A Connectionist Model of Unification

Andreas Stolcke[1]

TR-89-032

May 1989

Abstract

A general approach to encode and unify recursively nested feature structures in connectionist networks is described. The unification algorithm implemented by the net is based on iterative coarsening of equivalence classes of graph nodes. This method allows the reformulation of unification as a constraint satisfaction problem and enables the connectionist implementation to take full advantage of the potential parallelism inherent in unification, resulting in sublinear time complexity. Moreover, the method is able to process any number of feature structures in parallel, searching for possible unifications and making decisions among mutually exclusive unifications where necessary.

Keywords and phrases. Unification, constraint satisfaction, connectionism, feature structures.

[1] Part of the research reported here is work towards a Diploma thesis at Technische Universität München, Munich, West Germany. The author is currently affiliated to the Computer Science Division of the University of California at Berkeley and supported by a Research Assistantship at the International Computer Science Institute.

1. Introduction

Unification is a basic algebraic operation at the heart of a wide variety of symbolic formalisms in Artificial Intelligence (AI). Among its most common uses is the unification of terms in (resolution) theorem proving. Some rudimentary cases of unification have been given a connectionist treatment, without achieving full generality, however. An often-cited paper is Ballard's work on connectionist resolution [1], which relies on unifications of terms without recursive embedding.

The motivation for this work stems from a different application of unification, namely the unification-based grammar formalisms widely used in linguistics (Lexical Functional Grammar, Functional Unification Grammar, etc.; cf. [6]). Therefore, the structures operated on are not terms but *feature structures* (*f-structures*, for short), i.e. recursively embedded matrices of feature-value pairs, much in the spirit of frame-like data-structures traditionally found in AI. The feasibility and usefulness of such structures in connectionism has been demonstrated in a number of models [2, 5, 9]. On developing our approach, we aimed at fully general representation and processing of arbitrarily embedded structures.

The connectionist approach to unification described in this paper was implemented and tested using an interactive network simulator based on LOOPS running on Interlisp workstations [10]. It was later extended and applied to generation of sentences from PATR-like unification-based grammars (not covered here, cf. [7]).

2. F-Structure Unification

2.1. Feature Structures

One way of representing f-structures is as sets of features and associated values. Features are just atomic symbols, whereas values can be either atoms or full-fledged f-structures. Such structures can then be written as recursive *feature matrices*, as in (1).

$$s_A := \begin{bmatrix} \mathit{shape}: & \mathit{rectangle} \\ \mathit{length}: & \begin{bmatrix} \mathit{value}: & 5 \\ \mathit{unit}: & \mathit{inch} \end{bmatrix} \\ \mathit{width}: & \begin{bmatrix} \mathit{value}: & 2 \\ \mathit{unit}: & \mathit{cm} \end{bmatrix} \end{bmatrix} \tag{1}$$

Individual feature values are written $s_A.\mathit{shape} = \mathit{rectangle}$, $s_A.\mathit{length.value} = 5$, etc.

An important property of f-structures is that substructures may be 'shared', i.e. one single embedded f-structure may be 'pointed to' from several parts of the surrounding structure. To represent this in matrix notation some kind of indexing of substructures is needed. Consider

$$s_B := \begin{bmatrix} \mathit{shape}: & \mathit{square} \\ \mathit{length}: & {}^{\mathbf{1}}\begin{bmatrix} \mathit{value}: & 5 \\ \mathit{unit}: & \mathit{inch} \end{bmatrix} \\ \mathit{width}: & {}^{\mathbf{1}} \end{bmatrix}, \tag{2}$$

where identical boldface superscripts indicate sharing of the corresponding feature value. Due to this property f-structures are sometimes more conveniently represented as directed acyclic graphs

(DAGs) with labeled edges (arcs). Substructures are mapped to internal nodes, atomic values to terminal nodes, and feature names are translated into edge labels. Thus, structure (2) corresponds to the DAG in Figure 1.

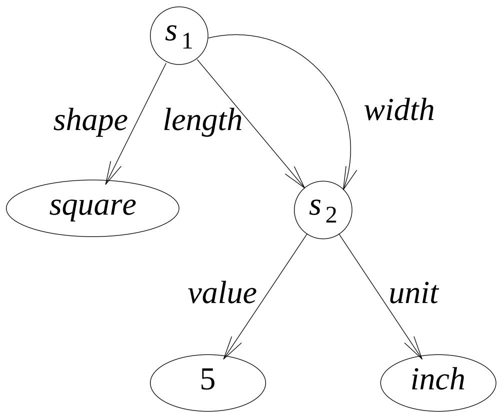


Fig. 1. DAG for f-structure (2).

Note that shared substructures are mapped to DAG nodes reachable from the root by more than one path. Multiple paths leading to the same substructure are refered to as *reentrancies*.

2.2. Unification

Intuitively, unification of two or more f-structures amounts to the creation of a new structure by *merging* the features contained in the original structures. The process is slightly complicated by two facts: First, due to the recursive nature of f-structures, unification has to proceed recursively, too. Thus, when unifying two f-structures both containing a non-atomic value for a certain feature, those values have to be unified recursively. Secondly, unification has to preserve all reentrancies contained in any of the unified structures.

As an example, reconsider structure (2); it may be obtained as the result of unifying the following structures (using $\sqcup$ as the unification operator):

$$\begin{bmatrix} \mathit{shape}: & \mathit{square} \\ \mathit{length}:^{\mathbf{1}} & [\,] \\ \mathit{width}: & \mathbf{1} \end{bmatrix} \sqcup$$

$$\begin{bmatrix} \mathit{length}: & \begin{bmatrix} \mathit{value}: & 5 \end{bmatrix} \end{bmatrix} \sqcup \tag{3}$$

$$\begin{bmatrix} \mathit{width}: & \begin{bmatrix} \mathit{unit}: & \mathit{cm} \end{bmatrix} \end{bmatrix} = s_B$$

A declarative definition would thus specify that the unification of a set of f-structures contain all and only those paths and reentrancies contained in either of the unified structures. In case identical paths in the operands lead to different atomic values, it is impossible to find such a ‘merged’ f-structure. In these cases the structures *don’t unify*, and the result of the unification is defined to be an ‘undefined’ value Ω, as in

$$\begin{bmatrix} \mathit{color}\colon & \mathit{red} \end{bmatrix} \sqcup \begin{bmatrix} \mathit{shape}\colon & \mathit{circle} \\ \mathit{color}\colon & \mathit{blue} \end{bmatrix} = \Omega$$

Here, the values *red* and *blue* for feature *color* clash, preventing unification.

