A procedure for the construction of a similarity relation

Pascual Julián-Iranzo
Dep. of Information Technologies and Systems,
University of Castilla-La Mancha (Spain).
Pascual.Julian@uclm.es

Abstract

In this paper we present a procedure that generates a reflexive, symmetric, transitive closure of a fuzzy relation. We formally prove that this method is sound, in the sense that it is able to produce a similarity relation starting from an initial set of similarity equations. Also we discuss the implementation of the algorithm and a refinement is given.

Keywords: Fuzzy Logic Programming, Unification by Similarity, Fuzzy relations, Transitive Closure.

1 Introduction and Motivation

Fuzzy Logic Programming integrates fuzzy logic and pure logic programming in order to provide these languages with the ability of dealing with uncertainty and approximated reasoning. During the last decades a great effort has been done to incorporate fuzzy concepts into the logic programming framework. However, it is noteworthy that there is no common method for this integration (See for instance: [7, 1, 9] and [15]; as well as [3, 4] and [14]). A possible way to go, which is the one adopted in this paper, is to follow the conceptual approach introduced in [14] where the notion of "approximation" is managed at a syntactic level by means of similarity relations. The objective in this case is to make more flexible the query answering process. A similarity relation is an extension of the crisp notion of equivalence relation and it can be useful in any context where the concept of equality must be weakened. In [14] a new modified version of the **L**inear resolution strategy with **S**election function for **D**efinite clauses (SLD resolution) is defined, which is named *similarity-based* SLD resolution (or *weak* SLD resolution). This operational mechanism can be seen as a variant of the SLD resolution procedure where the classical unification algorithm has been replaced by the weak unification algorithm formally described in [14] (and reformulated in terms of a transition system in [8]). Informally, Maria Sessa's weak unification algorithm states that two terms $f(t_1, \ldots, t_n)$ and $g(s_1, \ldots, s_n)$ weak unify if the root symbols f and g are considered similar and each of their arguments t_i and s_i weak unify. Therefore, the weak unification algorithm does not produce a failure when there is a clash of two syntactical distinct symbols whenever they are similar.

Recently, in [8], we solved the problem of adapting the implementation of a Warren Abstract Machine (WAM) to incorporate the Sessa's weak unification algorithm. As a result, we obtained a Prolog implementation, that we call S-Prolog, with an operational semantics based on the weak SLD resolution principle of [14]. Essentially, the S-Prolog syntax is just the Prolog syntax but enriched with a built-in symbol "$\sim$" used for describing similarity relations by means of *similarity equations* of the form:

```
<symbol> ~ <symbol> = <degree>
```

meaning that two constants, n-ary function

symbols or n-ary predicate symbols are similar with a certain degree. More precisely, we use the built-in symbol "$\sim$" as a compressed notation for the symmetric closure of an arbitrary fuzzy binary relation (that is, a similarity equation $a \sim b = \alpha$ can be understood in both directions: a is similar to b and b is similar to a with degree α). Hence, a S-Prolog program is a sequence of Prolog facts and rules followed by a sequence of similarity equations. The next is a very simple example that serves to illustrate the S-Prolog syntax as well as some features of its operational behavior. Also it informally shows how S-Prolog is well suited for flexible query answering.

Example 1 *Consider the program* `Autumn` *that consists of the following clauses and similarity equations:*

```
% FACTS              warm :- sunny.
autumn.              rainy :- spring.
% RULES              cold :- winter.
warm :- summer.      happy :- warm.

% SIMILARITY EQUATIONS
spring ~ autumn = 0.7
spring ~ summer = 0.5
autumn ~ winter = 0.5
```

In a standard Prolog system a query as "`?- happy`*" fails, since we are specifying that it is* `warm` *if it is* `summer` *time (first rule) and, actually, it is* `autumn`*. Similarly, the query "*`?- rainy`*" fails also.*

However, the S-Prolog system is able to compute the following successful derivations[1]*:*

- $\langle \leftarrow happy, id, 1\rangle$
  $\Longrightarrow_{WSLD} \langle \leftarrow warm, id, 1\rangle$
  $\Longrightarrow_{WSLD} \langle \leftarrow summer, id, 1\rangle$
  $\Longrightarrow_{WSLD} \langle \Box, id, 0.5\rangle$.

