# Similarity-based SLD Resolution and applications to Mobile Agents

Luciano Blandi, Vincenzo Loia, Sabrina Senatore, and Maria I. Sessa

Dipartimento Matematica e Informatica
Universita' di Salerno
S. Allende - 84081 Baronissi (SA), Italy
{lblandi, loia, senatore, misessa}@unisa.it

**Abstract.** This work presents an extension of SLD resolution towards approximate reasoning. The proposed refutation procedure overcomes failures in the unification process by exploiting Similarity relation defined between predicate and constant symbols. This enables to compute approximate solutions, with an associated approximation degree, when failures of the exact inference process occur. In this paper we outline the main ideas of this approach. Moreover, a survey of applications to Mobile Agents based systems is also presented.

## 1 Introduction and previous works

In the literature, approximate reasoning capabilities are introduced in the Logic Programming framework [1] by considering the inference system based on fuzzy logic rather than on conventional two-valued logic. Several PROLOG interpreters based on fuzzy logic have been presented [2] [3] [4]. The basic idea is that some facts have associated truth value in [0,1] representing their degree of membership in the set of true assertions. The truth value of a rule is computed by the truth values of its conditions, according to a given combination operator. In [5] a new methodology that allows us to enhance the Logic Programming paradigm with approximate reasoning capability has been introduced by exploiting a different approach. The basic idea is that the fuzziness feature is provided by an abstraction process which exploits Similarity relations between elements in the alphabet of the language (constants, functions, predicates). This approach enables to avoid both the introduction of weights on the clauses, and the use of fuzzy sets as elements of the language. In [6] the operational counterpart of this extension is faced by introducing a modified SLD Resolution procedure. Such a procedure allows us to compute numeric values belonging to the interval [0,1] providing an approximation measure of the obtained solutions. These numeric values are computed through a generalized unification mechanism. In [11] a Prolog interpreter written in Java which implements this Similarity-based extension has been presented. In this paper we outline these results and also provide a survey of applications to the design of Mobile Agents based system.

## 2 A formal view to Similarity

The mathematical notion of *Similarity relation* is a many valued extension of the equality, and it is widely exploited in any context where a weakening of the equality constraint is useful. We summarize some results concerning this notion. At first, let us recall that a *T-norm* is a binary operation $\wedge : [0,1] \times [0,1] \to [0,1]$ associative, commutative, non-decreasing in both the variables, and such that $x \wedge 1 = 1 \wedge x = x$ for any $x$ in [0,1]. In the sequel, we assume that $x \wedge y$ is the *minimum* between the two elements $x, y \in [0,1]$.

**Definition 1.** *A* similarity *on a domain $\mathcal{U}$ is a fuzzy relation $\mathcal{R} : \mathcal{U} \times \mathcal{U} \to [0,1]$ in $\mathcal{U}$ such that the following properties hold*

*i)* $\mathcal{R}(x,x) = 1$ *for any* $x \in \mathcal{U}$ *(reflexivity)*

*ii)* $\mathcal{R}(x,y) = \mathcal{R}(y,x)$ *for any* $x, y \in \mathcal{U}$ *(symmetry)*

*iii)* $\mathcal{R}(x,z) \geq \mathcal{R}(x,y) \wedge \mathcal{R}(y,z)$ *for any* $x, y, z \in \mathcal{U}$ *(transitivity).*

Similarity relations are strictly related to equivalence relations and, then, to closure operators.