Another special f-structure is the empty one [ ] which contains no features at all, and hence unifies with any f-structure as a neutral element of the operation.

A more formal treatment of unification is usually based on the lattice structure induced by the “more specific than” order between f-structures. f is said to be more specific than g iff f contains more paths and/or reentrancies than g. Relative to this partial order, we have [ ] as a minimum and Ω as a maximum, and the unification of a set of f-structures can be defined as their least upper bound.

Why f-strutures? As noted earlier, the method described here was based on f-structures because we had a linguistic application in mind. The reason for the popularity of f-structures in linguistics is a least two-fold: First of all, f-structures provide a simple and yet powerful formal representation of features associated with linguistic entities. For example, certain grammatical properties of noun phrases can naturally be expressed as an f-structure such as

$$\begin{bmatrix} \mathit{agreement}\colon & \begin{bmatrix} \mathit{person}\colon & \mathit{3rd} \\ \mathit{number}\colon & \mathit{plural} \\ \mathit{gender}\colon & \mathit{fem} \end{bmatrix} \end{bmatrix}.$$

Secondly, unification of such f-structures seems to naturally express a whole range of linguistic constraints and operations. As a typical example, consider the agreement of a subject noun phrase with its verb. This can easily be expressed as unifiability of the structure given above with a similar structure lexically associated with the verb. For an excellent survey of the role of unification in modern linguistic grammar theory see [6].

Term unification. Finishing this brief overview of unification, it is important to note that the type of unification usually found in logic-oriented applications in AI, term unification, can be reduced to f-structure unification in a straightforward way. This reduction involves an isomorphism from the set of terms into a subset of feature structures and affects in no way the applicability of the connectionist implementation presented in the following sections.

This map essentially translates argument positions into special features:

$$f(a_1, a_2, \ldots, a_n) \rightarrow \begin{bmatrix} \mathit{arity}\colon & n \\ \mathit{functor}\colon & f \\ \mathit{arg}_1\colon & a_1{}' \\ \mathit{arg}_2\colon & a_2{}' \\ \vdots & \vdots \\ \mathit{arg}_n\colon & a_n{}' \end{bmatrix},$$

where $a_i{}'$ is the mapping applied recursively to argument a_i. Variables are mapped to empty f-structures [ ], taking care that identical variables are mapped to identical [ ]’s (i.e. creating reentrancies).

This simple map constitutes a homomorphism relative to unification of terms and f-structures, respectively. Variable binding in terms corresponds to ‘filling’ empty f-structures by way of unifying them with other structures.

When talking about feasibility of (term) unification with connectionist means, it should be noted that virtually all of the potential applications of such a method would have to deal with thorny problems besides mere unification. For example, if one were to construct a connectionist theorem prover, an appropriate control structure for selecting resolvents would have to be devised. Nevertheless, a general tool for handling unification would certainly represent an important step toward any such application, besides its appeal from a theoretical point of view.

2.3. Unification as constraint satisfaction

A typical class of problems connectionist models excel in are constraint satisfaction type tasks. As a theoretical basis for the connectionist implementation described in the next section, we will now reformulate unification such that the operation becomes amenable to this kind of approach.

This reformulation hinges on the fact that the unification of a set of f-structures is equivalent to a certain type of equivalence relation on its node set $N \cup V$, where N are the internal DAG nodes and V is the set of atomic values (terminal nodes).

Consider the DAGs involved in the unification given in (3), as depicted in Figure 2. It is obvious from the graphical representation that unification implies a mapping from the set of nodes of the operands to the set of nodes of the unified structure. In the example, nodes s_1, s_3, and s_5 are mapped to s_7, while s_2, s_4, and s_6 are mapped to s_8.

This mapping can be made precise by observing that if node x in one of the operands is reached from the root by path p, then it will be mapped to the node that is reached from the root of the unification by following the same path p. This mapping, in turn, induces an equivalence relation on the set of nodes of the operands. In our case, we obtain the following partitions of the operand node set: $\{s_1, s_3, s_5\}$, $\{s_2, s_4, s_6\}$, $\{square\}$, $\{5\}$, $\{cm\}$.

Partitionings (equivalences) on node sets induced by unification are called *valid* and have the following properties (using $\sim$ to denote equivalence and $x.f$ for the value of feature f in structure x):

(a) For any pair of edges $x.f = x'$ and $y.f = y'$, $x \sim y$ implies $x' \sim y'$.

(b) For all atomic nodes (values) x and y, $x \sim y$ implies $x = y$.

(c) For all atomic nodes x and non-atomic nodes y, $x \sim y$ implies that $y.f$ is undefined, for all features f.

The term *validity* is chosen to conform to the terminology used by Paterson and Wegman [4], who used the idea of equivalence classes to devise a linear sequential algorithm for term unification.[2]

It can be shown that the existence of a valid equivalence relation with nodes $x_1 \sim x_2 \sim \cdots \sim x_n$ is both necessary and sufficient for the unifiability of the f-structures rooted in $x_1, x_2, \ldots, x_n$. Moreover, if we take the coarsest of all equivalence relations satisfying these conditions, the equivalence classes themselves can be viewed as the resulting unified structure. Inverting the

[2] I am grateful to Daniel Hernandez for pointing me to the commonalities between Paterson and Watson’s algorithm and the connectionist approach described in this paper.

