Fuzzy knowledge-base with fuzzy Datalog a model for handling uncertain information

Ágnes Achs
Departement of Computer Science
University of Pécs, Faculty of Engineering
Pécs
Boszorkány u. 2
Hungary
achs@witch.pmmf.hu

Abstract **In this paper a possible model of handling uncertain information is given. The model is worked out in the framework of DATALOG. We define the concept of fuzzy knowledge-base, which consist of a background knowledge, defined by the similarity of predicates and terms, and a deduction mechanism, the fuzzy Datalog.**

I. INTRODUCTION

Two people talk in the shop:

- I would like to buy this CD. Do you know: whether Mary like Bach?
- I think, because her favourite composer is Vivaldi.

It is a simple dialogue, but if we would like to model it, we have to deal with the background knowledge and the deduction mechanism, being behind the answer. Modelling this dialog we have to deduce from the statement "`love('Mary','Vivaldi')`" to the statement "`like('Mary','Bach')`" or the statement "`is_fond_of('Mary','Bach')`". Moreover we want to deduce from the statement "`as I know love('Mary','Vivaldi')`" to the statement "`maybe like('Mary','Bach')`".

In this paper we give a model of handling uncertain information, we build a model of fuzzy knowledge-base, which consists of a background knowledge – some similarity relations; and a fuzzy deduction mechanism – the fuzzy Datalog. The concept of fuzzy Datalog and it's possible evaluation strategies are given in [1], [2] and [3], in this paper we connect this concepts with a background knowledge, and give a possible model for the solution of the above example.

II. THE FUZZY DATALOG

In [1], [2] there was given a possible extension of Datalog-like languages to fuzzy relational databases using lower bounds of degrees of uncertainty in facts and rules. This language is called fuzzy Datalog (fDATALOG). In this language the rules are completed with 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.

For example: if we know, that John usually likes the beautiful girls, and Mary is fairly beautiful, whether John like Mary? The fact, that Mary is fairly beautiful, can be written as a DATALOG fact completed with a level:

`beautiful('Mary'); 0,7`

which means, that Mary is beautiful at least 0.7 level.

John's emotions can be written by a rule:

`likes('John',x)←beautiful(x);I;0.8`

This means, that the truth-value of the above fuzzy implication according to the implication operator I, is at least 0.8.

If for example, this implication operator is the Gödel-operator:

$$I(x,y)=\begin{cases}1 & \text{if } x \le y\\ y & \text{otherwise}\end{cases}$$

then the level of the rule-head can be count as the minimum of the level of the rule-body and the level of the rule. For John and Mary it does:

`likes('John','Mary'), 0.7`

that is John likes Mary at least 0.7 level.

For the universality, we allow the use of arbitrary implication operator, and the facts are given as rules without body. Now we are going to summarise the concept of fDATALOG precisely.

The concept of fDATALOG

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

$$Q \leftarrow Q_1,\ldots,Q_n \qquad (n\ge 0).$$

Q is an atom (the head of the rule), $Q_1,\ldots,Q_n$ are literals (the body of the rule); I is an implication operator and $\beta \in (0,1]$ (the level of the rule).

An fDATALOG rule is safe if all variables occurring in the head also occur in the body and all variables occurring in a negative literal also occur in a positive literal. For getting finite result, all the rules in the program must be safe. An fDATALOG program is a finite set of safe fDATALOG rules.

The semantics of fDATALOG is defined as the fixpoints of consequence transformations. Depending on these transformations we can define two semantics for fDATALOG. The deterministic semantics is the least fixpoint of deterministic transformation, the nondeterministic semantics is the least fixpoint of nondeterministic transformation. By the deterministic transformation the rules of a program are evaluated parallel, while in nondeterministic case the rules are considered independently one after another.

These transformations are the following:

Definition 2.
Let B_P the Herbrand base of the program P, and let $F(B_P)$ denote the set of all fuzzy sets over B_P. The consequence transformations $DT_P : F(B_P) \to F(B_P)$ and $NT_P : F(B_P) \to F(B_P)$ are defined for any $X \in F(B_P)$ as

$$DT_P(X)=\{\cup\{(A,\alpha_A)\} \mid (A \leftarrow A_1,\ldots,A_n;I;\beta) \in \mathtt{ground}(P),\ (|A_i|,\alpha_{A_i}) \in X \text{ for each } 1\le i\le n,\ \alpha_A=max(0,min\{\gamma \mid I(\alpha_{body},\gamma) \ge \beta\})\} \cup X$$