**Proposition 1.** *Let $\mathcal{U}$ be a domain and $\mathcal{R} : \mathcal{U} \times \mathcal{U} \to [0,1]$ a Similarity in $\mathcal{U}$. Then, for any $\lambda \in [0,1]$, the relation $\cong_{\mathcal{R},\lambda}$ in $\mathcal{U}$, named* cut of level $\lambda$ *(in short $\lambda$-cut) of $\mathcal{R}$, defined as*

$$x \cong_{\mathcal{R},\lambda} y \iff \mathcal{R}(x,y) \geq \lambda$$

*is an equivalence relation. Also, the operator $H_{\cong_{\mathcal{R},\lambda}} : \mathcal{P}(\mathcal{U}) \to \mathcal{P}(\mathcal{U})$ such that for any $X \in \mathcal{P}(\mathcal{U})$*

$$H_{\cong_{\mathcal{R},\lambda}}(X) = \{z \in \mathcal{U} \mid \exists x \in X : x \cong_{\mathcal{R},\lambda} y\} = \{z \in \mathcal{U} \mid \exists x \in X : \mathcal{R}(z,x) \geq \lambda\},$$

*is a closure operator.*

## 3 Logic Programming with Similarity

We briefly recall that a logic program $P$ is a set of universally quantified Horn clauses on a first order language $L$, denoted with $H \leftarrow B_1, \ldots, B_k$, and a goal is a negative clause, denoted with $A_1, \ldots, A_n$. We denote with $B_L$ the set of ground atomic formulae in $L$, i.e. the Herbrand base of $L$, and with $T_P$ the immediate consequence operator $T_P : P(B_L) \mapsto P(B_L)$ defined by:

$$T_P(X) = \{a | a \leftarrow a_1, \ldots, a_n \in \Gamma(P) \text{ and } a_i \in X, 1 \leq i \leq n\}$$

where $\Gamma(P)$ denotes the set of all ground instances of clauses in $P$. The application of Tarski's fixpoint theorem yields a characterization of the semantics of $P$, which is the least Herbrand model $M_P$ of $P$ given by:

$$M_P = lfp(T_P) = \bigcup_{n \geq 0} T_P^n(\emptyset)$$

where lfp stands for least fixpoint [1].

In the classical Logic Programming, function and predicate symbols of the language $L$ are crisp elements, i.e., distinct elements represent distinct information and no matching is possible. In [5] the exact matching between different entities is relaxed by introducing a Similarity relation R in the set of constant, function and predicate symbols in the language of a logic program P. In order to deal with the approximation introduced by a similarity relation $\mathcal{R}$, the program $P$ is extended by adding new clauses which are "similar" at least with a fixed degree $\lambda$ in (0,1] to the given ones. This program transformation is obtained by considering the closure operator $H_\lambda$ associated to $\mathcal{R}$. The new logic program

$$H_\lambda(\Gamma(P)) = \{C' \in L | \exists C \in \Gamma(P) \text{ such that } \mathcal{R}(C, C') \geq \lambda\},$$

named *extended-program of level* $\lambda$, allows us to enhance the inference process.

An alternative way to manage the information carried on by the Similarity introduced between function and predicate symbols in $P$, can be given by considering as an unique element different symbols which have Similarity degree greater or equal to $\lambda$. In other words, we consider the quotient set of $\cong_{\mathcal{R},\lambda}$ as a new alphabet $L_\lambda$, where $F/\cong_{\mathcal{R},\lambda}$ and $R/\cong_{\mathcal{R},\lambda}$ are the sets of function and predicate symbols, respectively. More formally, let us denote with $[s] \in L_\lambda$ the equivalence class of a symbol $s \in F \cup R$ with respect to $\cong_{\mathcal{R},\lambda}$We call *translation up* to $\cong_{\mathcal{R},\lambda}$ the function:

$$\tau_\lambda : F \cup R \mapsto F/\cong_{\mathcal{R},\lambda} \cup R/\cong_{\mathcal{R},\lambda}$$

defined by setting:

$$\tau_\lambda(x) = x \text{ for any variable } x \in V, \text{ and } \tau_\lambda(f) = [f]$$

for any function/predicate symbol $f \in F \cup R$.

Recursively, we can easily define the extension of $\tau_\lambda$ to the sets of formulae in $L$. Let us consider a logic program $P$ on the language $L$.