  Here, the last step is possible because `summer` *weak unifies with the fact* `autumn`*, since there is a transitive connection between* `summer` *and* `autumn` *with approximation degree 0.5 (the minimum of 0.7 and 0.5). Therefore, the system answers "*`Yes, with approximation degree 0.5`*".*

- $\langle \leftarrow rainy, id, 1\rangle$
  $\Longrightarrow_{WSLD} \langle \leftarrow spring, id, 1\rangle$
  $\Longrightarrow_{WSLD} \langle \Box, id, 0.7\rangle$.

  In this case, the system answers "`Yes, with approximation degree 0.7`*" because* `spring` *and* `autumn` *weak unify with approximation degree 0.7 and the last step is possible.*

In general, S-Prolog computes answers as well as approximation degrees which are the minimum of the approximation degrees obtained in each step.

We have seen that the weak SLD resolution procedure used by the S-Prolog system works jointly with a similarity relation defined on a syntactic domain. However, it is not easy to define a similarity relation on a set of elements due to the transitivity constrains, which may contradict the initial similarity values. Therefore, it is convenient to let the programmer free for introducing a partial specification of a similarity relation, providing a set of *similarity equations*, which are used by the S-Prolog compiler to obtain a reflexive, symmetric, transitive closure of the set of similarity equations. Hence, producing the complete specification of the similarity relation.

In this paper we present a procedure that generates a reflexive, symmetric, transitive closure of a fuzzy relation (and therefore a similarity relation) containing the initial relation. We formally prove that this method is sound, in the sense that it is able to produce a similarity relation starting from an initial set of similarity equations. We discuss the implementation of the algorithm and a refinement is given. Also it is discussed how the algorithm can be generalized to obtain a new algorithm, parametric with regard the t-conorm ∇ and the t-norm $\triangle$ used for computing the transitive closure. The new algorithm is able to transform the initial fuzzy binary relation into a reflexive, symmetric, $\triangle$-transitive fuzzy binary relation. That is, a similarity relation. However, it is not the closure of the initial

[1] The symbol "`id`" denotes the identity substitution and "$\Box$" the empty clause.

relation.

2 Fuzzy Binary Relations and Similarity Relations

Given a set U, an ordinary subset A of U can be defined in terms of its *characteristic function* $\chi_A(x)$ (that returns 1 if $x \in A$ or 0 otherwise). On the other hand, a *fuzzy subset* A of U is a function $A : U \to [0,1]$. The function A is called the *membership function*, and the value $A(x)$ represents the *degree of membership* of x in the fuzzy subset A, being a generalization of the notion of characteristic function. Similarly, an ordinary binary relation on U is a subset of $U \times U$ and it can be identified by its characteristic function $U \times U \to \{0,1\}$. Therefore, the easy extension of this concept to the fuzzy case is to agree that, a *fuzzy binary relation* R is a fuzzy subset on $U \times U$ (that is, a mapping $R : U \times U \to [0,1]$). So, given two elements u_i and u_j in U, $R(u_i, u_j) = \alpha_{ij}$ represents the degree to which the pair $\langle u_i, u_j \rangle$ is compatible with the relation R.

Definition 2.1 *A* similarity relation *on a set U is a fuzzy binary relation $R : U \times U \to [0,1]$ holding the following properties:*

1. **(Reflexive)** $R(x,x) = 1$ *for any* $x \in U$*;*
2. **(Symmetric)** $R(x,y) = R(y,x)$ *for any* $x, y \in U$*;*
3. **(Transitive)** $R(x,z) \geq R(x,y) \triangle R(y,z)$ *for any* $x, y, z \in U$*;*

where the operator '$\triangle$' is an arbitrary t-norm[2]*.*

Sometimes, transitivity is qualified by an specific t-norm, $\triangle$, and it is called $\triangle$-*transitivity*.