and

$$NT_P(X)=\{(A,\alpha_A)\} \cup X$$

for any

$$(A \leftarrow A_1,\ldots,A_n; I; \beta) \in \mathtt{ground}(P),\ (|A_i|,\alpha_{A_i}) \in X,\ 1\le i\le n,\ \alpha_A=max(0,min\{\gamma \mid I(\alpha_{body},\gamma) \ge \beta\})$$

respectively. |A| denotes $p(\underline{c})$ if either $A=p(\underline{c})$ or $A=\neg p(\underline{c})$ where p is a predicate symbol with arity k and $\underline{c}$ is a list of k ground terms; $\alpha_{body} = \alpha_{A_1\wedge\ldots\wedge A_n}$; ground(P) denotes the set of all ground instances of $(r;I;\beta)\in P$.

Let T_0 be the set of facts:

$$T_0 = \{\cup\{(A,\alpha_A)\} \mid (A \leftarrow ;I;\beta) \in \mathtt{ground}(P),\ \alpha_A= max(0,\ min\{\gamma \mid I(1,\gamma) \ge \beta\})\} \cup \{(A,0) \mid \exists (B \leftarrow \ldots \neg A\ldots;I;\beta) \in \mathtt{ground}(P)\}$$

and let for

$$T = DT_P : F(B_P) \to F(B_P) \text{ or}$$
$$T = NT_P : F(B_P) \to F(B_P)$$

$$T_1 = T(T_0)$$
$$\ldots$$
$$T_n = T(T_{n-1})$$

In [1] it was proved, that starting from T_0, both DT_P and NT_P have a fixpoint, which is the least fixpoint in the case of positive P. These fixpoints are the deterministic and the nondeterministic semantics of fDATALOG programs.

For 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 deterministic semantics is not always a minimal model, but the nondeterministic semantics is minimal for stratified fDATALOG programs, as was proved in [3].

Example 1.
Given the next fDATALOG program:

1. $r(a) \leftarrow ; I_1; 0.8$
2. $p(x) \leftarrow r(x),\neg q(x); I_2; 0.6$
3. $q(x) \leftarrow r(x); I_3; 0.5$
4. $p(x) \leftarrow q(x); I_4; 0.8$

Now let all of the implication operators be the Gödel-operator. The stratification is: {r,q}, {p}, so the evaluation order is: 1., 3., 2., 4. (Precisely: firstly 1. and 3. in arbitrary order, then 2. and 4. in arbitrary order.)
Then the least fixpoint of NT_P is:

{(r(a),0.8); (p(a),0.5); (q(a),0.5)}.

3. THE BACKGROUND KNOWLEDGE

The facts and rules of a fDATALOG program can be regarded as any kind of knowledge, but sometime – as in the case of our model – we are needed other information to give answer for a question. In this paragraph we give a model of background knowledge. We define similarity between predicates and between ground terms and these structures of similarity will construct the background knowledge.

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

$S_D(x,x) = 1$ for any $x\in D$ (reflexivity)
$S_D(x,y) = S_D(y,x)$ for any $x,y \in D$ (symmetry).

A similarity is transitive if

$$S_D(x,z) \ge max\{\, min\,(S_D(x,y), S_D(y,z))\,\}$$

for any $x,y,z \in D$

In our model we give a background knowledge by generating the similarity sets of predicates and ground terms. The similarity set is:

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

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

where $d_i \in D$ and $S_D(d, d_i) = \lambda_i$ for $i = 1,\ldots n$.

Sometime we are not interested in similarity when the two elements are too distant, namely the value of similarity is too small. Therefore we introduce the concept of λ-cut of similarity set:

Definition 5.

Let $d \in D$ any element of domain D and $0 < \lambda \leq 1$. The λ-cut of similarity set S_d is:

$$S_{d,\lambda} = \{(d_i, \lambda_i) \in S_d / \lambda_i \geq \lambda\}.$$

(In fuzzy terminology the α–cut is used rather than λ–cut, but for the sake of next notations we use this term.)

Easily can be shown that if the similarity is transitive, it defines λ-equivalence classifications over D.

Example 2.

Let us consider the next similarity 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 controllable, that the similarity written by this matrix is transitive.