The set

$$P_\lambda = \tau_\lambda(\Gamma(P)) = \{C' \in L_\lambda | C' = \tau_\lambda(C),\ C \ \textit{clause in}\ \Gamma(P)\}$$

is a logic program that we name *abstract-program* of level $\lambda$.

By considering the abstract program $P_\lambda$, it is possible to express information provided by the similarity relation in a syntectic way exploiting the quotient language $L_\lambda$. Then, $P_\lambda$ could be used to manage similarity-based reasoning as well as $H_\lambda(\Gamma(P))$. In [10] it has been shown the equivalence of these two approaches by using an abstract interpretation technique as follows:

**Proposition 2.** *Let $P$ be a program on a first order language $L$, $\mathcal{R}$ a similarity in $L$ and $\tau_\lambda$ the related translation up to $\cong_{\mathcal{R},\lambda}$Then, the functions*
*$\alpha : \mathcal{P}(B_{L_\lambda}) \mapsto \mathcal{P}(B_{L_\lambda})$ and $\gamma : \mathcal{P}(B_{L_\lambda}) \mapsto \mathcal{P}(B_L)$, such that $\forall X \in \mathcal{P}(B_L)$,*
*$\forall Y \in \mathcal{P}(B_{L_\lambda})$, $\alpha(X) = \tau_\lambda(X)$ and $\gamma(Y) = \tau_\lambda^{-1}(Y)$ provide a Galois surjection*

$$(\mathcal{P}(B_L), \subseteq) \underset{\alpha}{\overset{\gamma}{\rightleftharpoons}} (\mathcal{P}(B_{L_\lambda}), \subseteq)$$

*between the complete lattices* $(\mathcal{P}(B_L), \subseteq)$ *and* $(\mathcal{P}(B_{L_\lambda}), \subseteq)$, *such that* $\gamma\alpha = H_{\cong_{\mathcal{R},\lambda}}$. *Moreover, the abstract semantics is* $\alpha$*-optimal with respect to the immediate consequence operators* $T_{H_\lambda(\Gamma(P))}$ *and* $T_{P_\lambda}$, *i.e.,* $\alpha(T_{H_\lambda(\Gamma(P))}(X)) = T_{P_\lambda}(\alpha(X))$ *and* $\alpha(M_{H_\lambda(\Gamma(P))}) = T_{P_\lambda}$.

In [5], the formal notion of *fuzzy least Herbrand model* $M_{P,R} : B_L \mapsto [0,1]$ of the program $P$ with respect to the Similarity $\mathcal{R}$ is defined by setting for any $A \in B_L$:

$$\begin{aligned} M_{P,R}(A) &= Sup\{\lambda \in [0,1] \,|A \in M_{H_\lambda(\Gamma(P))}\} \\ &= Sup\{\lambda \in [0,1] \,|H_\lambda(\Gamma(P)) \vDash A \ \} \end{aligned}$$

Roughly speaking, for any $A \in B_L$ the value $M_{P,R}(A)$ provides the best deduction degree of $A$, i.e, the best level of approximation $\lambda$ that allows us to prove $A$ by considering an extended program $H_\lambda(\Gamma(P))$. Exploiting the $\alpha$-optimality of the Galois connection, it can be proved that:

$$\begin{aligned} M_{P,R}(A) &= Sup\{\lambda \in [0,1] \,|t_\lambda(A) \in M_{P_\lambda}\} \\ &= Sup\{\lambda \in [0,1] \,|P_\lambda \vDash \tau_\lambda(A)\} \end{aligned}$$

Thus, in order to compute the fuzzy least Herbrand model of a program $P$ extended with a Similarity $\mathcal{R}$, we can equivalently perform our computations in the extended or in the abstract domain.