In [10], when the operator $\triangle = \wedge$ (that is, it is the minimum of two elements), a similarity relation is called a *fuzzy equivalence relation*. Certainly, in this case, there exists a close relation between similarity relations and equivalence relations. The so called λ-cut[3] of a fuzzy equivalence relation R is an equivalence relation [10]. Since the λ-cut of R can be considered as a generalization of the identity relation, intuitively, a fuzzy equivalence on a set specifies when two elements may be considered equal with regard to a property that is not sharply defined.

Following [14], in the sequel, we restrict ourselves to similarity relations that are fuzzy equivalence relations. Moreover we are interested in fuzzy equivalence relations at a syntactic level. In other words, we deal with the minimum t-norm (and its dual t-conorm[4]: the maximum operator "$\vee$").

In general, a (fuzzy binary) relation R on a set U may or may not have some property $\mathcal{P}$, such as reflexivity, symmetry, or transitivity. If there is a relation R' which is the least relation holding property $\mathcal{P}$ and containing R, then R' is called the *closure* of R with respect to $\mathcal{P}$. In the fuzzy setting, a reasonable definition of the concept of a relation containing an other is the following.

Definition 2.2 *Let R and R' fuzzy binary relations on a set U. R is* contained *in R', denoted $R \sqsubseteq R'$, if and only if for each a, b in U $R(a,b) \leq R'(a,b)$. R' is the* least *relation containing R if and only if $R \sqsubseteq R'$ and for each fuzzy binary relation R'', such that $R \sqsubseteq R''$, $R' \sqsubseteq R''$.*

Let $D = \{\langle u, u \rangle \mid u \in U\}$ be the *diagonal relation* and R be a fuzzy binary relation on a set U. The *reflexive closure* of R can be formed by setting to one the compatibility degree of all pairs in D. That is, if we denote the reflexive closure of R by $R^{=}$, we are forcing that the new relation $R^{=}$ fulfills that $R^{=}(u,u) = 1$ for all $u \in U$. The *symmetric closure* of R is a relation $R^{\leftrightarrow}$ obtained by assigning the compatibility degree $R(b,a)$ of

[2] A t-norm $\triangle : [0,1] \times [0,1] \to [0,1]$ is a binary operator which is commutative, associative, monotone in both arguments and $1 \triangle x = x$ (hence, it subsumes the classical two-valued conjunction operator) [10].

[3] If R is a fuzzy binary relation on U, the binary relation $R_\lambda = \{(x,y) \mid \mathcal{R}(x,y) \geq \lambda\}$ is called the λ-*cut* of $\mathcal{R}$.

[4] A t-conorm $\nabla : [0,1] \times [0,1] \to [0,1]$ is a binary operator which is commutative, associative, monotone in both arguments and $0 \nabla x = x$ (hence, it subsumes the classical two-valued disjunction operator) [10].

each pair $\langle b, a\rangle$ in the domain of R to $R(a, b)$. So, we are making $R^{\leftrightarrow}(a, b) = R^{\leftrightarrow}(b, a)$ for all a, b in U. The *transitive closure* of R is a relation R^{+} such that, $R^{+}(a, b) = \alpha$ if there exists a sequence $a \equiv u_1, \ldots, u_n \equiv b$, with $n \geq 1$ and $R(u_i, u_{i+1}) = \alpha_{i(i+1)}$ for all $i = 1, \ldots, n-1$. The compatibility degree $\alpha = \alpha_{12} \triangle \cdots \triangle \alpha_{(n-1)n}$. In the sequel, we denote the reflexive, symmetric, transitive closure of R by the symbol $R^{\equiv}$. It is worth to mention that these definitions are assuming that compatibility degrees $R(b, a)$ (not necessary equal to zero) can be modified in order to fulfill the desired property. These are the least restricted notions of reflexive, symmetric and/or transitive closure of a fuzzy binary relation that can be conceived in a fuzzy context.

Before ending this section it is necessary to introduce several notations and concepts on representation of fuzzy relations that we shall use throughout the rest of the paper.