Let $\lambda = 0.8$. The λ-cuts of similarity sets of D = {a,b,c,d,e} are:

$S_{a,\lambda} = \{(a, 1), (c, 0.8), (e, 0.8)\}$
$S_{b,\lambda} = \{(b, 1), (d, 0.9)\}$
$S_{c,\lambda} = \{(a, 0.8), (c, 1), (e, 0.8)\}$
$S_{d,\lambda} = \{(b, 0.9), (d, 1)\}$
$S_{e,\lambda} = \{(a, 0.8), (c, 0.8), (e, 1)\}$

These sets may define a λ-equivalence relation over D. Two elements $d_1, d_2 \in D$ are in λ-equivalence relation if $S_D(d_1,d_2) \geq \lambda$. According to this relation we can define the next equivalence classes over D:

$$E_{0.8} = \{a,c,e\}, \{b,d\}.$$

By the similarity we can construct the background knowledge, which is information about the similarity of terms and predicate symbols:

Definition 6.

Let T be any set of ground terms and P any set of predicate symbols. Let ST and SP any similarity over T and P respectively. The background knowledge is:

$$Bk = \{ST_t / t \in T\} \cup \{SP_p / p \in P\}.$$

Now we are ready to define the concept of fuzzy knowledgebase, and to give two algorithm, one to modify the original fDATALOG program, and one to get the uncertainty level of resulting atoms.

III. FUZZY KNOWLEDGE-BASE

A fuzzy knowledge-base consists of a background knowledge, a fuzzy deduction mechanism, and a function, which computes the final value of the uncertainty.

Modified fDATALOG program

Let P be a fuzzy Datalog program, and let Bk be any background knowledge. With the similarities of the background knowledge we can define the modified fDATALOG program, mP: Let us replace each predicate $p \in P$ by SP_p, each ground term $t \in H_p$ by ST_t and each variable x by X={x}.
(H_p denotes the Herbrand universe of P.)

Algorithm 1.

```
Procedure modification (P, SP, ST)
     while not(empty(P)) do
            (r;I;β) := first rule of P
            (R;I;β) := (replace(r);I;β)
            P := P - {(r;I;β)}
     endwhile
endprocedure

procedure replace(r)
     Pred_r := set of r's predicates
     Term_r := set of r's ground terms
     Var_r := set of r's variables

     while not(empty(Pred_r)) do
            q := first predicate of r
            Q := SP_q
            Pred_r := Pred_r - {q}
     endwhile
     while not(empty(Term_r)) do
            t := first ground term of r
            T := ST_t
            Term_r := Term_r - {t}
     endwhile
     while not(empty(Var_r)) do
            x := first variable of r
            X := {x}
            Var_r := Var_r - {x}
     endwhile
endprocedure
```

The modified fDATALOG program is evaluable as an ordinary fDATALOG program. There are only two problems:

1. How can we compute the uncertainty level of the results?
2. What can we do with the huge mass of result facts obtained due to the similarities?

The decoding function

To solve the first problem, we define the decoding function:

Definition 6.
A decoding function is an (n+2) variable fuzzy function:

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

so that

$$\varphi(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) \leq min(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) \text{ and}$$
$$\varphi(\alpha, 1, 1, \ldots, 1) = \alpha.$$

Example 3.

$\varphi_1(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) = \min(\alpha, \lambda, \lambda_1, \ldots, \lambda_n)$;
$\varphi_2(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) = \min(\alpha, \lambda, (\lambda_1 \cdot \cdots \lambda_n))$;
$\varphi_3(\alpha, \lambda, \lambda_1, \ldots, \lambda_n) = \alpha \cdot \lambda \cdot \lambda_1 \cdot \cdots \lambda_n$

are decoding functions.

Definition 7.
Let P be a fuzzy DATALOG program. The decoding set of P is:

$$\Phi_P = \{ \varphi_q(\alpha_q, \lambda_q, \lambda_{t_1}, \ldots, \lambda_{t_k}) / \forall q(t_1, t_2, \ldots, t_k) \in P \}.$$

Now there are all together to define the concept of fuzzy knowledge-base precisely:

Definition 8.
A fuzzy knowledge-base (fKB) is a triplet {Bk, P, Φ_P}, where Bk is a background knowledge, P is a fuzzy DATALOG program, and Φ_P is a decoding set of P.

Notes:
1. There is no need to be an involving relation between the predicates or ground terms of Bk and P respectively.
2. It is possible to define different fuzzy knowledge-bases over the same background knowledge.