## 4 Similarity-based SLD Resolution

SLD Resolution in the extended and abstract program needs some preprocessing steps in order to transform the given program $P$. Thus, in [6] a new modified version of SLD Resolution, named Similarity-based SLD, which allows us to perform these kinds of extended computations exploiting the original program $P$, without any preprocessing steps, has been introduced. The failure of the unification between different function or predicate symbols is avoided by relaxing the equality constraint with the Similarity relation. It leads to the notion of unification-degree associated to a substitution, and a weak most general unifier (in short weak m.g.u.) is a more general substitution which provides the best unification-degree. These notions have been introduced in [7]. In [6] the following generalized unification algorithm based on Similarity has been proposed.

### WEAK-UNIFICATION ALGORITHM

Given two atoms $A = p(s_1, \ldots, s_n)$ and $B = q(t_1, \ldots, t_n)$ of the same arity with no common variables to be unified, construct the associated set of equation $W = \{p = q, s_1 = t_1, \ldots, s_n = t_n\}$. If $\mathcal{R}(p,q) = 0$, halts with failure, otherwise, set $U = \mathcal{R}(p,q)$ and $W = W - \{p = q\}$. Until the current set of equation $W$ does not change, non deterministically choose from $W$ an equation of a form below and perform the associated action.

1. $f(s_1, \ldots, s_n) = g(t_1, \ldots, t_n)$ where $\mathcal{R}(f, g) > 0$: replace by the equations $s_1 = t_1, \ldots, s_n = t_n$, and set $U = U \wedge \mathcal{R}(f, g)$;
2. $f(s_1, \ldots, s_n) = g(t_1, \ldots, t_m)$ where $\mathcal{R}(f, g) = 0$: halts with failure;
3. $x = x$: delete the equation;
4. $t = x$ where $t$ is not a variable; replace by the equation $x = t$;
5. $x = t$ where $x \neq t$ and $x$ has another occurrence in the set of equations: if $x$ appears in $t$ then halt with failure, otherwise perform the substitution $\{x/t\}$ in every other equations.

Exploiting weak m.g.u. we can overcome failures of the exact matching between function or predicate symbols if they are related by a non zero Similarity value.

As an example, let us consider the atoms $p(a)$ and $q(x)$, which are not unifiable, and a Similarity $\mathcal{R}$ such that $\mathcal{R}(p, q) = .7$ and $\mathcal{R}(a, b) = .5$. By assuming that the Similarity $\mathcal{R}$ replaces the equality relation, we have that the instances obtained by applying $\xi_1 = \{x/a\}$ and $\xi_2 = \{x/b\}$ given by

$$p(a)\xi_1 = p(a) \neq q(a) = q(x)\xi_1 \text{ and } p(a)\xi_2 = p(a) \neq q(b) = q(x)\xi_2$$

can be considered "'equal"', but some "'tolerance"' must be exploited to overcome the mismatch between the predicate symbols $p$ and $q$ and the constant symbols $a$ and $b$. In a straight way, a measure of this "'tolerance"' level can be expressed for $\xi_1$ by the similarity value $\mathcal{R}(p, q) = .7$, and for $\xi_2$ by $\mathcal{R}(p, q) \wedge \mathcal{R}(a, b) = .7 \wedge .5 = .5$ (i.e., the minimum between the similarity values which relate the mismatching symbols). Then, $\xi_1$ provides an "'unification degree"' better than $\xi_2$.

It is intuitive that an higher value of Similarity exploited to overcome failures of matching, corresponds to a better value of "tolerance" which is needed to consider "equal" different symbols. Thus, it is natural to assume that an acceptable unifier must provide the maximum value of Similarity between the obtained instances.

In general, a computed answer substitution can be obtained with different SLD refutations and different approximation-degrees, then the maximum of this values characterizes the best refutations of the goal. In particular, a refutation with approximation-degree $\lambda = 1$ provides an exact solution. In [6] it is proved that Similarity based SLD Resolution performed with respect to P and the SLD Resolution performed with respect to the extended and abstract programs $H_\lambda(\Gamma(P))$ and $P_\lambda$, are equivalent by the computational point of view. Moreover, for any $A \in B_L$ the membership value $M_{P,R}(A)$ of the fuzzy least Herbrand model is given by the best approximation-degree of the refutations in the Similarity-based SLD tree for $P \cup \{\leftarrow A\}$.