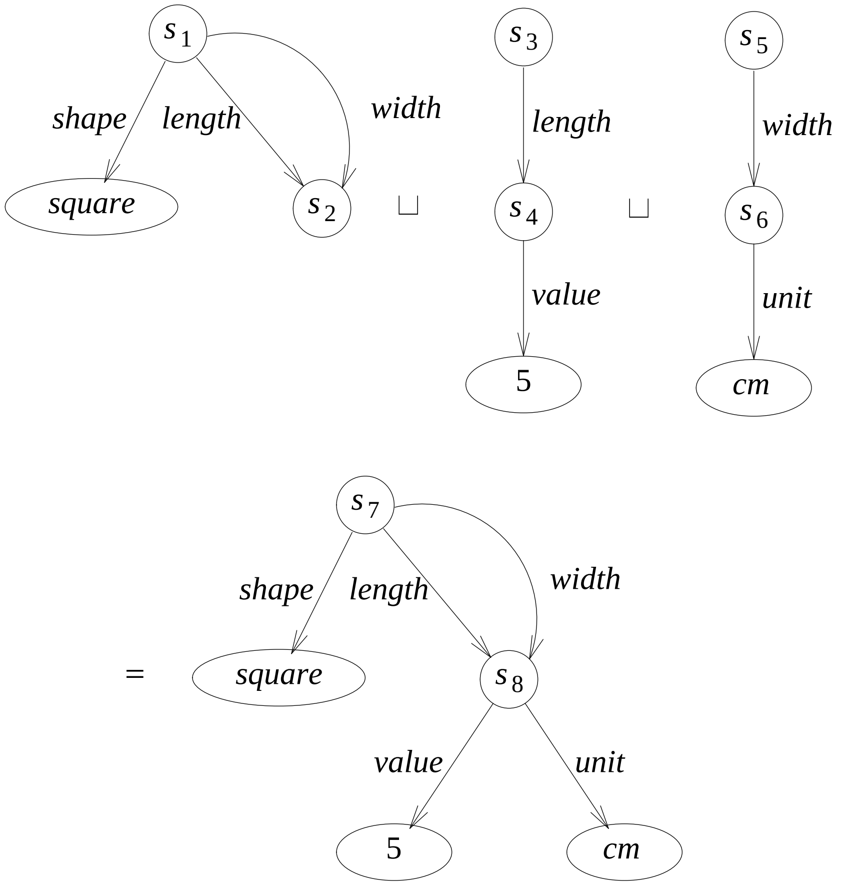


Fig. 2. DAGs corresponding to unification (3).

mapping discussed above, define each class as an f-structure node, and define features (edges) $X.f = Y$, for all classes X and Y with some $x \in X$ and $y \in Y$, such that $x.f = y$. The resulting f-structure will be well-defined as a consequence of the validity of ~, and isomorphic to the unification $x_1 \sqcup x_2 \sqcup \cdots \sqcup x_n$. A full formal discussion of the relationship between unification and valid equivalence relations can be found in [7].

As far as the connectionist implementation of unification is concerned, valid equivalence relations provide a basis for both representation and computation of unifications. Representation of node equivalences in terms of unit activations can be realized in a relatively straightforward way, as the next section will show. Validity can then be enforced as a set of constraints on the activation patterns, encoded by a special link structure.

3. A Connectionist Approach to Unification

We will give a detailed description of how a connectionist implementation of unification can be derived on the basis of the formal treatment outlined in the previous section.

The net will consist essentially of two spaces of linear threshold units,[3] one representing the f-

[3] As will become clear later, units of a more general type than linear threshold ones would have been convenient for the implementation of our model. The restriction to this basic type of processing elements is mainly due to historical

structures to be operated on, and the other one representing node equivalences on f-structures nodes. Links both internal to and between these two spaces will then enforce constraints on the activity of these units from which the desired behavior will follow.

3.1. Representing f-structures

For the discussion of the connectionist representation of f-structures and the implementation of unification, it is convenient to switch to DAGs as the primary formalism.

We chose to represent f-structures in a straightforward localist fashion, assigning one unit to each edge we (potentially) want to represent. The units allocated for this task can be viewed as being arranged in three dimensions, corresponding to the two nodes and the feature label, respectively. For instance, the presence of a value y for feature f in structure x is represented by an activation of unit $\langle x.f = y \rangle$ (we use angle brackets to denote units by indicating their symbolic 'meaning'). The units thus representing the edges of the f-structure DAGs are called *e-units*. The scheme adopted here is a localist version of the recursive representation of LISP S-expressions in BoltzCONS [8].

Given a set of features F, a set of atomic feature values V, and set of internal f-structure nodes N (which should be regarded as a pool out of which structure representations can be allocated), a total of $|N| \cdot |F| \cdot (|N| + |V|)$ e-units would be needed to allow any f-structure to be represented. Note, however, that no single set of f-structures would actually make use of all these e-units, since features have to well-defined, i.e. only one edge labeled f may start at node x (for any $f \in F$ and $x \in N$).

Fortunately, in many applications some or all of the f-structures involved in a task are given statically, the dynamical aspect being the changing unifications (or other operations) among them.[4] In such cases there is a fixed set of edges E, and precisely $|E|$ e-units are needed. Fixed f-structures allow further optimization of other parts of the net as well, and will be discussed later.

3.2. Representing node equivalence

Equivalence of two f-structure nodes x and y is represented by activation of a dedicated unit $\langle x \sim y \rangle$. It turns out that it is not sufficient to represent only equivalence of nodes, but also their non-equivalence where applicable. This is because validity constraints specify not only which nodes must fall in one equivalence class, but also which must not do so (cf. the contrapositives of the implications (a) through (c)). Hence we will also need a set of nodes $\langle x \not\sim y \rangle$ indicating non-equivalence of x and y. Informally, equivalence of f-structure nodes means unifiability of the corresponding f-structures, therefore the units encoding equivalence and non-equivalence are called *u-units* and *nu-units*, respectively.

---

reasons, related to the particular network simulator used [10]. The simulator was developed concurrently with this first application and initially featured only a restricted choice of unit models.

[4] For instance, when applying this kind of representation to unification-based grammars, each grammar rule can be encoded as a single f-structure DAG, such that unification of certain nodes corresponds to rule application (for details cf. [7]).

3.3. Enforcing consistency

Before tackling the problem of validity, it has to be ensured that the activation patterns on the space of u-units and nu-units be consistent, i.e. actually represent an equivalence relation. Reflexivity can be just assumed and taken into account when constructing the relevant parts of the network, i.e. we don't need to explicitly provide a set of units $\langle x \sim x \rangle$, which would have to be constantly activated anyway.

