Computing and Informatics, Vol. 26, 2007, 63–76

COMPUTED ANSWER FROM UNCERTAIN KNOWLEDGE: A MODEL FOR HANDLING UNCERTAIN INFORMATION

Ágnes Achs

Department of Computer Science
Faculty of Engineering, University of Pécs
Boszorkány u. 2
7622 Pécs, Hungary
e-mail: achs@witch.pmmf.hu

Manuscript received 13 April 2006; revised 24 October 2006
Communicated by Irina Ezhkova

Abstract. In this work we present a model for handling uncertain information. The concept of fuzzy knowledge-base is defined as a quadruple of background knowledge. Specifically, the latter is defined by the proximity of predicates and terms; a deduction mechanism: a fuzzy Datalog program; a connecting algorithm, which connects the background knowledge with the program, and a decoding set of the program which helps us determine the uncertainty level of the results. We also suggest a possible evaluation strategy.

Keywords: Uncertainty modelling, knowledge-based systems, fuzzy sets

1 INTRODUCTION

The dominant portion of human knowledge can not be modeled by pure inference systems, because this knowledge is often ambiguous, incomplete and vague. The study of inference systems is tackled by several – and often very different – approaches. When knowledge is represented as a set of facts and rules, this uncertainty can be handled by means of fuzzy logic. The concept of deductive databases and fuzzy logic is discussed in classical works such as [8, 9, 11, 12, 17].

A few years ago in [5, 1] a possible combination of Datalog-like languages and fuzzy logic was presented. In these works we introduced the concept of fuzzy Datalog

by completing the Datalog-rules and facts by an uncertainty level and an implication operator. In [6] we provided an extension of fuzzy Datalog to fuzzy relational databases.

In parallel with these works, research was conducted on the possible combination of the Prolog language and fuzzy logic. Several solutions were suggested for this problem: these solutions propose different methods for handling uncertainty. Most of them use the concept of similarity, but in various ways. They consider the similarity as a reflexive, symmetric and transitive relation. Some of them take an arbitrary classification according to similarities and then use equivalence classes to make unification and fuzzy resolution, for example [4, 15]. [18, 19] discuss linguistic variables and their linguistic values, and provide the definition of the fuzzy unification algorithm by means of these values. In [16] the authors discuss a special kind of fuzzy unification. They deal with the problem of missing parameters and mismatching predicates and parameters. They define the edit distance of terms, which is compound according to the number of mismatching, and give a unification algorithm according to these distances.

Based upon our previous work, here we present another possible model for handling uncertain information, based on the extension of fuzzy Datalog. We build the model of fuzzy knowledge-base as a quadruple of background knowledge, a deduction mechanism, a connecting algorithm, and a decoding set.

2 THE FUZZY DATALOG

A Datalog program consists of facts and rules. In fuzzy Datalog, we can complete the facts by an uncertainty level, the rules by an uncertainty level and an implication operator. This means that evaluating the fuzzy implication connecting to the rule, its truth-value according to the implication operator is at least the given uncertainty level.

For example, if the implication operator is the Gödel-operator:

$$\mathrm{I}(\alpha,\beta) = \begin{cases} 1 & \text{if} \quad \alpha \leq \beta, \\ \beta & \text{otherwise,} \end{cases}$$

then the level of the rule-head can be calculated as the minimum of the level of the rule-body and the level of the rule.

In [5, 6] we give an extension of Datalog-like languages to fuzzy relational databases. In these papers lower bounds of degrees of uncertainty in facts and rules are used. This language is called fuzzy Datalog (fDATALOG). In this language the rules are completed by an implication operator and a level. We can infer the level of a rule-head from the level of the body, and the level of the rule by the implication operator of the rule.

Here we are going to summarize the concept of fDATALOG based on [5, 6]. Similarly to Datalog programs, a fDATALOG program consists of rules and facts. The notion of fuzzy rule is given in definition below.

Definition 1. A fDATALOG rule is a triplet r;β;I, where r is a formula of the form

$$\mathrm{A} \leftarrow \mathrm{A}_1, \ldots, \mathrm{A}_n \qquad (n \geq 0).$$

A is an atom (the head of the rule), $\mathrm{A}_1, \ldots, \mathrm{A}_n$ are literals (the body of the rule); I is an implication operator and $\beta \in (0, 1]$ (the level of the rule).

For getting finite result, all the rules in the program must be safe. A fDATALOG rule is safe if

- all variables occurring in the head also occur in the body;
- all variables occurring in a negative literal also occur in a positive literal.

An fDATALOG program is a finite set of safe fDATALOG rules.

There is a special type of rule, called fact. A fact has the form $\mathrm{A} \leftarrow; \beta; \mathrm{I}$. From now on, we refer to facts as (A, β), because according to implication I, the level of A easily can be computed.