Let A be a finite set of cardinality n and assume we list the elements of A on an arbitrary sequence $\{a_1, a_2, \ldots, a_n\}$. Then a fuzzy binary relation R on A can be represented by a matrix $M = [m_{ij}]$ such that $m_{ij} = R(a_i, a_j)$. Sometimes we say that m_{ij} is the *entry* $\langle i, j\rangle$ of M. By reasons that soon will be evident, M is called the *adjacency matrix* of R.

Corresponding to any fuzzy binary relation R on A and its adjacency matrix representation M, there is a labeled *directed graph* (or *digraph*) G whose *nodes* (or *vertices*) are the members of the domain of R and whose labeled arcs are the triples $a_i \xrightarrow{\alpha_{ij}} a_j$ for which $R(a_i, a_j) = \alpha_{ij}$. A path (of length k) from node b to node c is a finite sequence of arcs $b \equiv b_0 \xrightarrow{\alpha_1} b_1 \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_k} b_k \equiv c$ such that the right node of each arc (other than the last) equals the left node of the succeeding arc. If $a \longrightarrow x_1 \longrightarrow x_2 \longrightarrow \cdots \longrightarrow x_l \longrightarrow b$ is a path from a to b, then $x_1, x_2, \ldots, x_l$ are its *interior vertices*. A simple path is a path in which no interior vertex is repeated. A cyclic path is a path whose first and end nodes are the same. The *successors* of a node a are all the nodes b for which there exists a path from a to b. Similarly, the *predecessors* of a node b are all the nodes a for which there exists a path from a to b. Node b is an *immediate successor* of node a if the arc $a \xrightarrow{\alpha} b$ exists, and *immediate predecessor* is dually defined.

In the adjacency matrix $M = [m_{ij}]$, representation of a relation R and its corresponding digraph G, the elements $m_{ik} \neq 0$ in row i identify the immediate successors a_k of node a_i. Similarly, the elements $m_{kj} \neq 0$ in column j identify the immediate predecessors a_k of node a_j. Hence, the name of "adjacency matrix".

3 A Direct Algorithm to Construct a Similarity Relation

In this section we present an algorithm able to transform a fuzzy binary relation R on a set A into a similarity relation $R^{\equiv}$ on the set A, which is the reflexive, symmetric, transitive closure of R (that is, the least fuzzy binary relation on the set A containing R and holding the reflexive, symmetric and transitive properties).

Before describing our algorithm we need to introduce the following definition.

Definition 3.1 *Let $M = [m_{ij}]$ be the $n \times n$ adjacency matrix representing a fuzzy binary relation R on a set A. An element $m_{ij} \neq 0$* preserves transitivity *if for each $k \in \{1, \ldots, n\}$, $m_{ik} \wedge m_{kj} \leq m_{ij}$*

As we shall see in the next section, this concept plays an important role in the soundness proof of the proposed algorithm.

Intuitively, if the elements of a matrix preserve transitivity, the relation represented by this matrix does not break one of the necessary conditions to be a similarity relation. Hence, we provide as input of the algorithm a matrix whose elements preserve transitivity and partially specify the similarity relation we can construct. To give this partially specified similarity relation it is enough to set some elements of the inferior triangular matrix[5] letting all the elements of the superior triangular

[5]The one formed by the elements below the diagonal.

matrix[6] set to zero. Then, the algorithm manipulates the initial matrix (trying to keep up the initial compatibility degrees as much as possible[7]) until a new one, holding the properties of a similarity relation, is constructed.

Algorithm 1

Input: *An adjacency matrix $M = [m_{ij}]$, representing a fuzzy binary relation R on a set A, whose elements preserve transitivity and with all the elements of the superior triangular matrix set to zero.*

Output: *The adjacency matrix $M^{\equiv}$ corresponding to the reflexive, symmetric, transitive closure of R.*

begin

Step 1. Build the reflexive closure of R: *for each entry $\langle i, i\rangle$ in M do $m_{ii} := 1$;*

Step 2. Build the symmetric closure of R: *for each entry $\langle i, j\rangle$ in M, such that $m_{ij} \neq 0$, do $m_{ji} := m_{ij}$;*