Now with the similarities of the background knowledge we can define the modified fDATALOG program, mP. Evaluating this program, we get a fuzzy set of the ground terms of mP's Herbrand base. Applying the decoding functions to these terms, we get the consequence of the knowledge-base. More precisely:

Definition 9.
Let {Bk, P, Φ_P} a fuzzy knowledge-base. The consequence of the knowledge-base is

$$C(Bk, P, \Phi_P) =$$
$$\{(q(t_1, t_2, \ldots, t_n); \varphi_q(\alpha, \lambda_q, \lambda_{t_1}, \ldots, \lambda_{t_n})) / \text{for each}$$
$$(Q(T_1, T_2, \ldots, T_n); \alpha) \in lfp(NT_{mP}),$$
$$(q, \lambda_q) \in Q, (t_i, \lambda_{t_i}) \in T_i\},$$

where $lfp(NT_{mP})$ is the least fixpoint of NT_{mP}.

According to the definitions above, it is obvious:

$$lfp(NT_{mP}) \subseteq C(Bk, P, \Phi_P).$$

To solve the second problem, we have to discuss the evaluation strategies of fDatalog and modified fDatalog.

Evaluation strategies

[3] deals with the evaluation strategies of fuzzy Datalog. An fDATALOG program can be evaluated by different strategies.

The *bottom-up evaluation* starts from the facts, applies the rules, so it can deduce all of the computable facts, that is it can determine both $lfp(DT_P)$ or $lfp(NT_P)$.

In many cases however, there is no need for all facts of the least fixpoint, we want to get an answer only for a concrete question. If a goal is specified together with an fDATALOG program, it is enough to consider only the rules and facts which are necessary to reach the goal. The *top-down evaluation* starts from the goal, and applies the suitable rules to reach the given facts of the program.

In the case of modified fuzzy DATALOG program the bottom-up evaluation seems to be unreal in practical respect because of the huge mass of result facts obtained due to the similarities.

To applying the top down evaluation, we have to give any goal to the program. A *goal* is a pair $(q(t_1, t_2, \ldots t_n); \alpha)$, where $q(t_1, t_2, \ldots t_n)$ is an atom, α is the level of the atom. It is possible, that among the arguments of q there are variables or constants, and α can be either a constant or a variable. A fuzzy DATALOG enlarged with a goal is a *query*.

A goal can be evaluated with the aid of sub-queries. This means, that all of the rules, whose head-predicate can be unificated with the given goal-predicate are selected, and the predicates of these rule-bodies are considered as new sub-goals. This procedure continues until obtaining the facts. [3] give an algorithm for top-down evaluation of fDATALOG.

Top down evaluation strategy of fuzzy knowledge-base

We can define a query over the fuzzy knowledge-base {Bk, P, Φ_P}. According to Algorithm1., P is modifiable into mP. Similarly the goal $(q(t_1, t_2, \ldots t_n); \alpha)$ is convertible into $(Q(T_1, T_2, \ldots T_n); \alpha)$, where $Q = SP_q$ and $T_i = ST_{t_i}$ for i = 1,...,n.

The modified fDATALOG program is evaluable as an ordinary fDATALOG program, so we get answer for the modified goal. Applying the decoding function we can decide the answers for the query over the fuzzy knowledge-base. The next algorithm gives the set of answers to the goal $(q(t_1, t_2, \ldots t_n); \alpha)$ by decoding the answers for the query $(Q(X_1, X_2, \ldots X_n); \alpha)$:

Algorithm 2.

```
Procedure decoding (Q, SP, ST, Φp)
  S : =set of answers for the query(Q; α)
  Answers := ∅
  while not(empty(S)) do
     (Q(T1,T2,…,Tn);α):=first element of S
     Answers:=Answers ∪
             decoded((Q(T1,T2,…,Tn); α))
     S:= S - {(Q(T1,T2,…,Tn); α)}
  endwhile
endprocedure

function decoded((Q(T1,T2,…,Tn); α))
  Decoded_set := ∅
  while not(empty(Q) do
     (q , λq) := first element of Q
     q_set := ∅
     for each (ti , λti) ∈ Ti do
       q_set:= q_set ∪
  {(q(t1,t2,…,tn); φq(α, λq, λt1 ,…, λtn))}
     endfor
     Q := Q - {(q , λq)}
     Decoded_set := Decoded_set ∪ q_set
  endwhile
endprocedure
```

Easily can be seen, that if Answers is the set of evaluated goals by Algorithm 2., then

$$Answers \subseteq C(Bk, P, \Phi_P).$$

Example 4.

Let us see the next program:

$p(a) \leftarrow ; I_1; 0.8$

$p(b) \leftarrow ; I_2; 0.7$

$r(c) \leftarrow ; I_3; 0.6$

$q(x,y) \leftarrow p(x), r(y) ; I_4; 0.7$

$q(x,y) \leftarrow q(y,x) ; I_5; 0.8$

$s(x) \leftarrow q(x,y) ; I_6; 0.9$

Now let all of the implication operators be the Gödelian, denoted by I. The background knowledge is given as follows:

SP =

	p	q	r	s
p	1	0.8		
q	0.8	1	0.7	
r		0.7	1	
s				1

ST =

	a	b	c
a	1		0.9
b		1	
c	0.9		1

Be the goal: (q(x,y); α).