Similarly, symmetry can be encoded implicitly by collapsing nodes $\langle y \sim x \rangle$ with $\langle x \sim y \rangle$, and likewise for nu-units. All in all, we get a two-dimensional arrangement of u-units and nu-units with each type of unit occupying one half-plane and the diagonal missing. Thus, a total of about $(|N|+|V|)^2$ u/nu-units is needed.

To encode the constraint of transitivity on ~, an explicit link structure is needed. For every group of three u-units $\langle x \sim y \rangle$, $\langle y \sim z \rangle$ and $\langle x \sim z \rangle$, activity of any two of them should cause activation of the third.

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

Thus, we have pairs of units which jointly activate a third one. There are several ways to realize this kind of conjunctive activation using 'neural primitives', such as multiplicative connections or link modifiers (cf. [3]). In our case, since the implementation relies solely on linear threshold units, auxiliary intermediate units and links were set up to implement conjunctive activation, as shown in Figure 3. The units allocated for 'mediating' transitivity are called *t-units* and depicted as squares A through E. Their shape is meant to identify them as conjunctive units, i.e. their threshold is set up such that all inputs have to be active (equal to 1) to activate the unit itself. Other units (represented as ellipses) generally work disjunctively, meaning that one active input is sufficient to exceed the unit's threshold turning it on.[5] The connections linking t-units to u-units and nu-units are called *t-links*, and are shown in Figure 3 without special labels.

Unfortunately, this setup of t-units and t-links creates an additional problem for the operation of the net. For example, once all three u-units in Figure 3 have become active (as a consequence of just two of them receiving activation from third sources), they form a stable coalition which sustains itself even if the 'independent evidence' is removed.

To prevent this, t-units belonging together have to be made mutually exclusive by wiring them up as a winner-take-all network. In the example, this is accomplished by strong inhibitory *x-links* between A, B, and C. Similarly, D and E are made mutually exclusive. This arrangement prevents stable coalitions without external input and still allows propagation of transitivity.

Another concern in ensuring consistency of the representation is the coordination of u-units with their corresponding nu-units. As will become clear in the following, the 'evidence' activating nu-units is stronger than the one for u-units. Therefore, strong inhibitory *nu-links* allow the nu-units to suppress activation of their counterparts once they become active.

This finishes the setup of the link structure as far as consistency in concerned. The overall effect of this structure is that the only stable activation patterns on the space of u/nu-units are those consistently representing equivalence relations (partitionings) of f-structure nodes.[6]

[5] Precise values for unit thresholds and link weights are ommitted here, focussing instead on the functionality of the net structure. Please refer to the appendix for a complete specification.

[6] Technically, this can only be guaranteed for asynchronous operation of the network.

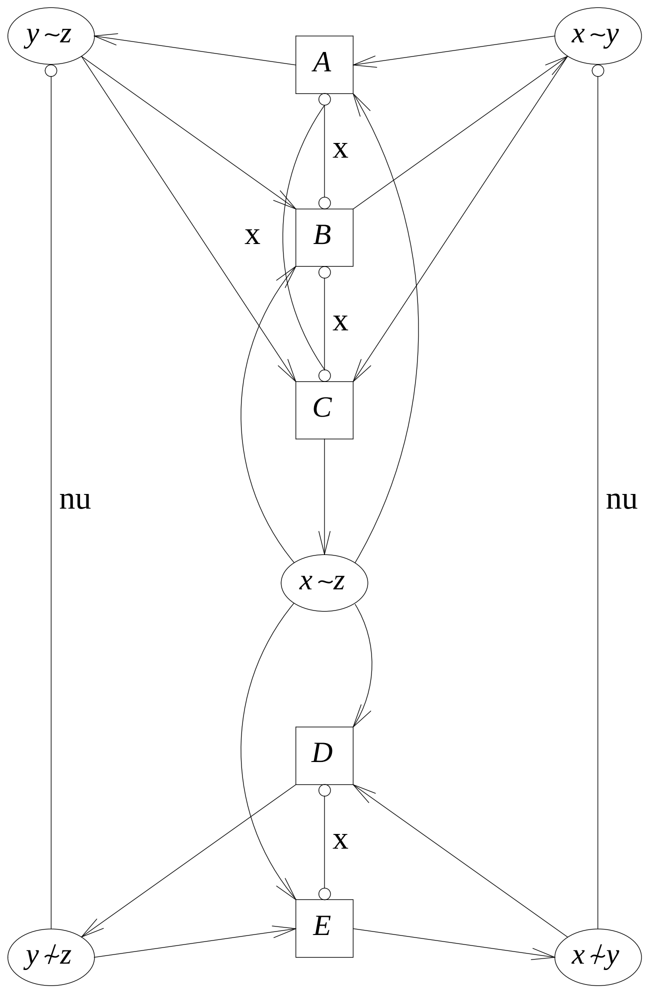


Fig. 3. Link structure enforcing transitivity.

3.4. Enforcing validity

Reconsider validity conditions (a) and (b). Taking an approach similar to the one adopted for transitivity, these implications can be implemented by additional links and conjunctive units.

Condition (a) translates into the connectivity pattern shown in Figure 4a. The left half of the pattern allows activation representing equivalence (i.e. unifiability) to spread top-down along paths of identical features. I.e., whenever the net (tentatively) unifies nodes x and y, and these nodes happen to have edges for the same feature f ($x.f = x'$ and $y.f = y'$), their child nodes x' and y' should be unified as well. Again, this conjunctive behavior is realized by a mediating *td-unit* (denoted A in the figure) and set of *td-links*.

The question arises where the initial activation for this top-down flow of activation comes from. As will be seen in more detail in the section decribing the operation of the net, the process is initiated by an external excitation of the u-unit representing the equivalence of the root nodes of the structures to be unified.

The right half of the link structure in Figure 4a is set up symmetrically, and corresponds to the contrapositive of condition (a). This part allows activation representing non-unifiability to flow in a bottom-up direction. Accordingly, we have *bu-units* and *bu-links* for this purpose.

