A Similarity-Based Unification Model for Flexible Querying

S. Krajči,[1] R. Lencses,[1] J. Medina,[2] M. Ojeda-Aciego,[2*] and P. Vojtáš[3]

[1] Inst. of Informatics. P.J. Šafárik University. Slovakia
[2] Dept. Matemática Aplicada. Univ. Málaga. Spain [†]
[3] Inst. of Computer Science. Acad. Sci. of the Czech Republic [‡]

Abstract. We use the formal model for similarity-based fuzzy unification in multi-adjoint logic programs to provide new tools for flexible querying. Our approach is based on a general framework for logic programming, which gives a formal model of fuzzy logic programming extended by fuzzy similarities and axioms of first-order logic with equality. As a source of similarities we consider different approaches, such as statistical generation of fuzzy similarities, or similarities generated by some information retrieval techniques or similarities arising from fuzzy conceptual lattices.

1 Introduction

There has been increasing interest in the development of formal tools to handle problems of users posing queries and systems producing answers. This focus has become highly relevant as the amount of information available from local and distributed information bases has increased drastically due to the expansion of the world wide web. The recent interest in search engines, and the increasing needs for adding quality in terms of flexibility, performance and precision to such engines, has further added to the importance of the topic of flexible query answering systems, the focus being to add flexibility to the systems for the storage and access to information.

It is customary to consider the following paradigm for flexible query answering: think about an expert human intermediary who is able to analyse users information needs and to evaluate the relevant information items from the available information sources. The knowledge on the information sources and the capability to interpret the user requests enable the expert to perform a good estimate of the items possibly satisfying the users needs, though the query, per se, may be *imprecise*, *incomplete*, or *vague*. Thus, one of the key issues for defining a flexible query answering system is the tolerance to imprecision and uncertainty in the formulation of user queries as well as in the representation of information. A significant effort has been made in representing imprecise information

[*] Corresponding author. `aciego@uma.es`
[†] Partially supported by Spanish DGI project BFM2000-1054-C02-02.
[‡] Partially supported by Czech Project GAČR 201/00/1489.

in database models by using fuzzy methods, and several approaches have been proposed for dealing with these problems; for instance, an access structure for similarity-based fuzzy databases is described in [8].

We are convinced that, in order to have a good approach to flexible query answering, one needs to clearly specify both the procedural and declarative parts of our systems. This is usual practice in mathematical logic, where formal models have clearly defined its syntactical part (which deals with proofs) and its semantical part (dealing with truth and/or satisfaction). When applying logic to computer science, mainly in logic programming, a different terminology is used, and we speak about the declarative part of the formal model (corresponding to truth and satisfaction) and the procedural part, more focused on algorithmic aspects of finding proofs (automated deduction).

In this paper we choose the way of including additional information about fuzzy similarities of different objects and using axioms of equality to transfer properties between these objects, our main aim being not to give practical advice on how to detect and handle similarities in practical applications, but to give a formal model for both the declarative and the procedural part of similarity-based fuzzy unification as a tool for flexible query answering.

For illustration purposes we will present a number of problems from the real world, whose solution needs an adequate treatment of similarity:

- Imagine a holidays database with names of touristic cities, depending on the language in which the database has been developed, we can find "London" in English, which in Spanish is "Londres", in Italian "Londra" and even, why not, could have been mistyped as "Lindon". Does a query using "Londres" unify with a database fact about "Londra" or "London"?
- Other problems come from the need of inter-operability, in which a client requires apparently homogeneous access to heterogeneous servers. This causes, for instance, that web users accessing these information sources usually require a multi-step process utilizing the intelligence of the end-user to navigate and to resolve heterogeneity by applying several similarity criteria [3]. There exist proposals of flexible architecture allowing users to specify a wide range of structured queries through a uniform query interface [1].
- Another example from internet: its initial design was made as an initiative for connecting sites with information stored for direct human processing, but the next generation web is aimed at storing machine processable information. For instance, implementing search engines based on ontologies to find pages with words that are syntactically different but semantically similar [2].