The decoding functions are defined as:

$$\varphi_p(\alpha, \lambda, \lambda_1) = \min(\alpha, \lambda, \lambda_1);$$
$$\varphi_q(\alpha, \lambda, \lambda_1, \lambda_2) = \alpha \cdot \lambda \cdot \min(\lambda_1, \lambda_2);$$
$$\varphi_r(\alpha, \lambda, \lambda_1) = \min(\alpha, \lambda \cdot \lambda_1);$$
$$\varphi_s(\alpha, \lambda, \lambda_1) = \alpha \cdot \lambda \cdot \lambda_1$$

The necessary part of modified fDATALOG for (Q(X,Y); α) goal is:

$P(A) \leftarrow ; I; 0.8$
$P(B) \leftarrow ; I; 0.7$
$R(C) \leftarrow ; I; 0.6$
$Q(X,Y) \leftarrow P(X), R(Y) ; I; 0.7$
$Q(X,Y) \leftarrow Q(Y,X) ; I; 0.8,$

where

$P = \{(p, 1), (q, 0.8)\}$; $Q = \{(q, 1), (r, 0.7), (p, 0.8)\}$;
$R = \{(r, 1), (q, 0.7)\}$;
$A = \{(a, 1), (c, 0.9)\}$; $B = \{(b, 1)\}$; $C = \{(c, 1), (a, 0.9)\}$.

The answer for this query is:

{(Q(A,C), 0.6), (Q(B,C), 0.6),
(Q(C,A), 0.6), (Q(C,B), 0.6)}

Applying the decoding algorithm, (Q(A,C), 0.6) can be decoded by the decoding function $\varphi_q(\alpha, \lambda, \lambda_1, \lambda_2)$:

{(q(a,c), 0.6), (q(a,a), $0.54 = \varphi_q(0.6, 1, 1, 0.9) = 0.6 \cdot 1 \cdot 0.9$), (q(c,a), 0.54), (q(c,c), 0.54), (r(a,c), $0.42 = 0.6 \cdot 0.7 \cdot 1$), (r(a,a), $0.378 = 0.6 \cdot 0.7 \cdot 0.9$), … etc.}

This example shows, that even a case of simple knowledge-base and a simple query, we can obtain a huge mass of result facts.

To reduce the number of result facts, we can extend the query with the levels of desirable similarities, so the wanted goal is:

$$(q(t_1, t_2, \ldots t_n); \alpha; \lambda_P; \lambda_T),$$

where $q(t_1, t_2, \ldots t_n)$ is the goal-atom, α is the uncertainty level of the goal (constant or variable), λ_P, λ_T are the acceptable similarity degree of predicates and terms respectively. Evaluating this goal – in case of given λ_P, λ_T – in the modified fDATALOG the similarity sets are replaced by their λ_P or λ_T –cuts.

IV. SUMMARY

In this paper we gave a possible model of handling uncertain information: we defined the concept of fuzzy knowledge-base by the triplet of a background knowledge, a fuzzy DATALOG program, and a decoding set. The background knowledge shows the similarities between predicates and terms, the program makes a fuzzy deduction, and the decoding functions compute the uncertainty level of the results.

The huge mass of result facts raises the question of a more effective evaluation strategy. Now we are working out one kind of the possible ones.

V. REFERENCES

[1] Ágnes Achs - Attila Kiss: Fixpoint query in fuzzy Datalog, Annales Univ. Sci. Budapest, Sect. Comp. 15 (1995) (223-231)

[2] Ágnes Achs - Attila Kiss : Fuzzy extension of Datalog, Acta Cybernetica 12 (1995), Szeged (153-166)

[3] Ágnes Achs: Evaluation Strategies of Fuzzy Datalog, Acta Cybernetica 13 (1997), Szeged (85-102)

[4] F.Arcelli, F.Formato and G.Gerla, "Fuzzy Unification as Foundations of Fuzzy Logic programming" in Logic Programming and Soft Computing, Ed. RSP-Wiley, England, 1998.

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

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

[7] Vilém Novák : Fuzzy sets and their applications, Adam Hilger Bristol and Philadelphia, 1989.

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

[9] J.D.Ullman : Principles of database and knowledge-base systems, Computer Science Press, Rockville, 1988.

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