The semantics of fDATALOG is defined as the fixed points of consequence transformations. Depending on these transformations, we can define two semantics for fDATALOG. The deterministic semantics is the least fixed point of the deterministic transformation DT_P, the nondeterministic semantics is the least fixed point of the nondeterministic transformation NT_P. According to the deterministic transformation, the rules of a program are evaluated in parallel, while in the nondeterministic case the rules are considered independently and sequentially.

These transformations are as follows:

Definition 2. Let B_P the Herbrand base of the program P, and let $\mathrm{F}(\mathrm{B}_P)$ denote the set of all fuzzy sets over B_P. The consequence transformations

$$\mathrm{DT}_P : \mathrm{F}(\mathrm{B}_P) \rightarrow \mathrm{F}(\mathrm{B}_P) \text{ and } \mathrm{NT}_P : \mathrm{F}(\mathrm{B}_P) \rightarrow \mathrm{F}(\mathrm{B}_P)$$

are defined as

$$\mathrm{DT}_P(\mathrm{X}) = \{\cup\{(\mathrm{A}, \alpha_{\mathrm{A}})\} | (\mathrm{A} \leftarrow \mathrm{A}_1, \ldots, \mathrm{A}_n; \mathrm{I}; \beta) \in \mathrm{ground}(P), (|\mathrm{A}_i|, \alpha_{\mathrm{A}_i}) \in \mathrm{X}$$

$$\text{for each} 1 \leq i \leq n, \alpha_{\mathrm{A}} = \max(0, \min\{\gamma | \mathrm{I}(\alpha_{\mathrm{body}}, \gamma) \geq \beta\})\} \cup \mathrm{X},$$

and

$$\mathrm{NT}_P(\mathrm{X}) = \{(\mathrm{A}, \alpha_{\mathrm{A}})\} \cup \mathrm{X},$$

Here

$$(\mathrm{A} \leftarrow \mathrm{A}_1, \ldots, \mathrm{A}_n; I; \beta) \in \mathrm{ground}(P), (|\mathrm{A}_i|, \alpha_{\mathrm{A}_i}) \in \mathrm{X}, 1 \leq i \leq n,$$

$$\alpha_{\mathrm{A}} = \max(0, \min\{\gamma | \mathrm{I}(\alpha_{\mathrm{body}}, \gamma) \geq \beta\}).$$

$|\mathrm{A}_i|$ denotes the kernel of the literal A_i, (i.e., it is the ground atom A_i, if A_i is a positive literal, and $\neg\mathrm{A}_i$, if A_i is negative).

In [6] it is proved that starting from the set of facts, both DT_P and NT_P have fixed points which are the least fixed points in the case of positive P. These fixed points are denoted by $\mathrm{lfp}(\mathrm{DT}_P)$ and $\mathrm{lfp}(\mathrm{NT}_P)$. It was also proved that $\mathrm{lfp}(\mathrm{DT}_P)$ and $\mathrm{lfp}(\mathrm{NT}_P)$ are models of P, so we could define $\mathrm{lfp}(\mathrm{DT}_P)$ as the deterministic semantics, and $\mathrm{lfp}(\mathrm{NT}_P)$ as the nondeterministic semantics of fDATALOG programs.

For a function- and negation-free fDATALOG, the two semantics are the same, but they are different if the program has any negation. In this case the set $\mathrm{lfp}(\mathrm{DT}_P)$ is not always a minimal model, but the nondeterministic semantics – $\mathrm{lfp}(\mathrm{NT}_P)$ – is minimal under certain conditions. This condition is referred to as stratification: stratification gives an evaluating sequence in which the negative literals are evaluated first. (For details, see [6].)

Example 1. Consider the fDATALOG program

1. $(\mathrm{r(a)}, 0.8)$.
2. $\mathrm{p(x)} \leftarrow \mathrm{r(x)}, \neg\mathrm{q(x)}; 0.6; I$.
3. $\mathrm{q(x)} \leftarrow \mathrm{r(x)}; 0.5; I$.
4. $\mathrm{p(x)} \leftarrow \mathrm{q(x)}; 0.8; I$.

Then the stratification is: $P_1 = \{r, q\}$, $P_2 = \{p\}$, so the evaluation order is: 1., 3., 2., 4. (More precisely: first 1. and 3. in arbitrary order, then 2. and 4. in arbitrary order.) Then in the case of Gödelian implication operator, the nondeterministic semantics of the program is $\mathrm{lfp}(\mathrm{NT}_P) = \{(\mathrm{r(a)}, 0.8); (\mathrm{p(a)}, 0.5); (\mathrm{q(a)}, 0.5)\}$.

From now on, we shall deal only with the case of nondeterministic semantics.

3 BACKGROUND KNOWLEDGE

The facts and rules of a fDATALOG program can be regarded as any kind of knowledge, but sometimes – as in the case of our model – we need also other information, in order to get an answer for a query. In this section, we give a model of background knowledge. We define proximity between predicates and between constants: these structures of proximity will serve for the background knowledge.