# 5 Applications

## 5.1 Similarity-based Agent

Agent technology characterizes the current state of software design in the area of distributed systems and applications [12]. Even though there is no universal definition accepted for the term "'agent"', the scientific community sees an agent as a software component equipped by a set of characteristics such as: autonomy, persistence, social ability, reactivity, pro-activeness. These features are important especially to handle real-world problems that are characterized by a wide space of solutions. In this case, the difficulty (or often, the impossibility) to provide a complete, precise model of the problem is faced thanks to the agent's ability to tolerate and process approximate or vague information.

A mobile agent is a software component that is able to migrate between different locations in a network, in order to execute its tasks by oneself or with a collaboration/interaction with other (mobile or not) agents.

There is substantial difference between mobile agents and a simple traditional mobile code; this difference is described by two kinds of mobility called, respectively, migration and remote execution. Remote execution, often called mobile code, is a program sent to a remote location and there it is activated; during its whole execution, it remains at current location. Migration, instead, realizes a passage to another location during its execution, that means a mobile agent starts its execution, for example, at current location, migrates to another location and there continues its execution, exactly at the point at which it has been interrupted before the migration.

Our applications are based on the development migration of specific, focus-oriented agents, called Similarity-based Agents.

A Similarity-based Agent is a mobile agent equipped with a basic (initial) knowledge and with an extended Prolog interpreter [11], named SiLog, which implements the Similarity based SLD Resolution. It also supports a user-friendly interface that allows us to define the Similarity relation. The extension has been realized considering Java as implementation tool. Thanks to Silog the agent may overcome situations of failure in the matching between entities involved in the information processing; in this way an approximate answer to a question can be obtained, when the exact one is not possible. The agent in matter is able to move on the net, to reach a site, to inspect it, to interpret the founded local documents, to generate new information, to represent them as Prolog facts and to deduce some evaluations of interest for the purpose assigned him. After that, if a maximum level of depth of spidering is considered and if this has not been reached, the agent leaves the current Web position in order to explore new URL addresses where the information acquisition process goes on.

We now give a survey on the applications that use this type of agents. They can be seen as Web assistance systems, since they try to get information of interest for the user, without forcing the user himself to have involved in the activity of search.

## 5.2 Discovery E-Mail System (D-Mail)

**Goal**

E-mail is nowadays the fastest and cheapest form of communication. From its initial use restricted to message exchange between colleagues or friends, E-mail is currently the worldwide medium into business sectors. With Internet use continuing to explode, and due to the simplicity of E-mail sending to many people, recent years have seen the time spent in dealing with unnecessary and irrelevant E-mails to reach an alarming rate. Then, the same virtues that made E-mail so popular are now becoming a negative technologic boomerang (see the volume of junk or spam mail). A strong aid to this problem is offered by smart filtering services, which distinguish, at the receiver-side, legitimate E-mail from spamming. The approach herein described is different: our mail system is skilled to find 'appropriate' destinations for a sending message, allowing so to the user to send an E-mail to appropriate but unknown readers. The (discovered) readers are users whose 'profile' is compatible with the E-mail message. The profile is described in terms of arguments which are connected to the E-mail topic with some degree of 'similarity'.

**D-Mail's workflow**

D-Mail [9] provides a general client-side application, as shown in Fig. 1.

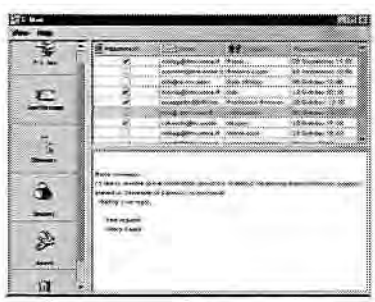

**Fig. 1.** D-Mail interface

It appears as a typical window of a common e-mail environment, splitted in zones. On the left side there is a set of buttons for rapid commands; the right side is divided in two sub-panels: the upper one maintains incoming E-mail history list, whereas the bottom sub-window is used to show the e-mail text related to selected message on the mailing list.