As a source of similarities we consider different approaches, in the next sections, such as statistical generation of fuzzy similarities, or similarities generated by some information retrieval techniques or similarities arising from fuzzy conceptual lattices. Finally, a formal model for similarity-based fuzzy unification is presented by using the multi-adjoint logic programming paradigm, and its relation with Sessa's approach [6] is stressed.

2 Similarity of Documents

Information Retrieval is an important branch of computer science which studies the representation and searching of information. Pieces of information are typically stored as a fully (or semi-)structured text in documents. When a user creates a query, an information retrieval system (IRS) finds documents whose content is relevant to user request. The similarity plays an important role in this process as we will show later. Formally, an IRS can be defined as a triple $I = (D, R, \delta)$, where D is a document collection, R is a set of queries and $\delta\colon R \to 2^D$ is a mapping assigning a set of relevant documents to each query. A widely used document representation is the vector space model (or "bag of words").

Table 1. Vector space representation of a document.

	t_1	t_2	$\dots$	t_n
d_1	w_{11}	w_{12}	$\dots$	w_{1n}
d_2	w_{21}	w_{22}	$\dots$	w_{2n}
$\dots$	$\dots$	$\dots$	$\ddots$	$\dots$
d_m	w_{m1}	w_{m2}	$\dots$	w_{mn}

Each document d_i from a document collection D is represented by a vector $(w_{i1}, \dots, w_{in})$, where w_{ij} is a measure of the importance (weight) of term t_j in document d_i. Terms t_j form a vector of terms (in the table with size n), which contains all the meaningful terms from the whole collection D. Weights can be determined by the well-established method TFIDF of document classification:

$$w_{ij} = \mathit{tf}_{ij} \cdot \log\left(\frac{M}{\mathit{df}_j}\right)$$

where tf_{ij} represents term frequency, that is, the number of occurrences of term t_j in document d_i; and the second factor is the inverse document frequency (which gives the name to the method) where M is the number of documents and df_j is document frequency, that is, the number of documents in which the term t_j occurs . The TFIDF method gives greater weight to terms which appear more frequently in lesser documents. There are also some variants of this method.

Usual similarity of documents (or documents and query) is based on the euclidean distance or cosine of angle of two vectors:

$$\mathit{Sim}(Q, d_i) = \frac{\displaystyle\sum_{j=1}^{n} w_{ij} \cdot w_{qj}}{\sqrt{\displaystyle\sum_{j=1}^{n} (w_{ij})^2 \cdot \sum_{j=1}^{n} (w_{qj})^2}}$$

where $(w_{q1}, \dots, w_{qn})$ is the vector of the query Q.

The similarity of documents is typically used in clustering of documents, which can be used for the visualization of a document collection, browsing without querying or quickly finding of documents similar to required one. The similarity of document and the query (also considered as a document) is used in finding documents relevant to the query, which is the primary purpose of IRS.

3 Similarity on Documents via Fuzzy Conceptual Lattices

There exists another interesting approach to obtain similarity of documents. Let us assume that numbers $R(d_i, t_j) = w_{ij}$ in the matrix from the Table 1 in the interval $[0, 1]$, and take some fix $\alpha \in [0, 1]$.

For every subset N of documents define the set ct_α of all common terms with degree at least α:

$$\mathrm{ct}_\alpha(N) = \{t : (\forall d \in N) R(d, t) \geq \alpha\},$$

and for every subset P of terms define the set cd_α of all common documents with degree at least α:

$$\mathrm{cd}_\alpha(P) = \{d : (\forall t \in P) R(d, t) \geq \alpha\}.$$

An α-concept is defined as a pair (N, P) of subsets of documents and terms, respectively, such that $\mathrm{ct}_\alpha(N) = P$ and $\mathrm{cd}_\alpha(P) = N$.