Definition 3. A proximity on a domain D is a fuzzy subset $\mathrm{S_D} : \mathrm{D} \times D \to [0, 1]$ such that the following properties hold:

$$\mathrm{S_D}(x, x) = 1 \text{ for any x } \in \mathrm{D} \text{ (reflexivity)}$$

$$\mathrm{S_D}(x, y) = \mathrm{S_D}(y, x) \text{ for any x, y } \in \mathrm{D} \text{ (symmetry).}$$

If a proximity is transitive, that is

$$\mathrm{S_D}(x, z) \geq \min(\mathrm{S_D}(x, y), \mathrm{S_D}(y, z)) \text{ for any x, y, z} \in \mathrm{D},$$

then it is called similarity.

Let us note that in most of practical cases the elements of a domain are in proximity relation. If they are in similarity, then we can define equivalence classifications over D, which allows us to develop simpler or more effective algorithms.

There is a simple method for deciding transitivity, using a matrix for describing proximity. A proximity matrix is a matrix containing the proximity values of each pair of elements in D.

Let S be a proximity matrix. The proximity is transitive (therefore it is a similarity) if and only if

$$\mathrm{S} \geq \mathrm{S} \cdot \mathrm{S},$$

where in the matrix-multiplication the minimum of elements are constituted instead of multiplying and the maximums are formed instead of sums.

Example 2. Let us consider the proximity matrix

	a	b	c	d	e
a	1	0.7	0.8	0.7	0.8
b	0.7	1	0.7	0.9	0.7
c	0.8	0.7	1	0.7	0.8
d	0.7	0.9	0.7	1	0.7
e	0.8	0.7	0.8	0.7	1

It can be easily checked that the relation defined by this matrix is transitive, so this proximity is similarity.

In our model the background knowledge is a set of proximity sets.

Definition 4. Let $\mathrm{d} \in \mathrm{D}$ any element of domain D. The proximity set of d is a fuzzy subset over D:

$$\mathrm{S_d} = \{(d_1, \lambda_1), (d_2, \lambda_2), \ldots, (d_n, \lambda_n)\},$$

where $\mathrm{d}_i \in \mathrm{D}$ and $\mathrm{S_D}(\mathrm{d}, \mathrm{d}_i) = \lambda_i$ for $i = 1, \ldots, n$.

Based on proximities we can construct the background knowledge which is information about the proximity of terms and predicate symbols.

Definition 5. Let C be any set of ground terms and R any set of predicate symbols. Let SC and SR be any proximity over C and R, respectively. The background knowledge is:

$$\mathrm{Bk} = \{\mathrm{SC_t} | \mathrm{t} \in \mathrm{T}\} \cup \{\mathrm{SR_p} | \mathrm{p} \in P\}.$$

4 FUZZY KNOWLEDGE-BASE

So far, we have made two steps on the way leading to the concept of fuzzy knowledge-base: we defined the concept of a fuzzy Datalog program and the concept of background knowledge. Now the question is: how can we connect this program with the

background knowledge? In [2, 3] we presented a possible connecting algorithm, now we show a different possibility.

To address this problem, we shall define the concept of modified fDATALOG program, which is the extension of the original one according to proximity; and the concept of decoding functions, which compute the final value of the uncertainty.

Evaluating a fDATALOG program, the facts and the result atoms are completed by uncertainty levels. The final uncertainty levels can be computed from these levels and from the proximity values of actual predicates and its arguments. It is expectable that in the case of identity the level must be unchanged, but in other cases it is to be less than or equal to the original level or to the proximity values. Furthermore we require the decoding function to be monotonically increasing.

Definition 6. A decoding function is an $(n+2)$-ary function:

$$\varphi(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) : (0,1] \times (0,1] \times (0,1] \times \cdots \times (0,1] \to [0,1]$$

such that

$$\varphi(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) \leq \min(\alpha, \lambda, \lambda_1, \ldots, \lambda_n),$$

$$\varphi(\alpha, 1, 1, \ldots, 1) = \alpha$$

and

$$\varphi(\alpha, \lambda, \lambda_1, \ldots, \lambda_n)$$

is monotonically increasing in all arguments.

Example 3.

$$\begin{aligned}
\varphi_1(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) &= \min(\alpha, \lambda, \lambda_1, \ldots, \lambda_n),\\
\varphi_2(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) &= \min(\alpha, \lambda, (\lambda_1 \cdot \cdots \cdot \lambda_n)),\\
\varphi_3(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) &= \alpha \cdot \lambda \cdot \lambda_1 \cdot \lambda_n
\end{aligned}$$

are decoding functions.

It is worth mentioning that any triangular norm is suitable for decoding function, for example the above min and product operators are t-norms.