Step 3. Build the transitive closure of R: *for each column k and entry $\langle i, j\rangle$ in M do $m_{ij} := m_{ij} \vee (m_{ik} \wedge m_{kj})$; where "$\vee$" and "$\wedge$" are, respectively, the maximum and the minimum operators;*

Return $M^{\equiv} := M$.

end

Step 3 is an extension of the wellknown Warshall's algorithm [16] for computing the transitive closure of a relation, where the classical *meet* and *joint* operators on the set $\{0, 1\}$ have been changed by the *maximum* and the *minimum* operators on the real interval $[0, 1]$ respectively. A fact that makes Warshall's algorithm attractive is that it computes the transitive closure in only one pass over M (in the sense that each element is tested once), a fact that is not obvious [16]. To get the intuition behind this third step some more comments are needed:

- The computation of the transitive closure of a digraph G amounts to adding a minimal set of arcs to G that makes all successor and predecessor relationships in G immediate successor and predecessor relationships (hence, the "immediate" relationships vary with time as the closure calculation progresses).

- In graph terms, one can understand the third step of the algorithm as follows:

  For every node a_k
  For every immediate predecessor a_i of a_k
  For every immediate successor a_j of a_k,
  Make a_j an immediate successor of a_i

  So, when the algorithm processes a column k, it is adding arcs from all the current immediate predecessors of a_k to all its current immediate successors. From this observation it is obvious that the algorithm adds only arcs that should be added; i.e. if it sets m_{ij} to a value different from zero, then there is a path from the node a_i to the node a_j. Warshall proved that, when all the columns have been processed, the algorithm have added all the arcs $a \longrightarrow b$ for which there is a path from a to b.

The following example illustrates the behavior of the algorithm.

Example 2 *Let $A = \{a_1 \equiv$ `spring`$, a_2 \equiv$ `summer`$, a_3 \equiv$ `autumn`$, a_4 \equiv$ `winter`$\}$ be the syntactic domain of Example 1. Given the fuzzy relation R on A, such that $R(a_2, a_1) = 0.5$, $R(a_3, a_1) = 0.7$, $R(a_4, a_3) = 0.7$, Algorithm 1 constructs the following sequence of matrices:*

- *Input Matrix M.*

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0.5 & 0 & 0 & 0 \\ 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0 \end{bmatrix}$$

- *M after steps 1 and 2.*

$$\begin{bmatrix} \mathbf{1} & \mathbf{0.5} & \mathbf{0.7} & 0 \\ 0.5 & \mathbf{1} & 0 & 0 \\ 0.7 & 0 & \mathbf{1} & \mathbf{0.5} \\ 0 & 0 & 0.5 & \mathbf{1} \end{bmatrix}$$

- *M after iter. 1 of step 3.*

$$\begin{bmatrix} 1 & 0.5 & 0.7 & 0 \\ 0.5 & 1 & \mathbf{0.5} & 0 \\ 0.7 & \mathbf{0.5} & 1 & 0.5 \\ 0 & 0 & 0.5 & 1 \end{bmatrix}$$

- *M after iter. 2 of step 3.*

$$\begin{bmatrix} 1 & 0.5 & 0.7 & 0 \\ 0.5 & 1 & 0.5 & 0 \\ 0.7 & 0.5 & 1 & 0.5 \\ 0 & 0 & 0.5 & 1 \end{bmatrix}$$

- *M after iter. 3 of step 3.*

$$\begin{bmatrix} 1 & 0.5 & 0.7 & \mathbf{0.5} \\ 0.5 & 1 & 0.5 & \mathbf{0.5} \\ 0.7 & 0.5 & 1 & 0.5 \\ \mathbf{0.5} & \mathbf{0.5} & 0.5 & 1 \end{bmatrix}$$

- *M after iter. 4 of step 3.*

$$\begin{bmatrix} 1 & 0.5 & 0.7 & 0.5 \\ 0.5 & 1 & 0.5 & 0.5 \\ 0.7 & 0.5 & 1 & 0.5 \\ 0.5 & 0.5 & 0.5 & 1 \end{bmatrix}$$

[6] The one formed by the elements above the diagonal.