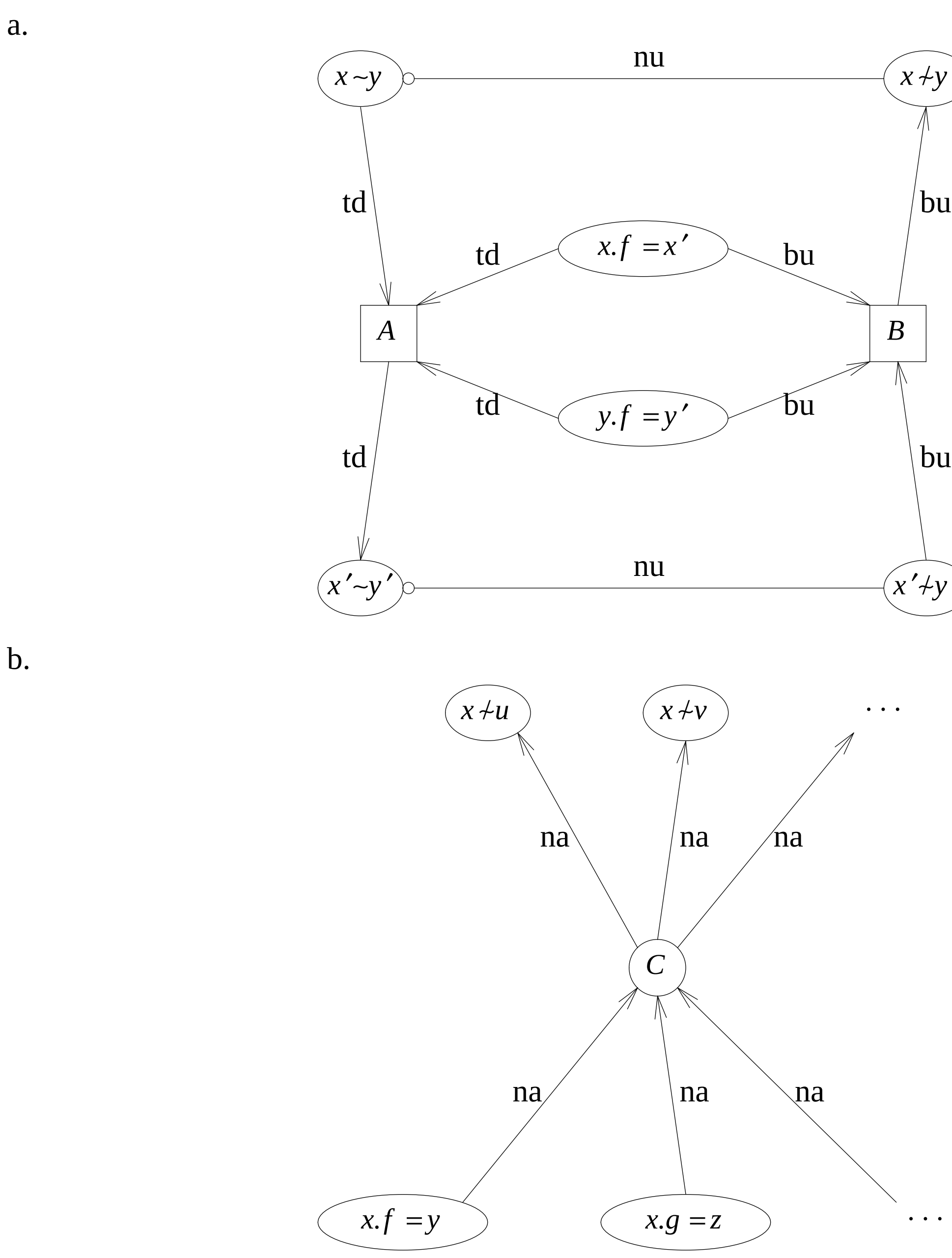


Fig. 4. Link structure enforcing validity.

Here, too, some source has to provide initial input for this type of activation flow. It turns out that this occurs due to validity conditions (b) and (c), which specify that certain atomic (leaf) nodes of an f-structure can not be equivalent (unify). In particular, (b) says that different atomic values can never be equivalent. This could be realized by clamping nu-units of the form $\langle u \not\sim v \rangle$, $u, v \in V$, in an 'on' state. The same effect can be achieved at less cost, however, by simply lowering the threshold of all the bu-units that would receive input from $\langle u \not\sim v \rangle$ by an amount equal to the weight of the corresponding bu-link. The nu-unit and the bu-link can then be dispensed with altogether. Simlarly, the u-unit $\langle u \sim v \rangle$ may be dropped since it would have to be constantly inactive anyway.

Validity condition (c) says that a non-atomic node $x \in N$, once it has any features, can never be equivalent to an atomic node $u \in V$. This gives rise to a linkage between e-units and nu-units as shown in Figure 4b. For every non-atomic node x, there is disjunctive *na-unit* C which transmits activation from e-units to nu-units using *na-links*. C wouldn't be strictly necessary but helps to save na-links.

3.5. Stability of the net

The overall link structure presented here guarantees that the only stable states of the net are those representing consistent and valid equivalence relations, which are, as discussed above, isomorphic to unifications of f-structures. Conversely, it is easily shown that whenever unification is possible (and hence a corresponding equivalence relation exists), the net has a stable configuration representing precisely that unification.

Intuitively, it is clear that the dynamics of the net will actually lead into such stable states if unification is possible. Although we haven't proved this formally, our simulations confirmed that the net reliably finds all possible unifications.

A potentially more serious problem is posed by the fact that the net is not guaranteed to have stable states for *non*-unifiable f-structures. In fact, simulations showed that a certain type of oscillating behavior occurred as a consequence of a specific case of non-unifiability. The next section will show, however, that even in these cases the net operates in a relatively well-behaved manner.

3.6. Time complexity

Unification potentially allows a large amount of parallelism. Whereever an f-structure branches in a tree-like way, i.e. without sharing of substructures, subtrees can be processed without interaction and hence in parallel.

The net presented here naturally explores these cases, needing time linear in the *depth* of the structure, as opposed to its size (number of nodes), as would be the case for a sequential algorithm. This is because we have to allow for both top-down and bottom-up activation to (simultaneously) cross the distance between root and leaves of the structures. Actually, when unifying two structures, time proportional to the depth of the smaller of the two structures is sufficient.

If units are forced to update their activity at least once within a certain period t, then the time needed to spread activation to a depth d will not exceed $2dt$. (The factor 2 is due to the relaying performed by td/bu-units.)