We have to order decoding functions to all predicates of the program. The set of decoding functions will be the decoding set of the program. To define this set, we need the concept of the functor, which is characterized by the predicate symbol and the argumentum's number of an atom, that is for example in the case of $\mathrm{p}(\mathrm{t}_1, \mathrm{t}_2, \ldots, \mathrm{t}_n)$, the functor is p/n.

Definition 7. Let P be a fuzzy Datalog program, and F_P be the set of the program's functors. The decoding set of P is:

$$\Phi_P = \{\varphi_{\mathrm{q}}(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) | \forall q/n \in F_P\}$$

Let P be a fuzzy Datalog program, Bk be any background knowledge and Φ_P be the decoding set of P. Now we want to connect the program with background knowledge. For this purpose we decide the modified fDATALOG program mP. This is the original program with modified consecution transformation. The original consequence transformation is defined over the set of all fuzzy sets of P's Herbrand base, that is over $\mathrm{F}(\mathrm{B}_P)$. To define the modified transformation's domain, let us extend P's Herbrand universe with all possible ground terms occurring in background knowledge: this way, we obtain the modified Herbrand universe mH_P. Let the modified Herbrand base mB_P be the set of all possible ground atoms whose predicate symbols occur in $P \cup \mathrm{Bk}$ and whose arguments are elements of mH_P. This leads to

Definition 8. The modified consequence transformation

$$\mathrm{mNT}_P : \mathrm{F}(\mathrm{mB}_P) \to \mathrm{F}(\mathrm{mB}_P)$$

is defined as

$$\mathrm{mNT}_P(\mathrm{X}) = \{(\mathrm{q}(\mathrm{s}_1, \ldots, s_n), \phi_{\mathrm{p}}(\alpha, \lambda_{\mathrm{q}}, \lambda_{s_1}, \ldots, \lambda_{s_n}) | (\mathrm{q}, \lambda_{\mathrm{q}}) \in \mathrm{SR}_{\mathrm{p}};$$

$$(\mathrm{s}_i, \lambda_{s_i}) \in \mathrm{SC}_{t_i}, 1 \leq i \leq n\} \cup \mathrm{X},$$

where

$$(\mathrm{p}(\mathrm{t}_1, \ldots, \mathrm{t}_n) \leftarrow \mathrm{A}_1, \ldots, \mathrm{A}_k; I; \beta) \in \mathrm{ground}(P),$$

$$(|\mathrm{A}_i|, \alpha_{A_i}) \in X, 1 \leq i \leq k, \alpha = \max(0, \min\{\gamma | I(\alpha_{\mathrm{body}}, \gamma) \geq \beta\}),$$

and $|\mathrm{A}_i|$ denotes the kernel of the literal A_i.

Then starting from the facts of the program and creating the powers of the transformation mNT_P, we finally reach the fixed point.

Proposition 1. The modified consequence transformation mNT_P has a fixed point, i.e. there exists $\mathrm{X} \in \mathrm{F}(\mathrm{mB}_P)$ for which $\mathrm{mNT}_P(\mathrm{X}) = \mathrm{X}$. If P is positive, then this is the least fixed point, denoted by $\mathrm{X} = \mathrm{lfp}(\mathrm{mNT}_P)$, for any $\mathrm{Z} = \mathrm{mNT}_P(\mathrm{Z}) : \mathrm{X} \leq \mathrm{Z}$.

Proof. It was shown in [8, 10] that if T is an inflationary transformation over a complete lattice L, then T has a fixed point. (T is inflationary if $\mathrm{X} \leq \mathrm{T}(\mathrm{X})$ for every $\mathrm{X} \in \mathrm{L}$.) If T is monotone ($\mathrm{T}(\mathrm{X}) \leq \mathrm{T}(\mathrm{Y})$ if $\mathrm{X} \leq \mathrm{Y}$) then T has a least fixed point (see in [11]). Since mNT_P is inflationary and $\mathrm{F}(\mathrm{mB}_P)$ is a complete lattice, thus it has an inflationary fixed point. If P is positive, then this transformation is monotone, and thus the proposition is true. □

It can be shown that this fixed point is a model of P, but $\mathrm{lfp}(\mathrm{NT}_P) \subseteq \mathrm{lfp}(\mathrm{mNT}_P)$, so it is not a minimal model.

As the modifying algorithm has no effect on the order of rules, therefore it does not change the stratification. Therefore we can state

Proposition 2. In the case of stratified program P, mNT_P has least fixed point.

Now we have all components together to define the concept of a fuzzy knowledge-base.

Definition 9. A fuzzy knowledge-base (fKB) is a quadruple (Bk, P, Φ_P, mA), where Bk is a background knowledge, P is a fuzzy Datalog program, Φ_P is a decoding set of P and mA is any modifying (or connecting) algorithm.

Definition 10. Let (Bk, P, Φ_P, mA) be a fuzzy knowledge-base. Then lfp(mNT_P) is the consequence of the knowledge-base, denoted by C(Bk, P, Φ_P, mA).