[7] Observe that this objective is not always pursued by other authors. See for instance [5].

Modifications between each iteration are boldfaced. Last matrix represents the reflexive, symmetric, transitive closure of R and, as we shall prove, a similarity relation.

This last example gives us an alternative way to understand the computation performed at Step 3 as the construction of a sequence of matrices. We shall take advantage of it in the next section.

4 Soundness of the Algorithm

Analysing Step 3 of Algorithm 1 in a more detailed way, we can observe that it is based on the construction of a sequence of matrices $W_0, W_1, \ldots, W_k, \ldots, W_n$, where W_0 is the matrix representing the reflexive, symmetric closure of a fuzzy binary relation R (obtained after steps 1 and 2 of the algorithm) and $W_k = [w_{ij}^k]$ is the one obtained in the iteration k of Step 3. By construction, if $w_{ij}^k \neq 0$ then there is a path from a_i to a_j with all the interior vertices of this path belonging to the set of nodes $\{a_1, a_2, \ldots, a_k\}$. In other words, W_k contains information of all paths that can be built from the digraph G with zero or more interior vertices in $\{a_1, a_2, \ldots, a_k\}$. It is important to note that, there is a path from a_i to a_j with all interior vertices in $\{a_1, a_2, \ldots, a_k\}$ if one of the following cases holds: i) there is a path from a_i to a_j with all interior vertices in $\{a_1, a_2, \ldots, a_{k-1}\}$; or ii) there is a path from a_i to a_k with all interior vertices in $\{a_1, a_2, \ldots, a_{k-1}\}$ and there is a path from a_k to a_j with all interior vertices in $\{a_1, a_2, \ldots, a_{k-1}\}$. The first type of path occurs if and only if $w_{ij}^{k-1} \neq 0$, and the second type of paths occurs both $w_{ik}^{k-1} \neq 0$ and $w_{kj}^{k-1} \neq 0$. Hence, $w_{ij}^k \neq 0$ if and only if either $w_{ij}^{k-1} \neq 0$ or both $w_{ik}^{k-1} \neq 0$ and $w_{kj}^{k-1} \neq 0$. Formally: $w_{ij}^k = w_{ij}^{k-1} \vee (w_{ik}^{k-1} \wedge w_{kj}^{k-1})$. Therefore, we can compute W_k directly from W_{k-1}.

Also note that when the computation of W_n has finished, one of its elements w_{ij}^n differs from zero if and only if there is a path from a_i to a_j with all interior vertices in $\{a_1, a_2, \ldots, a_n\}$. But these are the only vertices in the digraph G. This last observation proves that when Step 3 of Algorithm 1 has been concluded all the arcs $a \longrightarrow b$ for which there is a path from a to b have been added to the original digraph G, completing the computation of the transitive closure of the relation. Therefore, in the rest of this section we concentrate in those specific aspects that guarantee that Algorithm 1 is able to produce a similarity relation starting from an initial fuzzy binary relation.

Lemma 4.1 *Let A be a finite set and $\{a_1, a_2, \ldots, a_n\}$ an arbitrary sequence of its elements. Let W_0 be the reflexive, symmetric closure of a fuzzy binary relation R on A obtained after Step 1 and Step 2 of Algorithm 1. Each matrix W_k, in the sequence of matrices $W_1, \ldots, W_n$ obtained by Step 3 of Algorithm 1, fulfils the following properties for all i, j in $\{1, 2, \ldots, n\}$:*

1. *(*reflexivity*)* $w_{ii}^k = 1$;
2. *(*symmetry*)* $w_{ij}^k = w_{ji}^k$
3. *(*transitivity*)* *if* $w_{ij}^k \neq 0$, $w_{il}^k \wedge w_{lj}^k \leq w_{ij}^k$ *for each* $l \leq k$.

