2

From Fuzzy Datalog to Multivalued Knowledge-Base

Agnes Achs
University of Pecs Faculty of Engineering, Hungary

1. Introduction

Despite the fact that people have very different and ambiguous concepts and knowledge, they are able to talk to one another. How does human mind work? How can people give answers to questions? Modelling human conversation and knowledge demands to deal with uncertainty and deductions.

Human knowledge consists of static and dynamic knowledge chunks. The static ones include the so called lexical knowledge or the ability to sense similarities between facts and between predicates. Through dynamic attainments one can make deductions or one can give answers to a question. There are several and very different approaches to make a model of human knowledge, but one of the most common and widespread fields of research is based on fuzzy logic.

Fuzzy sets theory, proposed by Zadeh (1965), is a realistic and practical means to describe the world that we live in. The method has successfully been applied in various fields, among others in decision making, logic programming, and approximate reasoning. In the last decade, a number of papers have dealt with that subject, e.g. (Formato et al 2000, Sessa 2002, Medina et al 2004, Straccia et al 2009). They deal with different aspects of modelling and handling uncertainty. (Straccia 2008) gives a detailed overview of this topic with widespread references. Our investigations have begun independently of these works, and have run parallel to them. Of course there are some similar features, but our model differs from the others detailed in literature.

As a generalization of fuzzy sets, intuitionistic fuzzy sets were presented by Atanassov (Atanassov 1983), and have allowed people to deal with uncertainty and information in a much broader perspective. Another well-known generalization of an ordinary fuzzy set is the interval-valued fuzzy set, which was first introduced by Zadeh (Zadeh 1975). These generalizations make descriptions and models of the world more realistic, and practical.

In the beginning, our knowledge-base model was based on the concept of fuzzy logic, later on it was extended to intuitionistic and interval-valued logic. In this model, the static part is a background knowledge module, while the dynamic part consists of a Datalog based deduction mechanism. To develop this mechanism, it was necessary to generalize the Datalog language and to extend it into fuzzy and intuitionistic direction. (Achs 1995, 2007, 2010).

In many frameworks, in order to answer a query, we have to compute the whole intended model by a bottom-up fixed-point computation and then answer with the evaluation of the query in this model. This always requires computing a whole model, even if not all the facts and rules are required to determine answer. Therefore a possible top-down like evaluation algorithm has been developed for our model. This algorithm is not a pure top-down one but the combination of top down and bottom up evaluations. Our aim is to improve this algorithm and perhaps to develop a pure top down evaluation based on fuzzy or multivalued unification algorithm. There are fuzzy unification algorithms described for example in (Alsinet et al 1998, Formato et al 2000, Virtanen 1994), but they are inappropriate for evaluating our knowledge-base.

However, the concept of (Julian-Iranzo et al 2009, 2010) is similar but not identical with one of our former ideas about evaluating of special fuzzy Datalog programs (Achs 2006). Reading these papers has led to the assumption that this former idea may be the base of a top-down-like evaluation strategy in special multivalued cases as well. Based on this idea, a multivalued unification algorithm was developed and used for to determine the conclusion of a multivalued knowledge-base.

In this chapter this possible model for handling uncertain information will be provided. This model is based on the multivalued extensions of Datalog. Starting from fuzzy Datalog, the concept of intuitionistic Datalog and bipolar Datalog will be described. This will be the first pillar of the knowledge-base. The second one deals with the similarities of facts and concepts. These similarities are handled with proximity relations. The third component connects the first two with each other. In the final part of the paper, an evaluating algorithm is presented. It is discussed in general, but in special cases it is based on fuzzy, or multivalued unification, which is also mentioned.

2. Extensions of datalog

When one builds a knowledge-base, it is very important to deal with a database management system. It is based on the relational data model developed by Codd in 1970. This model is a very useful one, but it can not handle every problem. For example, the standard query language for relational databases (SQL) is not Turing-complete, in particular it lacks recursion and therefore concepts like transitive closure of a relation can not be expressed in SQL. Along with other problems this is why different extensions of the relational data model or the development of other kinds of models are necessary. A more complete one is the world of deductive databases. A deductive database consists of facts and rules, and a query is answered by building chains of deductions. Therefore the term of deductive database highlights the ability to use a logic programming style for expressing deductions concerning the contents of a database. One of the best known deductive database query languages is Datalog.

As any deductive database, a Datalog program consists of facts and rules, which can be regarded as first order logic formulas. Using these rules, new facts can be inferred from the program's facts so that the consequence of a program will be logically correct. This means that evaluating the program, the result is a model of the formulas belonging to the rules. On the other hand, it is also important that this model will contain only those true facts, which are the consequences of the program; that is, the minimality of this model is expected, i.e. in

this model it is impossible to make any true fact false and still have a model consistent with the database.