Example 4. Let the fDATALOG program, the background knowledge and the decoding set be as follows (according to the modifying algorithm, it is sufficient to consider only the decoding functions of head-predicates). The implication operator of the program let be the Gödelian.

$$\mathrm{lo(x,y)} \leftarrow \mathrm{gc(y)}, \mathrm{mu(x)}; 0.7; \mathrm{I}.$$
$$(\mathrm{fv(V)},\ 0.9).$$
$$(\mathrm{mf(M)},\ 0.8).$$

$$\mathrm{I}(\alpha,\beta) = \begin{cases} 1 & \text{if} \quad \alpha \le \beta, \\ \beta & \text{otherwise.} \end{cases}$$

	B	V	M
B	1	0.9	
V	0.9	1	
M			1

	lo	li	gc	fv	mu	mf
lo	1	0.8				
li	0.8	1				
gc			1	0.75		
fv			0.75	1		
mu					1	0.6
mf					0.6	1

$\phi_{\mathrm{lo}} := \phi = \phi(\alpha,\mathrm{x},\mathrm{y},\mathrm{z}) := \min(\alpha,\mathrm{x},\mathrm{y},\mathrm{z})$
$\phi_{\mathrm{fv}} := \theta = \theta(\alpha,\mathrm{x},\mathrm{y}) := \alpha{\cdot}\mathrm{x}{\cdot}\mathrm{y}$
$\phi_{\mathrm{mf}} := \omega = \omega(\alpha,\mathrm{x},\mathrm{y}) := \min(\alpha,\ \mathrm{x}{\cdot}\mathrm{y})$

Applying the modification algorithm, and making use of the facts, we obtain

$$\{(\mathrm{fv(V)},0.9),\ (\mathrm{mf(M)},0.8)\}$$

$$\Downarrow \quad \text{(according to proximity)}$$

$$\{(\mathrm{fv(V)},0.9),\ (\mathrm{gc(V)},\theta(0.9,0.75,1) = 0.9{\cdot}0.75{\cdot}1 = 0.675),\ (\mathrm{fv(B)},\ \theta(0.9,1,0.9) = 0.81),\ (\mathrm{gc(B)},\ \theta(0.9,0.75,0.9) = 0.6075),\ (\mathrm{mf(M)},0.8),\ (\mathrm{mu(M)},\ \omega(0.8,0.6,1) = \min(0.8,0.6{\cdot}1) = 0.6)\}$$

$$\Downarrow \quad \text{(applying the rules)}$$

$$\mathrm{lo(M,V)} \text{ :- } \mathrm{gc(V)},\ \mathrm{mu(M)}\ 0.7;\ \mathrm{I}. \quad \Rightarrow \quad (\mathrm{lo(M,V)},\ \min(0.675,\ 0.6,\ 0.7) = 0.6)$$
$$\mathrm{lo(M,B)} \text{ :- } \mathrm{gc(B)},\ \mathrm{mu(M)};\ 0.7;\ \mathrm{I}. \quad \Rightarrow \quad (\mathrm{lo(M,B)},\ \min(0.6075,\ 0.6,\ 0.7) = 0.6)$$

$$\Downarrow \quad \text{(according to proximity)}$$

$$\text{li(M,V)},\phi(0.6,0.8,1,1) = \min(0.6,0.8,1,1) = 0.6,\ \text{li(M,B)},\phi((0.6,0.8,0.9,1) = \min(0.6,1,0.9,1) = 0.6.$$

So the consequence of knowledge-base is:

$$\begin{aligned} C(\text{Bk}, P, \Phi_P, \text{mA}) = \{&(\text{fv(V)},0.9);\ (\text{gc(V)},0.675);\\ &(\text{fv(B)},0.81);\ (\text{gc(B)},0.6075);\ (\text{mf(M)},0.8);\ (\text{mu(M)},0.6);\\ &(\text{lo(M,V)},0.6);\ (\text{lo(M,B)},0.6);\ (\text{li(M,V)},0.6);\ \text{li(M,B)},0.6)\}. \end{aligned}$$

5 EVALUATION STRATEGIES

According to Example 3 (especially in the case of enlarging the program with other facts and rules), it can be seen that the fixed point-query – that is the *bottom-up evaluation* – may involve many superfluous calculations, because sometimes we want to give an answer to a concrete question, and we are not interested in the whole sequence. If a goal (query) is specified together with the fuzzy knowledge-base, then it is sufficient to consider only the rules and facts necessary to reach the goal. In this section we deal with the *top-down evaluation* of knowledge-base: this starts from the goal, and applies the suitable rules and similarities to find the required starting facts and rules to get the answer to this query.

A *goal* is a pair $(\text{q}(\text{t}_1, \text{t}_2, \ldots, \text{t}_n), \alpha)$, where $\text{q}(\text{t}_1, \text{t}_2, \ldots, \text{t}_n)$ is an atom, α is the level of the atom. It is possible that among the arguments of q there are given or wanted variables, and α can also be either a given or a wanted variable.