Before establishing the main result of this paper, it is necessary to prove the following proposition which states two interesting properties of Algorithm 1: i) once an element w_{ij}^k in a matrix W_k is set to a value different from zero it remains an invariant of the sequence of matrices $W_{k+1}, \ldots, W_n$; ii) if $w_{ij}^{k-1} = 0$ then the entry $\langle i, j \rangle$ of W_k is set to minimum of w_{ik}^{k-1} and w_{ik}^{k-1} in the iteration k.

Proposition 4.2 *Let A be a finite set and $\{a_1, a_2, \ldots, a_n\}$ an arbitrary sequence of its elements. Let W_0 be the reflexive, symmetric closure of a fuzzy binary relation R on A obtained after Step 1 and Step 2 of Algorithm 1. Let $W_1, \ldots, W_n$ be the sequence of matrices obtained by Step 3 of Algorithm 1. For each k, i and j in $\{1, 2, \ldots, n\}$:*

1. *If* $w_{ij}^{k-1} \neq 0$, $w_{ij}^k = w_{ij}^{k-1}$;
2. *If* $w_{ij}^{k-1} = 0$, $w_{ij}^k = w_{ik}^{k-1} \wedge w_{ik}^{k-1} \geq 0$.

Theorem 4.3 *Given an adjacency matrix, representing a fuzzy binary relation R on a set A, whose elements preserve transitivity and with all the elements of the superior triangu-*

lar matrix set to zero, Algorithm 1 produces the adjacency matrix corresponding to the reflexive, symmetric, transitive closure of R

We end this section with the following observation: It is possible to generalize Lemma 4.1 to an arbitrary distributive idempotent t-conorm ∇ and to an arbitrary distributive t-norm $\triangle$ (because we do not use any other specific requirement that the idempotence property of the t-conorm $\vee$; the rest of properties used in the proof are shared by all distributive t-conorms and t-norms). Therefore we can generalize Algorithm 1 to obtain a new parametric algorithm which is able to transform the initial fuzzy binary relation R on U into a reflexive, symmetric, $\triangle$-transitive fuzzy binary relation $R^{\equiv\triangle}$. That is, a similarity relation. However, $R^{\equiv\triangle}$ is not the closure of R because may happen that $R^{\equiv\triangle}(a_i, a_j) \leq R(a_i, a_j)$ for some a_i and a_j in U.

5 Refinement and Implementation Issues

Proposition 4.2 gives us some clues for the refinement of the algorithm. Certainly, it is possible to disregard entries different from zero, since they remains invariant. On the other hand, it is not necessary to inspect all the matrix entries, but only the entries of the inferior triangular matrix, because, once an element of the inferior triangular matrix is computed, the symmetric element (in the superior triangular matrix) can be set to the same value. Bearing in mind these comments, we give the following refinement of Algorithm 1.

Algorithm 2 (Step 3)
Input: *Matrix* $W_0 = [w_{ij}]$.
Output: *Matrix* W_n.
begin
`for` $k := 1$ `to` n `do`
`for` $i := 2$ `to` n `do`
`for` $j := 1$ `to` $i - 1$ `do`
`if` $w_{ij} \neq 0$
$\{w_{ij} := w_{ik} \wedge w_{kj}$;
$w_{ji} := w_{ij}\}$
end

Our algorithm has been implemented in the core of a Similarity WAM machine [8] which is the basis for the implementation of the S-Prolog compiler. The Similarity WAM machine has been designed to incorporate weak unification without altering the main structure of the classical WAM. It consist of a *Similarity Matrix* memory area, containing a representation of the adjacency matrix W_n computed by our algorithm (as well as other structures —See [8] for technical details.—). The *Similarity Matrix* memory area and its information is used at compilation time, by the Adapter (a part of the S-Prolog compiler) in order to encode the information about the similarity of some predicates into an intermediate code which is then used by the Code Generator to produce machine instruction able to manage the weak unification of predicates as a crisp process; and at execution time, when it is necessary during the argument's weak unification process.