It is easy to see that set L_α of all α-concepts with ordering $\leq$, defined by $(N_1, P_1) \leq (N_2, P_2)$ iff $N_1 \subseteq N_2$ (or, equivalently, $P_2 \subseteq P_1$), is a lattice, so-called *fuzzy conceptual lattice.*

It is not difficult to define some type of similarity of documents in this context: Documents d_1 and d_2 are said to be α-equivalent iff there is no set of documents N in the lattice L_α which separates them, i.e. for all N, either both d_1 and d_2 belong to N or neither does. The difference of d_1 and d_2 is defined as

$$\mathrm{diff}(d_1, d_2) = \mu(\{\alpha \in [0, 1] : \ d_1 \text{ and } d_2 \text{ are not } \alpha\text{-equivalent}\})$$

where μ is Lebesgue's measure. This difference is a pseudo-metric and, of course, similarity is defined as the complement of this difference to 1.

In practice, values in the table R are usually rational (even decimal) numbers, i.e. there exists some positive integer n that all values have a form $p(d,t)/n$ for some integer $p(d, t)$. This fact allows for a simpler definition of the differences function, since for all $p < n$ and all $\alpha \in \left(p/n, (p+1)/n\right]$ all lattices L_α are identical and it follows that

$$\mathrm{diff}(d_1, d_2) = \frac{\mathrm{card}\{p : 1 \leq p \leq n \wedge (d_1 \text{ and } d_2 \text{ are not } \frac{p}{n}\text{-equivalent})\}}{n}$$

Note that roles of documents and terms can be interchanged, therefore it can be defined similarity of terms in the same way. Another approach to similarity of terms is given in the next section.

4 Similarity of Terms

To define similarity for two terms we can work either syntactically or semantically. Syntactical approaches to similarity can be based on the definition of distance functions over terms. These functions must be metrics (reflexive, symmetric and satisfy the triangle inequality). One example of such a function is *Humming distance*, which is defined as the number of positions with different characters in two terms with equal length. Another distance is the *edit* or *Levenshtein* distance. This function is defined as the minimum number of operations (insertion, deletion and substitution of one character) that are necessary to make one term equal to another.

Semantical similarity of terms is part of the relationships between terms included in a typical IR structure, such as a thesaurus. Thesauri can contain equivalences (synonyms and quasi-synonyms), hierarchical structures (hypernyma and hyponyma, meronyma) and associative relationships (other than equivalence or hierarchical relations). Another approach concerns grouping semantically similar terms (synonyms) into clusters.

Thesauri or clusters can be built by hand or automatically generated. The automatic generation of such structures can be based on the joint occurrences of terms in documents. Terms which occur simultaneously very often in documents, should have some relationship (which can be given a name). This idea can be extended with a notion of distance (terms occurring jointly should be in stronger relationship, if they their meaning is closer to each other). Another approach is based on the idea that two terms are similar if their contexts (terms in the neighbourhood) are similar (have relationship)—this is so called indirect similarity.

Similarity of terms is typically used in query expansion. A user constructs the query, then the IRS returns the relevant documents and, finally, the IRS or the user (or both) can refine the query. IRS can analyse retrieved documents (what do they have in common?) and build clusters of terms. These clusters are utilized to expand the query—either the user chooses the proper terms or the system does it automatically. A whole document collection can be used for thesaurus generation.

5 Subjective Similarities

The previous approaches correspond to object-attribute data model, where answering a query we have to fulfil some selection conditions. For instance, in a classical query

```
SELECT  Hotel
FROM    Hotels, Distance, Building
WHERE   Price < 1000, Distance < 50, Age of Building > 1995
```

we have to specify selection conditions using comparison on domains. This approach is not satisfactory because of two reasons:

1. First, it does not rank results starting with the best.
2. If there are too many or no results, we have to iterate the query by changing selection conditions: increasing or decreasing parameters $1000, 50$ and 1995, but we do not know which up which down, in what steps ...

In a fuzzy query we can specify selection conditions by fuzzy sets

```
SELECT Hotel
FROM   Hotels, Distance, Building
WHERE  Price is Cheap, Distance is Close, Age of Building is New
```

and we can order results by a (user tuned) aggregation, e.g. a weighted sum.