It supervises and manages the similarity agents which are sent to workstations in the net. In the "normal" mode it guarantees usual services of sending and receiving e-mails.

It is not easy to define a similarity in a set of elements by providing the similarity values $\mathcal{R}(x, y)$ for any $x, y \in U$. Indeed, the transitivity constraint in Definition 1 can produce side effects which can contradict the assigned similarity values.

The D-Mail system provides a user-friendly interface to help the user to define a new similarity relation on a loaded dictionary, as shown in Fig. 2.

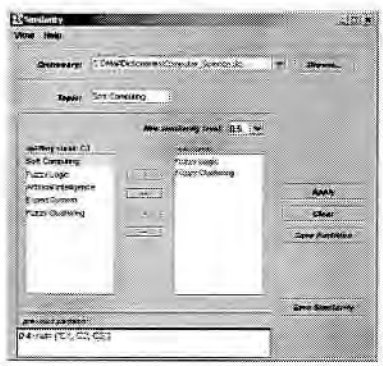

**Fig. 2.** Similarity panel

The system provides a default set of similarity values $\lambda_i$ given by

$$\lambda_0 = 0, \qquad \lambda_{i+1} = \lambda_i + 0.1 \qquad 1 \leq i \leq 10$$

When this activity is completed, the user sets the parameter related to the similarity-based agents. Besides, the user has to specify a threshold value, so that solutions with approximation degrees below this threshold are automatically discarded by the agents. This information is introduced in the system through the panel of Fig. 3.

**Fig. 3.** Setting of Agent's properties

The parameter depth level spidering defines the "'pervasivity"' of search in terms of sub-domains reacheable from the input Web sites starting domains).

The activation of the Start Activity button triggers the spidering process. Then, once setting these parameters, a number of autonomous agents are created which, through a spidering process on a portion of the Web, analyze documents (HTML-based pages) on web sites of potential receivers. Any agent extracts knowledge from the Web document, represented as ground facts of a classical Prolog program. Then, these facts and the Prolog goal are processed by similarity-based engine, inside the agent itself . In case of success, the agent returns to the home station the e-mail of the discovered potential readers together with the value $\lambda$ representing their degree of interest about the given topic. A collector agent on the client-side, gathers all these responses, ranks and presents them as a mailing list. Now, the user can write the mail message in a text area and sent it to the users shown in the ranked list.

### 5.3 Masir architecture's overview

**Goal**

The uncontrolled growth and the dissemination of information on the Internet is very hard to manage: information is difficult to reach, due to the lack of direct links; documents are logically related to others but the relationship has gone unnoticed, or some documents have unlinked parts of information, the examination of which would produce new knowledge. One of the main difficulty is to return accurate information that fully describes the human request: the user is often able to give an exhaustive explanation about the content of his request; the human expressiveness is complex, the same phrase can be customized in different manners, and the meaning may be vague. Standard Web search engines provide two mechanisms to handle user's query: to input the search keywords or to restrict the searching within a certain topic. These approaches appear too weak to cope with a so vastity of information and with a so rich human expressiveness.

Our model of distributed IR, namely Masir [8], standing for Mobile Agents for Similarity-based Information Retrieval, provides a wide range of information retrieval services by exploiting mobile computation of similarity-based agents.

**Distributed Information Retrieval in Masir**

Masir is an architecture mostly implemented in Java and running on top of TCP/IP derived protocols. Usually, traditional information retrieval strategies return myriad of independent, disaggregated solutions. Masir works in order to return, through a further deductive process, the final results free from redundancy and inconsistence. Fig. 4 shows the logical components of Masir environment.