6 Conclusions

In this paper we defined a procedure (Algorithm 1) that generates a reflexive, symmetric, closure of a fuzzy relation (and therefore a similarity relation containing the initial relation). The algorithm, has three steps. The first step computes the reflexive closure of the initial relation; the second the symmetric closure. The third step is an extension of the wellknown Warshall's algorithm for computing the transitive closure of a binary relation. We formally proved that the algorithm is sound (Theorem 4.3), in the sense that it is able to produce a similarity relation starting from an initial set of similarity equations. An interesting property of this algorithm is that once an element in a matrix is set to a value different from zero it remains an invariant of the sequence of matrices computed by the algorithm. In other words, it preserves the approximation degrees provided by the programmer in the similarity equations.

This algorithm has been implemented in the core of a Similarity WAM machine [8] where the computed similarity relation is stored into

the similarity matrix memory area.

Also it was discussed how the algorithm could be generalized to obtain a new algorithm, parametric with regard to an arbitrary distributive t-conorm, ∇, and an arbitrary distributive t-norm, $\triangle$. The new algorithm is able to transform the initial fuzzy binary relation into a reflexive, symmetric, $\triangle$-transitive fuzzy binary relation. That is, a similarity relation. However, in general, it is not the closure of the initial relation.

Acknowledgements

This work has been partially supported by the European Union FEDER project and the Spanish Science and Education Ministry under grants TIN 2004-07943-C04-03 and TIN 2007-65749.

References

[1] J. F. Baldwin, T. P. Martin, and B. W. Pilsworth. *Fril- Fuzzy and Evidential Reasoning in Artificial Intelligence*. John Wiley & Sons, Inc., 1995.

[2] Shaul Dar and Raghu Ramakrishnan. A performance study of transitive closure algorithms. In *Proc of ACM-SIGMOD*, pages 454–465, 1994.

[3] Francesca Arcelli Fontana and Ferrante Formato. Likelog: A logic programming language for flexible data retrieval. In *Proc of ACM-SAC*, pages 260–267, 1999.

[4] Francesca Arcelli Fontana and Ferrante Formato. A similarity-based resolution rule. *Int. Journal of Intelligent Systems*, 17(9):853–872, 2002.

[5] L. Garmendia, C. Campo, S. Cubillo, and A. Salvador. A method to make some fuzzy relations t-transitive. *Int. Journal of Intelligent Systems*, 14(9):873–882, 1999.

[6] L. Garmendia and A. Salvador. On a new method to t-transitivize fuzzy relations. In *Proc. of IPMU*, pages 864–869, 2000.

[7] S. Guadarrama, S. Muñoz, and C. Vaucheret. Fuzzy Prolog: A new approach using soft constraints propagation. *Fuzzy Sets and Systems*, 144(1):127–150, 2004.

[8] P. Julián-Iranzo and C. Rubio-Manzano. A WAM implementation for flexible query answering. In *Proc. of IASTED-ASC*, pages 262–267. ACTA Press, 2006. An extended version can be found at: `http://www.inf-cr.uclm.es/www/pjulian/swam.html`.

[9] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Similarity-based unification: a multi-adjoint approach. *Fuzzy Sets and Systems*, 146(1):43–62, 2004.

[10] H.T. Nguyen and E.A. Walker. *A First Course in Fuzzy Logic*. Chapman & Hall/CRC, Boca Ratón, Florida, 2000.

[11] Esko Nuutila. *Efficient Transitive Closure Computation in Large Digraphs*. PhD thesis, Helsinki University of Technology, June 1995.

[12] S. Dar R. Agrawal and H. V. Jagadish. Direct transitive closure algorithms: Design and performance evaluation. *ACM Transactions on Database Systems*, 15(3):427–458, 1990.

[13] Maria I. Sessa. Flexible querying in deductive database. In *School on Soft Computing at Salerno University: Selected Lectures 1996-1999*, pages 257–276. Springer Verlag, 2000.

[14] Maria I. Sessa. Approximate reasoning by similarity-based SLD resolution. *Theoretical Computer Science*, 275(1-2):389–426, 2002.

[15] P. Vojtas. Fuzzy Logic Programming. *Fuzzy Sets and Systems*, 124(1):361–370, 2001.

[16] S. Warshall. A theorem on boolean matrices. *Journal of ACM*, 1(9):11–12, Jan. 1962.