSD 88/415, University of California, Berkeley, 1988.

[Colmerauer, 1982] A. Colmerauer. Prolog and infinite trees. In Clark and Tarnlund, editors, *Logic Programming*, pages 231–251. Academic Press, 1982.

[Colmerauer, 1984] A. Colmerauer. Equations and inequations on finite and infinite trees. In *Proceedings of the International Conference on Fifth Generation Computer Systems*, pages 85–99, 1984.

[Corbin and Bidoit, 1983] J. Corbin and M. Bidoit. A rehabilitation of robinson's unification algorithm. In *Proceedings of the World Computer Congress of the IFIP*, pages 909–914. North-Holland, 1983.

[Diederich, 1988] J. Diederich. Connectionist recruitment learning. In *Proceedings of the European Conference on Artificial Intelligence*, 1988.

[Dwork *et al.*, 1984] C. Dwork, P. C. Kannelakis, and J. C. Mitchell. On the sequential nature of unification. *Journal of Logic Programming*, 1:35–50, 1984.

[Fages and Huet, 1986] F. Fages and G Huet. Complete sets of unifiers and matchers in equational theories. *Journal of Theoretical Computer Science*, 43:189–200, 1986.

[Feldman and Ballard, 1982] J. A. Feldman and D. H. Ballard. Connectionist models and their properties. *Cognitive Science*, 6(3):205–254, 1982.

[Fodor and Pylyshyn, 1988] J. A. Fodor and Z. W. Pylyshyn. Connectionism and cognitive architecture: A critical analysis. In Pinker and Mehler, editors, *Connections and Symbols*, pages 3–71. MIT Press, 1988.

[Hager and Moser, 1989] J. Hager and M. Moser. An approach to parallel unification using transputers. In *Proceedings of the German Workshop on Artificial Intelligence*, 1989.

[Herbrand, 1930] J. Herbrand. Sur la theorie de la demonstration. In Goldfarb, editor, *Logical Writings (1971)*. Cambridge, 1930.

[Ito *et al.*, 1985] N. Ito, H. Shimizu, M. Kishi, E. Kuno, and K. Rokusawa. Data-flow based execution mechanisms of parallel and concurrent Prolog. *New Generation Computing*, 3:15–41, 1985.

[Kirchner, 1984] C. Kirchner. A new equational unification method: A generalisation of Martelli-Montanari's algorithm. In *Proceedings of the Conference on Automated Deduction*, pages 224–247, 1984.

[Kowalski, 1979] R. Kowalski. *Logic for Problem Solving*, volume 7 of *Artificial Intelligence*. North Holland, New York/Oxford, 1979.

[Lange and Dyer, 1989a] T. E. Lange and M. G. Dyer. Frame selection in a connectionist model of high-level inferencing. In *Proceedings of the Annual Conference of the Cognitive Science Society*, pages 706–713, 1989.

[Lange and Dyer, 1989b] T. E. Lange and M. G. Dyer. High-level inferencing in a connectionist network. Technical Report UCLA-AI-89-12, Artificial Intelligence Laboratory, UCLA, 1989.

[Lloyd, 1984] J. W. Lloyd. *Foundations of Logic Programming.* Springer, 1984.

[MacQueen *et al.*, 1984] D. MacQueen, G. D. Plotkin, and R. Sethi. An ideal model for recursive polymorphic types. In *Proceedings of the ACM Symposium on Principles of Programming Languages*, 1984.

[Maher, 1988] M. J. Maher. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *Symposium on Logic in Computer Science.* IEEE, 1988.

[Marriott and Sondergaard, 1988] K. Marriott and H. Sondergaard. On Prolog and the occur check problem. Technical Report 88/21, Department of Computer Science, University of Melbourne, 1988.

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

[Mjolsness *et al.*, 1989] E. Mjolsness, G. Gindi, and P. Anandan. Optimization in model matching and perceptual organization. *Neural Computation*, 1:218–229, 1989.

[Paterson and Wegman, 1978] M. S. Paterson and M. N. Wegman. Linear unification. *Journal of Computer and System Sciences*, 16:158–167, 1978.

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

[Shastri and Ajjanagadde, 1989] L. Shastri and V. Ajjanagadde. A connectionist system for rule based reasoning with multi-placed predicates and variables. Technical Report MS-CIS-89-06, Department of Computer and Information Science, School of Engineering and Applied Science, University of Pennsylvania, Philadelphia, PA 19104-6389, 1989.

[Shastri and Ajjanagadde, 1990] L. Shastri and V. Ajjanagadde. From associations to systematic reasoning: A connectionist representation of rules, variables and dynamic bindings. Technical Report MS-CIS-90-05, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, 1990.

[Shastri, 1988] L. Shastri. *Semantic Networks: An Evidential Formalization and its Connectionist Realization.* Research notes in Artificial Intelligence. Pitman, London, 1988.

[Shieber, 1986] S. M. Shieber. An introduction to unification-based approaches to grammar. CSLI Lecture Note Series, Center for Study of Language and Information, Stanford, 1986.

[Siekmann, 1989] J. H. Siekmann. Unification theory. *Journal of Symbolic Computation*, 1989.

[Singhal and Patt, 1989] A. Singhal and Y. N. Patt. A high performance Prolog processor with multiple function units. In *Proceedings of the 16th Annual International Symposium on Computer Architecture*, pages 195 - 202, 1989.

[Smolensky, 1987] P. Smolensky. 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, 1987.

[Smolensky, 1988] P. Smolensky. On the proper treatment of connectionism. *Behavioral and Brain Sciences*, 11:1–74, 1988.

[Stolcke, 1989a] A. Stolcke. A connectionist model of unification. Technical Report TR-89-032, International Computer Science Institute, 1989.

[Stolcke, 1989b] A. Stolcke. Processing unification-based grammars in a connectionist network. In *Proceedings of the Annual Conference of the Cognitive Science Society*, pages 908 - 915, 1989.

[Stolcke, 1989c] A. Stolcke. Unification as constraint satisfaction in structured connectionist networks. *Neural Computation*, 1(4):559 - 567, 1989.

[Touretzky and Hinton, 1988] D. S. Touretzky and G. E. Hinton. A distributed connectionist production system. *Cognitive Science*, 12:423 - 466, 1988.

[Valiant, 1988] L. G. Valiant. Functionality in neural nets. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, pages 629 - 634, 1988.

[Vitter and Simons, 1986] J. S. Vitter and R. A. Simons. New classes for parallel complexity: A study of unification and other complete problems for P. *IEEE Transactions on Computers*, pages 403–418, 1986.