It consists, essentially, of different types of task-oriented agents, which work together to reach a specific goal. The Masir platform works in background (transparent mode) on the user's machine. There are appropriate modules that allow the user to design (or load) the current dictionary and the similarity relation. Similarity Definition appears as a user-side interface tool: it allows the user to

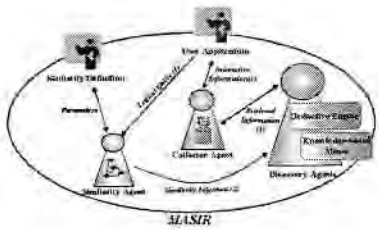


**Fig. 4.** Masir- Logical Architecture

acquire the defined Similarity relationships upon a suitable fixed set of terms (the universe or dictionary) in order to assure the correct Similarity's building. Then, the Similarity Agent collects the similarity relations defined upon the input dictionaries and manages the dictionary-based operation involved by users.

When the user wants to interact with the system, triggers a User Application: the corresponding environment is activated and the human interaction is viewed in terms of a query construction. Each User Application exhibits an user interface designed to transform the user's request into a logic goal to be proven with a related knowledge-base. Different IR-oriented services are available in Masir; in the following we will show InfoMiner an application able to search scientific papers on the Web. The query and the current Similarity are hence injected into the Discovery Agents, becoming part of its knowledge. The Discovery Agent, with this baggage, moves on the net to reach the destination site, parses the web documents, extracts the inherent knowledge in terms of Prolog facts and through the similarity-based reasoning (SiLog) returns, as result, answers to the input query. The obtained answers have an associated relevance degree of the returned information for a given request that expresses a measure of the weakening of the equality constraint. While it examines the local page, it clones itself, every time that an external link is founded (that allows the agent to reach another machine). The collection of the Discovery Agents' results is gathered by the Collector Agent, active on the user's machine. Its ability in filtering enables to obtain a synthetic and error-pruned answer. This information is then sent to the User Application, in order to customize the final exploitation.

### Info-Miner: Example of user application

Fig. 5 shows the Info-Miner query interface. It appears as a typical interface with several text fields, to be filled by the user during the query formulation.

The information required by the user concerns scientific documents, and is defined in terms of correlated fields (authors, proceedings or journal contribution, publication date, topic(s), ... ). At query mode, the user specifies the search mode, as the web domain to be analyzed and the depth level searching (a parameter that limits the visiting of possible Web sites). In our example of Fig. 5, the user

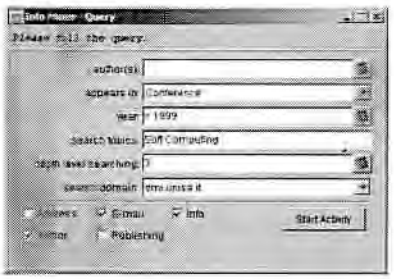


**Fig. 5.** Web search interface

looks for papers appeared in a Conference after the year 1999, dealing with the topic Soft Computing.

The query is transformed into the following Prolog-like goal:

```
? -   doc(IdPage, IdDoc, Author, Title), topic(IdDoc,
      ['Soft Computing']), doc-reference(IdDoc, 'Conference'),
      after(IdDoc, Date, 1999), ..., photo(IdPage, Author, File).
```

The Discovery Agent dispatched on the specified Web area (in the example our department site), tries to prove the previous goal, using the facts extracted from the "contacted" Web page and its knowledge-base of Prolog-like rules. These facts are processed by the Similarity-based Resolution, taking into account the similarity relations established by the user. To better clarify the extended resolution procedure, let us suppose the user has defined a similarity degree l=0.7 between the term Soft Computing and the term Fuzzy and that, in the proof of the main goal, the variable IdDoc has as value 'idDoc25'. During the SLD resolution process, the sub-goal topic(IdDoc, ['Soft Computing']) has to be proved. The knowledge-base used in the proof consists of a number of logic predicates, among them the ones used to treat the subgoal topic. Let us consider the predicate topic:

```
topic(IdDoc, Arg) :- doc(IdPage, IdDoc,Author,Title),
                     is_in(Arg, Title).
```