In some cases, during the top down evaluation the goal is evaluated through sub-queries. This means that all possible rules are selected, whose head can be unified with the given goal, and the atoms of the body are considered as new sub-goals. This procedure continues until the facts are obtained.

The paper [1] deals with the evaluation strategies of a fuzzy Datalog. The top-down evaluation of a fuzzy Datalog does not terminate by obtaining the facts, because we need to determine the uncertainty level of the goal. The algorithm given in [1] calculates this level in a bottom-up manner: starting from the leaves of the evaluating graph, going backward to the root, and applying the uncertainty-level functions along the suitable path of this graph, finally we get the uncertainty level of the root.

In the more recent model, we rely on the bottom-up evaluation, but the selection of required starting facts takes place in a top-down fashion. Since only the required starting facts are sought, in the top-down part of the evaluation there is no need for the uncertainty levels. Hence, we search only among the ordinary facts and rules. To do this, we need the concept of substitution and unification which are given for example in [1, 8, 14, 17], etc. But now sometime we also need other kinds of substitutions: to substitute some predicate p or term t for their proximity sets S_p and S_t, and to substitute some proximity sets for their members.

Next, for the sake of simpler terminology, by goal, rules and facts we mean these concepts without uncertainty levels. An AND/OR tree arises during the evaluation; this is the *searching tree*. Its root is the goal; its leaves are either YES or NO. The parent nodes of YES are the required starting facts. This tree is built up by alternating proximity-based and rule-based unification.

The proximity-based unification unifies the predicate symbols of sub-goals by the members of its proximity set, except the first and last unification. The first proximity-based unification unifies the ground terms of the goal with their proximity sets, and the last one unifies the proximity sets among the parameters of resulting facts with their members.

The rule-based unification unifies the sub-goals with the head of suitable rules, and continues the evaluating by the bodies of these rules. During this unification the proximity sets of terms are considered as ordinary constants, and a constant can be unify with its proximity set. The searching graph according to its depth is build up in the following way:

If the goal is on depth 0, then every successor of any node on depth $3k+2$ $(k=0,1,\dots)$ is in AND connection, the others are in OR connection. In detail:

The successors of goal $\mathrm{g}(\mathrm{t}_1,\mathrm{t}_2,\dots,\mathrm{t}_n)$ will be all possible $\mathrm{g}'(\mathrm{t}'_1,\mathrm{t}'_2,\dots,\mathrm{t}'_n)$, where $\mathrm{g}'\in \mathrm{S}_g; \mathrm{t}'_i=\mathrm{t}_i$ if t_i is some variable and $\mathrm{t}'_i=\mathrm{S}_{t_i}$ if t_i is a ground term.

If the atom $\mathrm{p}(\mathrm{t}_1,\mathrm{t}_2,\dots,\mathrm{t}_n)$ is in depth $3k$ $(k=1,2,\dots)$, then the successor nodes are all possible $\mathrm{p}'(\mathrm{t}_1,\mathrm{t}_2,\dots,\mathrm{t}_n)$, where $\mathrm{p}'\in \mathrm{S}_p$.

If the atom L is in depth $3k+1$ $(k=1,2,\dots)$, then the successor nodes will be the bodies of suitable unified rules, or the unified facts, if L is unifiable with any fact of the program, or NO, if there is not any unifiable rule or fact. That is, if the head of rule $\mathrm{M}\leftarrow \mathrm{M}_1,\dots,\mathrm{M}_n$ $(n>0)$ is unifiable with L, then the successor of L be $\mathrm{M}_1\theta,\dots,\mathrm{M}_n\theta$, where θ is the most general unification of L and M. If $n=0$, that is, in the program there is any fact with the predicate symbol of L, then the successors are the unified facts. If $\mathrm{L}=\mathrm{p}(\mathrm{t}_1,\mathrm{t}_2,\dots,\mathrm{t}_n)$ and in the program there is any fact with predicate symbol p, then the successor nodes are all possible $\mathrm{p}(\mathrm{t}'_1,\mathrm{t}'_2,\dots,\mathrm{t}'_n)$, where $\mathrm{t}'_i\in \mathrm{S}_{\mathrm{t}_i}$ if $\mathrm{t}_i=\mathrm{S}_{\mathrm{t}_i}$ or $\mathrm{t}'_i=\mathrm{t}_i\theta$, if t_i is a variable, and θ is a suitable unification.

According to the previous paragraph, there are three kinds of nodes in depth $3k+2$ $(k=1,2,\dots)$: a unified body of a rule; a unified fact with ordinary ground term arguments; or the symbol NO. In the first case the successors are the members of the body. They are in AND connection, which is not important in our context, but maybe important for possible future development. If the body has only one literal, then the length of evaluating path would be reduced to one, but it would "damage" the view of homogeneous treatment. In the second case the successors are the symbol YES or NO, depending on whether the unified fact is among the ground atoms of the program. The NO-node has no successor.