3.7. Space complexity

The size of the net in terms of its various types of units and links is summarized in Table 1. The space requirements are expressed in function of the number of edges $|E|$, of internal nodes $|N|$, of atomic values $|V|$, and of features $|F|$.

4. Operation of the net

At this point it has become evident why the inhibitory nu-links that coordinate between u-units and nu-units are asymmetrical. Whereas nu-units receive their 'evidence' from a violation of validity, u-units are activated merely as a direct or indirect consequence of an external attempt to make the root nodes equivalent (i.e. to unify the corresponding structures). This attempt may fail if the structures are not unifiable, in which case the nu-units will prevail, turning the corresponding u-units off. This in turn can be interpreted as a negative response of the net.

Table 1.

object type	*amount required*
e-units	$\lvert E\rvert$
u/nu-units	$\approx \lvert N\rvert\cdot(\lvert N\rvert+\lvert V\rvert)$
nu-links	$\approx \frac{1}{2}\lvert N\rvert\cdot(\lvert N\rvert+\lvert V\rvert)$
t-units	$\approx \frac{3}{2}(\lvert N\rvert+\lvert V\rvert)^3$
t-links	$\approx \frac{9}{2}(\lvert N\rvert+\lvert V\rvert)^3$
x-links	$\approx (\lvert N\rvert+\lvert V\rvert)^3$
td/bu-units	$\approx 2\frac{\lvert E\rvert^2}{\lvert F\rvert}$
td/bu-links	$\approx 8\frac{\lvert E\rvert^2}{\lvert F\rvert}$
na-units	$\lvert N\rvert$
na-links	$\lvert N\rvert\cdot\lvert V\rvert + \lvert F\rvert$

The standard mode of operation of the net can thus be summarized as follows:

- ‘Input’ all f-structures to the net by selecting their edges through activation of the corresponding e-units.
- Activate the u-units corresponding to intended unifications. I.e., to unify f-structures with root nodes x and y, excite $\langle x \sim y\rangle$.
- Run the net for an amount of time proportional to the depth of the f-structures. (An upper bound may be derived from the total size of the net).
- If the root equivalence originally activated remains active, the unification was successful and its result is encoded in the activities of u-units. Otherwise, the structures are not unifiable.

The last point needs some elaboration: The net will always show unifiability by letting the initially activated unit remain active. When unification fails, however, two cases are conceivable: either the unit is constantly turned off, or it is at least intermittently suppressed (the case of instability alluded to earlier).

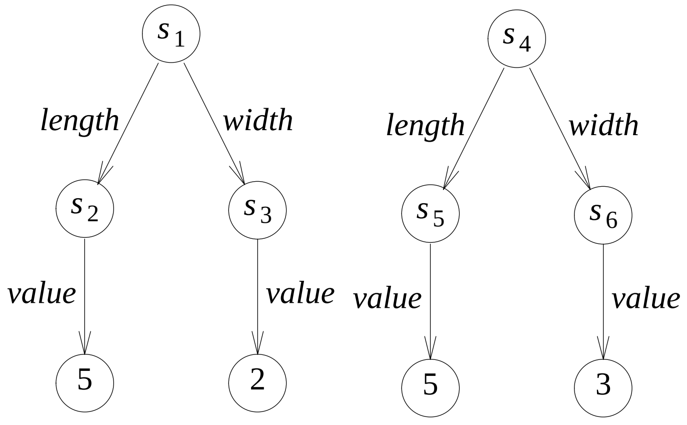
These different types of behavior are exemplified and analyzed in the following case studies.

4.1. Examples

As noted earlier, the net is guaranteed to find a unification if possible and settle into a stable state. For most cases the net is equally well-behaved if unification is not possible, and indicates failure by overruling the unit receiving external input. This case is illustrated in Figure 5, which shows two incompatible f-structures and the u/nu-units involved in their processing.

Two details of the graphical respresentation of the net structure in Figure 5b should be noted. The arrows pointing in vertical direction are really a shorthand for the combined effect of td-units/links and bu-units/links. These indirect connections are ‘enabled’ by pairs of e-units which are omitted in the picture. For instance, the activation path from $\langle s_1 \sim s_4\rangle$ down to $\langle s_3 \sim s_6\rangle$ is enabled by simultaneous activity of e-units $\langle s_1.g = s_3\rangle$ and $\langle s_4.g = s_6\rangle$, as an instance of the general scheme from Figure 4a. The same applies to all other top-down and bottom-up connections shown.

a.



b.

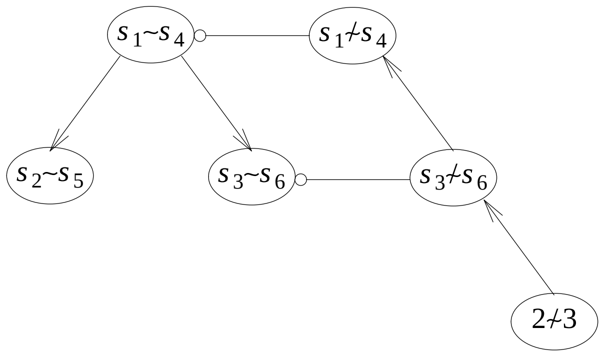


Fig. 5. A failed unification.

Furthermore, nu-unit $\langle 2{\not\sim}3\rangle$ exists only virtually because, as explained earlier, it is constantly active and can thus be omitted.

Activation throughout the net proceeds as follows: Initially, e-units and $\langle s_1{\sim}s_4\rangle$ are activated externally. This will cause $\langle s_2{\sim}s_5\rangle$ and $\langle s_3{\sim}s_6\rangle$ to become temporarily active. But at the same time, the incompatible leaves ‘2’ and ‘3’ will eventually cause nu-units $\langle s_3{\not\sim}s_6\rangle$ and $\langle s_1{\not\sim}s_4\rangle$ to go on, as a result of bottom-up activation. This supresses $\langle s_1{\sim}s_4\rangle$ and $\langle s_3{\sim}s_6\rangle$ via nu-links, and eventually turns $\langle s_2{\sim}s_5\rangle$ off as well, since it no longer receives top-down activation.

Note that if node ‘3’ had been replaced by ‘2’, or missing, unification would have been successful and all three u-units would have become permanently active.