Here we would like to show another advantage of such an approach. Namely, we can deduce similarities on the respective domains from these fuzzy sets.

The approach has a probabilistic motivation: In probability theory there is a procedure which reasonably describes the geometry in the sample space. The likelihood distance is a "natural" pseudometric generated by the distribution function. In [7] a procedure was described for generating such a pseudometric (and hence a similarity) on an arbitrary interval. The only problem is the assumption of differentiability of the distribution.

We can understand our fuzzy sets for "close" and "cheap" as subjective probability density functions (up to some assumption on normalisation). We describe some heuristical construction which is under testing:

– Given a probability measure P with distribution function $F(x) = P((-\infty, x))$ and density $f(x) = dF(x)/dx = \mu$ (where μ is the fuzzy set) an alternative (pseudo)metric describing the geometry of the sample space is defined as
$$\rho(x_1, x_2) = |F(x_1) - F(x_2)|$$
– Another possibility is to avoid derivation and integration and assumptions on this. A similar effect can be obtained by using a monotone function T_μ generated from the density by "inverting" descending parts of the graph of the density (fuzzy set on an ordered domain, e.g. the real line or an interval).

Having a fuzzy set $\mu\colon [a, b] \longrightarrow [0, 1]$ take all local maxima $t_1, t_2, \ldots, t_i, \ldots$ and all local minima $b_1, b_2, \ldots, b_i, \ldots$ of the function μ and assume that
$$t_1 < b_1 < t_2 < b_2 < \ldots < t_i < b_i < \ldots$$
The function T_μ is defined by induction through i as follows:
$$\begin{array}{rl}
x \in (a, t_1] & \text{then } \ T_\mu(x) = \mu(x) \\
x \in (t_1, b_1] & \text{then } \ T_\mu(x) = T_\mu(t_1) + (\mu(t_1) - \mu(x)) \\
x \in (b_1, t_2] & \text{then } \ T_\mu(x) = T_\mu(b_1) - (\mu(b_1) - \mu(x)) \\
\ldots & \\
x \in (t_i, b_i] & \text{then } \ T_\mu(x) = T_\mu(t_i) + (\mu(t_i) - \mu(x)) \\
x \in (b_i, t_{i+1}] & \text{then } \ T_\mu(x) = T_\mu(b_i) - (\mu(b_i) - \mu(x))
\end{array}$$
then the pseudometric $\rho_\mu(x_1, x_2) = |T_\mu(x_1) - T_\mu(x_2)|$ generates a similarity after a normalisation.

6 Multi-Adjoint Similarity-Based Unification

We have opted to approach a querying problem (finding an information, or an object of our interest) by using logic methods defined by a suitable declarative and computation model. Here we briefly present a fuzzy similarity-based unification procedure, which is based on a theory of fuzzy logic programming with crisp unification constructed on the multi-adjoint framework introduced in [4].

We recall definitions of declarative and procedural semantics of multi-adjoint logic programming and show our model of similarity-based unification. The fact that this theory of fuzzy unification is developed inside the realm of fuzzy logic programming is very important for later integration of fuzzy similarity-based unification and fuzzy logic programming deduction.

Considering different implication operators, such as Łukasiewicz, Gödel or product implication in the same logic program, naturally leads to the allowance of several adjoint pairs in the lattice of truth-values. This idea is used in to introduce multi-adjoint logic programs, so that it is possible to use a number of different implications in the rules of our programs in a more general set of truth-values (a *multi-adjoint lattice*).

The definition of multi-adjoint logic program is given, as usual in fuzzy logic programming, as a set of weighted rules and facts of a first-order language $\mathfrak{F}$.