From the construction of searching graph, we conclude

Proposition 3. Let X_0 be the set of ground facts being in parent-nodes of symbols YES. Starting from X_0, the fixed point of mNT_P contains the answer to the query.

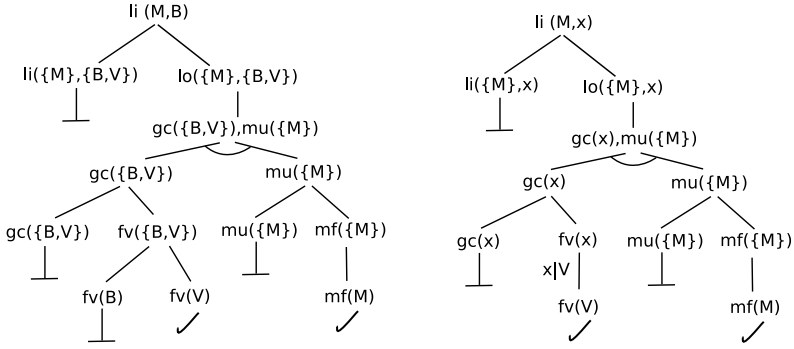
From the viewpoint of the query, this fixed point may contain more superfluous ground atom, but generally it is smaller than the consequence of knowledge-base. More reduction of the number of superfluous resulting facts is the work of a possible further development.

Example 5. Let us consider the knowledge-base of Example 3. (Now it is sufficient to consider only the program and the background knowledge.)

Let the goal be:

1. li(M,B).
2. li(M,x), where x is a variable.

Then the searching graphs are:



According to the above construction, the algorithm of searching the starting facts is the following alternation of proximity-based and rule-based unification.

Algorithm

procedure evaluation(g($\underline{\mathbf{t}}$)) /* g($\underline{\mathbf{t}}$) *is the goal* */
  Heads := {the heads of the program's rules}
  Facts := {the facts of the program}
  Resulting_Facts := ∅ /* *the set of resulting starting facts* */
  for all t ∈ $\underline{\mathbf{t}}$ do
    if is_variable(t) then s := t
      else s := St /* *St is the proximity set of t* */
    end_if
  end_for
  Nodes := {g($\underline{\mathbf{s}}$)}
    /*Nodes *is the set of evaluable nodes,*
    $\underline{\mathbf{s}}$ *is the vector of elements* s *in the original order* */
  New_nodes := ∅ /* *the successor nodes of* Nodes */
  while not_empty(Nodes) do
    p($\underline{\mathbf{t}}$) := element(Nodes)
    Spnodes := ∅ /* *the successor nodes of* p($\underline{\mathbf{t}}$) */
    proximity_evaluation(p($\underline{\mathbf{t}}$),Spnodes)

```
        New_nodes := New_nodes ∪ Spnodes
        Nodes := Nodes – {p(t)}
    end_while
    Nodes := New_nodes
    New_nodes := ∅
    while not_empty(Nodes) do
        p(t) := element(Nodes)
        Spnodes := ∅              /* the successor nodes of p(t) */
        rule_evaluation(p(t),Spnodes)
        New_nodes := New_nodes ∪ Spnodes
        Nodes := Nodes – {p(t)}
    end_while
    return Resulting_Facts
end_procedure
procedure proximity_evaluation(p(t),Spnodes)
    for all q ∈ Sp do             /* Sp is the proximity set of p */
        Spnodes := Spnodes ∪ {q(t)}
    end_for
end_procedure
procedure rule_evaluation(p(t),Spnodes)
    for all p(v) ∈ Heads do
        if is_unifiable(p(t),p(v)) then
        Spnodes := Spnodes ∪ {unified predicates of the body belonging to p(vθ)}
                        /* θ is the suitable unifier */
        end_if
    end_for
    for all p(v) ∈ Facts do
        if is_unifiable(p(t),p(v)) then
        for all St ∈ vθ do          /* θ is the suitable unifier */
            if is_variable(St) then
            t := Stτ          /* τ is the suitable unifier */
            else if is_proximity_set(St) then
            t := element(St)
            end_if
        end_for
        end_if
        for all possible t do
                /* t is the vector of elements t in the right order */
            if p(t) ∈ Facts then
                Resulting_Facts := Resulting_Facts ∪ {p(t)}
            end_if
        end_for
    end_for
end_procedure
```

This algorithm can be applied for stratified fDATALOG too, by determining the successor of a rule-body without negation.

6 CONCLUSIONS

In this paper we have presented a model of handling uncertain information by defining the fuzzy knowledge-base as a quadruple of background knowledge, a deduction mechanism, a decoding set and some modifying algorithm which connects the background knowledge to the deduction mechanism. We also have presented a possible evaluation strategy. To improve this strategy and/or the modifying algorithm will be the subject of further investigations. An efficient fuzzy knowledge base could be the basis of decisions based on uncertain information, or would be a possible method for handling argumentation or negotiation of agents.