The second example shows that in some cases non-unifiability can lead to oscillations of parts of the net, due to feedback paths being enabled. This typically happens when two structures fail to unify not because of incompatible leaves, but due to reentrancies in one of the structures. A minimal example is shown in Figure 6a. If the structure on the left weren’t reentrant (with two distinct nodes in place of s_2), unification would be possible (note that a non-atomic DAG node without outgoing edges corresponds to an empty f-structure, which unifies with any single other structure). It is the reentrancy that causes the two leaves from the *same* structure to clash. Figure 6b shows the activation paths relevant in this case. The dashed arrows are indirect connections due to transitivity, enabled by $\langle 2{\not\sim}3\rangle$. They close two negative feedback loops which allow $\langle s_1{\sim}s_3\rangle$ to turn itself off. I.e., as long as this unit receives external input it will oscillate, which can be taken as another indication of failed unification.[7]

[7] If the net is to be used as a module in a larger system, it would probably be necessary to smooth the output (integrat-

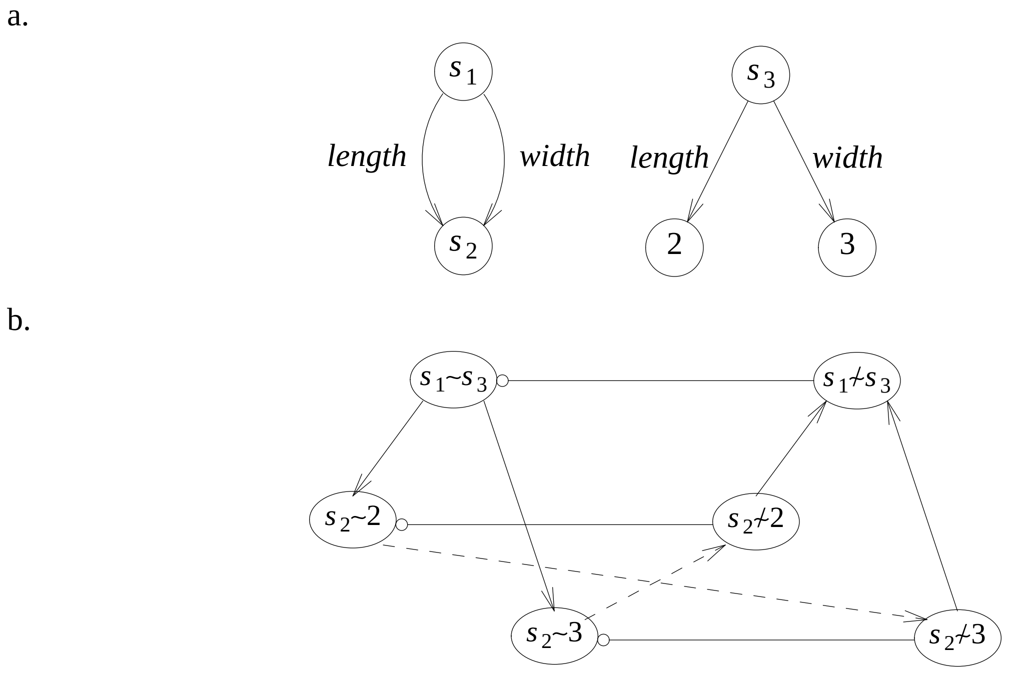


Fig. 6. Unification failed due to reentrancy.

4.2. Simultaneous unification of multiple f-structures

It should be obvious that nothing prevents the net from storing sets of more than two f-structures and perform their unification in parallel.[8] An interesting behavior arises when unifications between pairs of f-structures are mutually exclusive.

As an example, consider the three structures in Figure 7a. Here s_2 unifys with s_1 and s_3 individually, but the respective results don't unify with the remaining structure. This situation is reflected in the activation paths shown in Figure 7b. Dashed arrows represent t-links, A and B are t-units. Transitivity causes $\langle s_1{\sim}s_2\rangle$ and $\langle s_2{\sim}s_3\rangle$ to indirectly inhibit each other. Due to its asychronous character, the net will randomly 'chose' one of the possible unifications. The coalition of u-unit, t-unit, and nu-unit which is established first will then continuously suppress its rival.

This means that the net effectively tries to perform as many unifications as possible if told to do so, making random choices where necessary.

4.3. Optimizations

As noted earlier, in many applications the set of f-structures is partially fixed. This means that a considerable portion of the units in the net will end up always having the same activation, which can be precomputed.

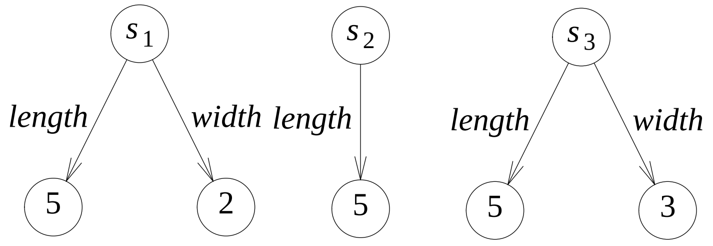
For instance, assume the structures in Figure 7 are given and remain fixed within a certain context, i.e. they are encoded constantly in the net. Then $\langle 2{\not\sim}3\rangle$ is known to be always active, but so is $\langle s_1{\not\sim}s_3\rangle$.

———————

ing over time) to allow its further processing.

[8] Unification is associative, so this task would normally be done sequentially.

a.



b.

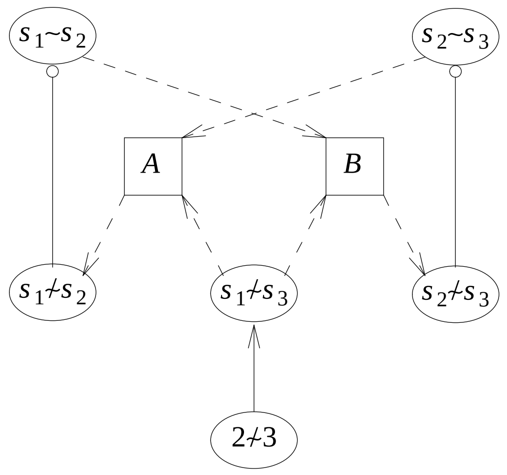


Fig. 7. Mutually exclusive unifications.