Definition 1. *A* multi-adjoint logic program *is a set $\mathbb{P}$ of weighted rules of the form $\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle$ such that:*

1. *The consequent of the implication, A, is an atom which is called the* head.
2. *The antecedent of the implication, $\mathcal{B}$, is called the* body*, and is a formula built from atoms $B_1, \ldots, B_n$ $(n \geq 0)$ by the use of conjunctors, disjunctors, and aggregators.*
3. *The* confidence factor *ϑ is an element (a truth-value) of L.*

Facts *and* goals *or* queries *are understood as usual. Free occurrences of variables in the program are assumed to be universally quantified.*

Definition 2.

1. *An* interpretation *is a mapping $I\colon B_{\mathbb{P}} \to L$ from the Herbrand base of $\mathbb{P}$ to the multi-adjoint lattice of truth-values $\langle L, \preceq\rangle$.*
2. *I* satisfies *a weighted rule $\langle A \leftarrow_i \mathcal{B}, \vartheta\rangle$, if and only if its extension to the whole set of formulas $\hat{I}$ satisfies $\vartheta \preceq \hat{I}(A \leftarrow_i \mathcal{B})$;*
3. *I is said to be a* model *of a program $\mathbb{P}$ if and only if all weighted rules in $\mathbb{P}$ are satisfied by I.*

Definition 3. *A pair $(\lambda; \theta)$ where $\lambda \in L$ and θ is a substitution, is a* correct answer *for a program $\mathbb{P}$ and a query $?A$ if for any model of $\mathbb{P}$ we have $\lambda \preceq \hat{I}(A\theta)$.*

As usual in logic programming, the semantics of a multi-adjoint logic program $\mathbb{P}$ is defined as the least fix-point of the immediate consequences operator $T_{\mathbb{P}}$, which is monotone and continuous (under general hypotheses); as a consequence, the least model can be reached in at most countably many iterations.

The computational model proceeds by substitution of atoms by lower bounds of their truth-value until, eventually, an extended formula with no atom is obtained, which will be interpreted in the multi-adjoint lattice to get the computed answer. Formally, given a program $\mathbb{P}$, we define the following admissible rules:

Definition 4. Admissible rules *for a pair (F, θ) where F is a formula and θ is a substitution, and A is an atom occurring in F (denoted $F[A]$), are the following:*

R1 Substitute $F[A]$ by $\big(F[A/\vartheta \mathbin{\bar{\&}_i} \mathcal{B}]\big)\theta'$, and θ by $\theta' \circ \theta$ whenever
  (a) θ' is the mgu of C and A,
  (b) there exists a rule $\langle C \leftarrow_i \mathcal{B}, \vartheta\rangle$ in $\mathbb{P}$,

R2 Substitute A by $\bot$ (just to cope with unsuccessful branches), and do not modify θ.

R3 Substitute $F[A]$ by $\big(F[A/\vartheta]\big)\theta'$ and θ by $\theta' \circ \theta$ whenever
  (a) θ' is the mgu of C and A
  (b) there exists a fact $\langle C \leftarrow_i \top, \vartheta\rangle$ in $\mathbb{P}$.

Note that if a formula turns out to have no atoms, then can be directly interpreted in the lattice. This justifies the following definition of *computed answer*:

Definition 5. *Let $\mathbb{P}$ be a program in a multi-adjoint language interpreted on a multi-adjoint lattice $\langle L, \preceq\rangle$ and let $?A$ be a goal. An element $(\dot{@}[r_1, \dots, r_m], \theta)$, with $r_j \in L$, for all $j = 1, \dots, m$ is said to be a* computed answer *if there is a sequence $G_0, \dots, G_{n+1}$ such that*

1. *$G_0 = (A, id)$ and $G_{n+1} = (\bar{@}[r_1, \dots, r_m], \theta')$ where $\theta = \theta'$ restricted to the variables of A and $r_j \in L$ for all $j = 1, \dots m$.*
2. *Every G_i, for $i = 1, \dots, n$, is a pair of a formula and a substitution.*
3. *Every G_{i+1} is inferred from G_i by one of the admissible rules.*

Assuming the necessary technical hypotheses, the following approximate completeness result was proven in [5]:

Theorem 1 (Approximate-completeness). *Given a program $\mathbb{P}$, for every correct answer $(\lambda; \theta)$ for a program $\mathbb{P}$ and a ground goal $?A$, there is a sequence of computed answers (λ_n, id) such that $\lambda \preceq \sup\{\lambda_n : n \in \mathbb{N}\}$.*

Our approach to similarity-based unification considers similarities acting on elements of domains of attributes. The idea is based on the fact that part of our knowledge base, the multi-adjoint program $\mathbb{P}$, might consist of graded facts representing information about existent similarities on different domains which depend on the predicate they are used. The particular semantics of the multi-adjoint paradigm, enables us to easily implement a version of fuzzy unification by extending suitably our given program.

Given a program $\mathbb{P}$ we construct an extension by adding a parametrized theory E (which introduces a number of similarities depending on the predicate and function symbols in $\mathbb{P}$), such as those below

$$\langle s(x,x), \top\rangle \qquad \langle s(x,y) \leftarrow s(y,x), \top\rangle \qquad \langle s(x,z) \leftarrow s(x,y) \,\&\, s(y,z), \top\rangle$$

For all function symbol we also have

$$\langle s(f(x_1,\dots,x_n), f(y_1,\dots,y_n)) \leftarrow s_1^f(x_1,y_1) \,\&\, \cdots \,\&\, s_n^f(x_n,y_n), \top\rangle$$

Finally, given a predicate symbol, then the following rules are added

$$\langle P(y_1,\dots,y_n) \leftarrow P(x_1,\dots,x_n) \,\&\, s_1^P(x_1,y_1) \,\&\, \cdots \,\&\, s_n^P(x_n,y_n), \top\rangle$$

where & is some conjunction suitably describing the situation formalized by $\mathbb{P}$.

This way we get a multi-adjoint logic program $\mathbb{P}_E$ in which it is possible to get computed answers wrt $\mathbb{P}_E$ with similarity match in unification. This justifies the introduction of a similarity-based computed answer simply as a computed answer with crisp unification on a program extended by axioms of equality.

It is worth to remark the interesting connection existing between this approach and Sessa's weak unification algorithm [6], provided we work with a particular case of the multi-adjoint framework in which $L = [0,1]$ and the only connectives will be Gödel's implication and Gödel's conjunction.

In [6], similarities are considered to act on constants, function symbols and predicate symbols, so we will assume we have a similarity relation $\mathcal{R}$ acting on $\mathcal{F} \cup \mathcal{P} \cup \mathcal{C}$ (function, predicate and constant symbols) with truth-values ranging in the unit real interval $[0,1]$. In our approach, the similarity $\mathcal{R}$ is internalized, that is, we include a new predicate symbol in our language, denoted $s_{\mathcal{R}}$.

Now, for every $f, g \in \mathcal{F}$ with $\mathcal{R}(f,g) > 0$ let us extend our logic program by the following schema of axioms

$$\langle s_{\mathcal{R}}(f(x_1,\dots,x_n), g(y_1,\dots,y_n)) \leftarrow_G s_{\mathcal{R}}(x_1,y_1) \,\&_G\, \cdots \,\&_G\, s_{\mathcal{R}}(x_n,y_n), \mathcal{R}(f,g)\rangle$$

which, in the case of constants it is understood as $\langle s_{\mathcal{R}}(c,d), \mathcal{R}(c,d)\rangle$.

In addition, for every $P, Q \in \mathcal{P}$ with $\mathcal{R}(P,Q) > 0$ let us extend our logic program by a schema of axioms

$$\langle P(y_1,\dots,y_n) \leftarrow_G Q(x_1,\dots,x_n) \,\&_G\, s_{\mathcal{R}}(x_1,y_1) \,\&_G\, \cdots \,\&_G\, s_{\mathcal{R}}(x_n,y_n), \mathcal{R}(P,Q)\rangle$$

With this notation, it is possible to show that Sessa's approach to unification can be embedded in ours.