To illustrate our discussion with some realistic content, in Examples 3 and 4 a the knowledge base could have the following interpretation. Let us suppose that music listeners are "generally" (level 0.7) fond of the greatest composers. Assume furthermore that Mary is a "rather devoted" (level 0.8) fan of classical music (mf), and Vivaldi is "generally accepted" (level 0.9) as a "great composer". It is also widely accepted that the music of Vivaldi and Bach are fairly "similar", being related in overall structure and style. On the basis of the above information, how strongly can be stated that Mary likes Bach? To continue with this idea, next we can assume that an internet agent wants to suggest a good CD for Mary, based on her interests revealed through her actions at an internet site. A fuzzy knowledge base could help the agent get a good answer. As some of the readers may well know, similar mechanisms – but possibly based on entirely different modelling paradigms – are in place in prominent websites such as Amazon and others.

REFERENCES

[1] Achs, Á: Evaluation Strategies of Fuzzy Datalog. Acta Cybernetica, Szeged, Vol. 13, 1997, pp. 85–102.

[2] Achs, Á: Fuzzy Datalog with Background Knowledge. Teaching Mathematics and Computere Science, Debrecen, 2005, pp. 1–25.

[3] Achs, Á: Fuzzy Knowledge-Base with Fuzzy Datalog – A Model for Handling Uncertain Information. ISDA 2004 – IEEE 4th International Conference on Intelligent System Design and Application, August 26–28, 2004, Budapest, pp. 55–60.

[4] Arcelli, F.—Formato, F.—Gerla, G.: Fuzzy Unification as Foundations of Fuzzy Logic Programming. In Logic Programming and Soft Computing, RSP-Wiley, England, 1998.

[5] Achs, Á—Kiss, A.: Fixed Point Query in Fuzzy Datalog. Annales Univ. Sci. Budapest, Sect. Comp., Vol. 15, 1995, pp. 223–231.

[6] Achs, Á—Kiss, A.: Fuzzy Extension of Datalog. Acta Cybernetica, Szeged , Vol. 12, 1995, pp. 153–166.

[7] Baldwin, J. F.—Martin, T. P.: Learning Uncertain Logic Programs from Examples. Logic Programming and Soft Computing, LPSC98, Manchester, 1998.

[8] Ceri, S.—Gottlob, G.—Tanca, L.: Logic Programming and Databases. Springer-Verlag Berlin, 1990.

[9] Dubois, D.—Prade, H.: Fuzzy Sets in Approximate Reasoning. Part 1: Inference with Possibility Distributions. Fuzzy Sets and Systems, Vol. 40, 1991, pp. 143–202.

[10] Gurevich, Y.—Shelah, S.: Fixed-Point Extensions of First-Order Logic. IEEEE Symp. on FOCS, 1985, pp. 346–353.

[11] Lloyd, J. W.: Foundations of Logic Programming. Springer-Verlag, Berlin, 1987.

[12] Novák, V.: Fuzzy Sets and Their Applications. Adam Hilger, Bristol and Philadelphia, 1989.

[13] Ovchinnikov, S.: Similarity Relations, Fuzzy Partitions, and Fuzzy Ordering. Fuzzy Sets and Systems, Vol. 40, 1991, pp. 107–126.

[14] Pásztorné Varga, K.: A Matematikai Logika és Alkalmazásai. Tankönyvkiadó, Bp, 1986.

[15] Sessa, M. I.: Approximate Reasoning by Similarity-Based SLD Resolution. Theoretical Computer Science, Vol. 275, 2002, pp. 389–426,

[16] Schroeder, M.—Schweimeier, R.: Arguments and Misunderstandings: Fuzzy Unification for Negotiating Agents. Electronic Notes in Theoretical Computer Science, Vol. 70, 2002, No. 5, Elsevier Proceedings of the ICLP workshop CLIMA02, Copenhagen, Aug. 2002.

[17] Ullman, J. D.: Principles of Database and Knowledge-Base Systems. Computer Science Press, Rockville, 1988.

[18] Virtanen, H. E.: Fuzzy unification. 5th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, 1994, July 4–8, Paris.

[19] Virtanen, H. E.: Vague Domains, S-Unification and Logic Programming. Electronic Notes in Theoretical Computer Science, Vol. 66, 2002, No. 5, URL: `http://www.elsevier.nl/locate/entcs/volume66.html`.

Ágnes Achs received the M. Sc. in mathematics (1976) from the Kossuth Lajos University of Debrecen, M. Sc. in computing (1993) form the Eötvös Lóránt University of Budapest, Univ. D. in mathematics (1995) and Ph. D. in computing (2006) from University of Debrecen. Her research interests include fuzzy databases and soft computing.