To save units and links, all units for which activation can be predetermined may be eliminated, as well as the links incident upon them. If the ‘virtual’ unit is constantly on, the thresholds of all the units it gives input to have to be lowered by the amounts of the corresponding link weights.

5. Shortcomings and Directions for Further Research

The connectionist approach to unification described here opens up several possibilities for future research. Our approach becomes relatively costly once a fully flexible representation of f-structures is wanted (with no fixed parts). This is due mainly to the inefficient, completely local encoding of edges in their threedimensional space of e-units. A distributed representation (e.g., using coarse coding methods) suggests itself. However, non-local representation of f-structures would inhibit a discrete (in fact, binary) processing, giving yes/no results for unifications. I.e., such an approach would probably have to depart from the exact definition of f-structure unification as it is implemented by the current version.

At this point one might ask whether the concept of unification can be generalized to some *continuous* measure of structured, recursive matching, more natural to connectionist processing. Interesting questions arise related to the application of such a notion to areas which currently employ unification in its standard, discrete form, like theorem proving and natural language grammars.

As noted earlier when discussing the relation of our method to term unification, a workable connectionist approach to this problem seems to raise more questions than it answers: How can the method be integrated with suitable control structures to yield solutions to classical problems in AI? Is it possible to reformulate these control problems (e.g., as constraint systems) in order to make them amenable to connectionist processing ([1] is a nice example for this type of approach)? How to avoid intractable space requirements when doing so? All these questions will certainly have to be dealt with in some way or other in future research.

6. Conclusions

The approach described in this paper shows that recursive processing of f-structures, and unification in particular, is feasible with connectionist means. Unification can be reformulated and implemented as a special form of constraint satisfaction problem, from which a connectionist implementation follows in a rather natural way.

The network thus created parallelizes unification as far as possible, while requiring a tolerable amount of units and links (at least by connectionist standards).

7. Acknowledgements

I am greatly indebted to all members of the AI research group at TU Munich, in particular to my thesis advisor Erwin Klöck. I would also like to thank Joachim Diederich and Jerry Feldman at ICSI for reading draft versions of this paper and providing valuable criticism.

8. References

[1] Dana H. Ballard, “Parallel Logical Inference and Energy Minimization”. In *Proceedings of the 5th National Conference on Artificial Intelligence*, Philadelphia, Pa., August 1986, pp. 203–208.

[2] Garrison W. Cottrell, “A Connectionist Approach to Word Sense Disambiguation”. Technical Report TR 154. Computer Science Department, University of Rochester, Rochester, N.Y., May 1985. Cf. especially chapter 7.

[3] Jerome A. Feldman, Dana H. Ballard, “Connectionist Models and Their Properties”. *Cognitive Science* **6**:205–254, 1982, 205–254.

[4] M. S. Paterson, M. N. Wegman, “Linear Unification”. IBM Research Report RC5904 (#25518). IBM Thomas J. Watson Research Center, Yorktown Heights, N.Y., 1976.

[5] Lokenda Shastri, *Semantic Networks: An Evidential Formalization and its Connectionist Realization*. San Mateo, Calif.: Morgan Kaufmann, 1988.

[6] Stuart M. Shieber, “An Introduction to Unification-Based Approaches to Grammar”. CSLI Lecture Note Series. Center for Study of Language and Information, Stanford, Calif., 1986.

[7] A. Stolcke, “Generierung natürlichsprachlicher Sätze in unifikationsbasierten Grammatiken. Ein konnektionistischer Ansatz”. Report FKI-95-88. Institut für Informatik, Technische Universität München, Munich, October 1988.

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

[9] David S. Touretzky, “Representing Conceptual Structures in a Neural Network”. In Maureen Caudill, Charles Butler (Ed.), *Proceedings of the 1st IEEE International Conference on Neural Networks*, San Diego, Calif., June 1987, pp. II-279–II-286.

[10] K. Zimmermann, “Der Netzeditor. Eine komfortable Umgebung zum Erstellen und Testen von konnektionistischen Netzen”. Report FKI-94-88. Institut für Informatik, Technische Universität München, Munich, 1988.

9. Appendix

Units in our model are exclusively *linear threshold units* u which update their activations a_u asynchronously according to their thresholds Θ_u and their inputs $a_v w_{vu}$:

$$a_i' = \begin{Bmatrix} 1 \\ a_i \\ 0 \end{Bmatrix} \text{ iff } \operatorname{sign}(\sum_v a_v w_{vu} - \Theta_i) = \begin{cases} +1 \\ \ \ 0 \\ -1 \end{cases}$$

To specify the thresholds and weights required to implement the functionality described in this paper, we refer to units and links by their type names introduced earlier. For example, Θ_{td} is the threshold on a td-unit, whereas w_{x} is the weight of an x-link.

Since there is an infinity of possible combinations of Θ's and w's, we merely give the inequalities specifying their relative values.

Transitivity (cf. Figure 3):

$$0 < \Theta_{\mathrm{u}} < w_{\mathrm{t}}$$

$$0 < \Theta_{\mathrm{nu}} < w_{\mathrm{t}}$$

$$w_{\mathrm{t}} < \Theta_{\mathrm{t}} < 2w_{\mathrm{t}}$$

$$2w_{\mathrm{t}} - \Theta_{\mathrm{t}} < -w_{\mathrm{x}}$$

Validity conditions (a) and (b) (cf. Figure 4a):

$$0 < \Theta_{\mathrm{u}} < w_{\mathrm{td}}$$

$$0 < \Theta_{\mathrm{nu}} < w_{\mathrm{bu}}$$

$$2w_{\mathrm{td}} < \Theta_{\mathrm{td}} < 3w_{\mathrm{td}}$$

$$2w_{\mathrm{bu}} < \Theta_{\mathrm{bu}} < 3w_{\mathrm{bu}}$$

$$Nw_{\mathrm{td}} + Mw_{\mathrm{t}} - \Theta_{\mathrm{u}} < -w_{\mathrm{nu}}$$

Validity condition (c) (cf. Figure 4b):

$$0 < \Theta_{\mathrm{nu}} < w_{\mathrm{na}}$$

$$0 < \Theta_{\mathrm{na}} < w_{\mathrm{na}}$$

Here N and M are constants determined by the network size, giving the maximum number of td-links and t-links leading to a u-unit, respectively.