This rule succeeds when in the title of the document (logic variable Title) appears the term Arg. The unification of terms in the list ['Genetic-based', 'Fuzzy', 'Clustering', ... ] (Title) and 'Soft Computing' (Arg), does not fail, as in the case of "'standard"' SLD Resolution, thanks to an existing similarity relationship. Thus the subgoal topic(IdDoc, 'Soft Computing') succeeds with approximation degree 0.7. The results are finally returned to the Collector Agent, charged to fuse the different replies in an efficient visualization. The system displays the authors' photos (if available) and other information related to the found Web page, according to the user's requirements. The relevance of the results is expressed through the Similarity value computed for the specific Web page.

## 6 Conclusions

The approach to the approximate reasoning herein followed exploits formal tools belonging to distinct fields. On one hand, the classical declarative paradigm of the Logic Programming provides the inference system. On the other hand, the weakening of the equality notion is managed by means of the fuzzy Similarity relation. The work herein presented is part of a larger research project where different technologies, in particular mobile computing, approximate reasoning, agents, are combined to design advanced systems for Web searching. In order to provide a robust approach to treat the difference between the information available and the information necessary to the agent to make the best decision, we have embedded into the agents the Similarity-based Reasoning as key issue for a flexible interpretation of information. As first two prototypes we report the positive evaluation both of a system (D-Mail) able to detect the right subset of potential unknown readers to whom an email message can be targeted and of a system (Masir) that supplies a wide variety of services of Information Retrieval which, as in the InfoMiner application, the searching of scientific papers, conferences, scientists, and so on.

## References


1. Apt R.K., *Logic Programming*, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, vol. B, (Elsevier, Amsterdam, 1990) 492-574.
2. Ishizuka M., Kanai N., *PROLOG-Elf incorporating fuzzy logic*, Proc. 9th Int. Joint. Conf. on Artificial Intelligence (Springer, Berlin, 1985) 701-703.
3. Mukaidono M., Shen Z.L., Ding L., *Fundamentals of fuzzy PROLOG*, Int. Journal of Approximate Reasoning 3 (1989) 179-193.
4. Martin T.P., Baldwin J.F., Pilsworth B.W., *The implementation of FPROLOG a fuzzy PROLOG interpreter*, Fuzzy Sets and Systems 23 (1987) 119-129.
5. Gerla G., Sessa M.I., *Similarity in Logic Programming*, in: G. Chen, M. Ying, K.-Y. Cai (Ed.s), Fuzzy Logic and Soft Computing, (Kluwer Acc. Pub., Norwell, 1999), 19-31.
6. Sessa M.I., *Approximate Reasoning by Similarity-based SLD Resolution*, Theoretical Computer Science, 275 (2002) 389-426.
7. Formato F., Gerla G., Sessa M.I., *Similarity-based unification*, Fundamenta Informaticae 40 (2000) 1-22.
8. Loia V., Luongo P., Senatore S. and Sessa M.I., *A Similarity-based View to Distributed Information Retrieval with Mobile Agents*. In The 10th IEInternational Conference on Fuzzy Systems, Melbourne, Australia, December 2-5, 2001.
9. Loia V., Senatore S. and Sessa M.I., *Similarity-based agents for email mining*. In Proc. of the joint Conference IFSA/NNAFIPS 2001, Vancouver, Canada, July 25-28, 2001.
10. Sessa M.I., *Translations and Similarity-based Logic Programming*, Soft Computing 5(2) (2001).
11. Loia V., Senatore S. and Sessa M.I., *Similarity-based SLD Resolution and its implementation in an Extended Prolog System*, Fuzzy-Set and System, in press.
12. Genesereth M.R., Ketchpel S.P. *Software agents*. Communications of the ACM, 37(7):48-53, 1994.