Theorem 2 ([5]). *Let $P(t_1,\dots,t_n)$ and $Q(t'_1,\dots,t'_n)$ be two atoms, assume that some substitution θ is a λ-unifier (for $\lambda \in [0,1]$) obtained by the weak unification algorithm, then (λ,θ) is a computed answer to the query $?P(t_1,\dots,t_n)$ wrt the program $E \cup \mathcal{R} \cup \{\langle Q(t'_1,\dots,t'_n), 1\rangle\}$.*

It is remarkable that the equality axioms introduced in order to obtain $\mathbb{P}_E$ are simply the similarity-based extension obtained when the 'external' similarity $\mathcal{R}$ is the usual equality relation.

Furthermore Sessa's similarity-based SLD derivation can be completely emulated by our multi-adjoint computation, since it is applied on a classical program, and the only place where uncertainty appears is in the similarity coming from unification. Now, as a consequence of the theorem on emulation of unification by the computational model of multi-adjoint programs, the following theorem holds, in which we are assuming the language of [6] (Defn.7.2).

Theorem 3. *Given a similarity $\mathcal{R}$, a crisp program $\mathbb{P}$ and a goal G_0, and a similarity-based derivation $G_0 \Longrightarrow_{C_1,\theta_1,\lambda_1} G_1 \Longrightarrow \cdots \Longrightarrow_{C_m,\theta_m,\lambda_m} G_m$ the approximation degree of $\theta_1 \cdots \theta_n$ restricted to the variables of G_0 is set to be $\lambda = \min_{1\le i\le m}\{\lambda_i\}$, then there exists a multi-adjoint computation for $?G_0$ and the (crisp) program $\mathbb{P}$ in the logic with Gödel connectives and $L = [0, 1]$ such that the computed answer is (λ, θ).*

To finish with, it is important to note that all theorems on fix-point, H_λ and $\mathcal{P}_\lambda$ semantics given in [6] state that Sessa's approach perfectly embeds in this more general multi-adjoint approach suitable restricted to the unit interval and Gödel connectives.

7 Conclusions

Different approaches have been introduced to generate similarities to be used in flexible query answering systems, such as statistical generation of fuzzy similarities, or generation by some information retrieval techniques or similarities arising from fuzzy conceptual lattices. Later, a formal model for similarity-based fuzzy unification is presented by using the multi-adjoint logic programming paradigm to provide new tools for flexible querying. The approach gives a formal model of fuzzy logic programming extended by fuzzy similarities and axioms of first-order logic with equality. Finally, it is shown how the given framework perfectly emulates Sessa's approach to similarity-based unification.

References

1. S. Adali and C. Bufi. A flexible architecture for query integration and mapping. In *Cooperative Information Systems Conference*, 1998.
2. S. Decker, S. Melnik, F. van Harmelen, D. Fensel, M. Klein, J. Broekstra, M. Erdmann, and I. Horrocks. The semantic web: the roles of XML and RDF. *IEEE Internet Computing*, 43:2–13, 2000.
3. K.G. Jeffery. What's next in database. *ERCIM News*, 39:24–26, 1999.
4. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. A procedural semantics for multi-adjoint logic programming. In *Progress in Artificial Intelligence, EPIA'01*, pages 290–297. Lect. Notes in Artificial Intelligence 2258, 2001.
5. J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Similarity-based unification: a multi-adjoint approach. *Fuzzy Sets and Systems*, 2002. Submitted.
6. M.I. Sessa. Approximate reasoning by similarity-based SLD resolution. *Theoretical Computer Science*, 275(1–2):389–426, 2002.
7. P. Vojtáš and Z. Fabián. Aggregating similar witnesses for flexible query answering. In H.L. Larsen et al., editor, *Flexible Query Answering Systems, FQAS'00*, pages 220–229. Physica Verlag, 2000.
8. A. Yazici and D. Cibiceli. An access structure for similarity-based fuzzy databases. *Information Sciences*, 115(1–4):137–163, 1999.