An interpretation assigns truth or falsehood to every possible instance of the program's predicates. An interpretation is a model, if it makes the rules true, no matter what assignment of values from the domain is made for the variables in each rule. Although there are infinite many implications, it is proved that it is enough to consider only the Herbrand interpretation defined on the Herbrand universe and the Herbrand base.

The Herbrand universe of a program P (denoted by H_P) is the set of all possible ground terms constructed by using constants and function symbols occurring in P. The Herbrand base of P (B_P) is the set of all possible ground atoms whose predicate symbols occur in P and whose arguments are elements of H_P.

In general, a term is a variable, a constant or a complex term of the form $f(t_1, \ldots, t_n)$, where f is a function symbol and $t_1, \ldots, t_n$ are terms. An atom is a formula of the form $p(\underline{t})$, where p is a predicate symbol of a finite arity (say n) and $\underline{t}$ is a sequence of terms of length n (arguments). A literal is either an atom (positive literal) or its negation (negative literal). A term, atom or literal is ground if it is free of variables. As in fuzzy extension, we did not deal with function symbols, so in our case the ground terms are the constants of the program.

In the case of Datalog programs there are several equivalent approaches to define the semantics of the program. In fuzzy extension we mainly rely on the fixed-point base aspect. The above concepts are detailed in classical works such as (Ceri et al 1990, Loyd 1990, Ullman 1988).

2.1 Fuzzy Datalog

In fuzzy Datalog (fDATALOG) the facts can be completed with an uncertainty level, the rules with an uncertainty level and an implication operator. With the use of this operator and these levels deductions can be made. As in classical cases, logical correctness is extremely important as well, i.e., the consequence must be a model of the program. This means that for each rule of the program, the truth-value of the fuzzy implication following the rule has to be at least as large as the given uncertainty level.

2.1.1 Syntax and semantics of fuzzy datalog

More precisely, the notion of fuzzy rule is the following:

Definition 1. An fDATALOG rule is a triplet $r; \beta; I$, where r is a formula of the form

$$A \leftarrow A_1,\ldots,A_n \qquad (n \geq 0),$$

where A is an atom (the head of the rule), $A_1,\ldots,A_n$ are literals (the body of the rule); I is an implication operator and $\beta \in (0,1]$ (the level of the rule).

For getting a finite result, all the rules in the program must be safe. 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 one. 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 $A \leftarrow; \beta; I$. From now on, the facts are referred as (A,β), because according to implication I, the level of A easily can be computed and in the case of the implication operators detailed in this chapter it is β.

For defining the meaning of a program, we need again the concepts of Herbrand universe and Herbrand base, but this time they are based on fuzzy logic. Now a ground instance of a rule $r; \beta; I$ in P is a rule obtained from r by replacing every variable in r with a constant of H_P. The set of all ground instances of $r; \beta; I$ is denoted by $ground(r); \beta; I$. The ground instance of P is $ground(P) = \cup_{(r;\, I;\, \beta) \in P} (ground(r); I; \beta)$.

An interpretation of a program P is a fuzzy set of the program's Herbrand base, B_P, i.e. it is: $\cup_{A \in BP} (A; \alpha_A)$. An interpretation is a model of P if for each $(A \leftarrow A_1,\ldots,A_n; \beta; I) \in ground(P)$

$$I(\alpha_{A1 \wedge \ldots \wedge An}, \alpha_A) \geq \beta.$$

A model M is least if for any model N, $M \leq N$. A model M is minimal if there is not any model N, where $N \leq M$.

To be short $\alpha_{A1 \wedge \ldots \wedge An}$ will be denoted as α_{body} and α_A as α_{head}.

In the extensions of Datalog several implication operators are used, but all cases are restricted to min-max conjunction and disjunction, and to the complement to 1 as negation. So: $\alpha_{A \wedge B} = min(\alpha_A, \alpha_B)$, $\alpha_{A \vee B} = max(\alpha_A, \alpha_B)$ and $\alpha_{\neg A} = 1 - \alpha_A$.

The semantics of fDATALOG is defined as the fixed points of consequence transformations. Depending on evaluating sequences two semantics can be defined: a deterministic and a nondeterministic one. Further on only the nondeterministic semantics will be discussed, the deterministic one is detailed in (Achs 2010). It was proved that the two semantics are equivalent in the case of negation- and function-free fDatalog programs, but they differ if the program has any negation. In this case merely the nondeterministic semantics is applicable. The nondeterministic transformation is as follows:

Definition 2. Let B_P be the Herbrand base of the program P, and let $F(B_P)$ denote the set of all fuzzy sets over B_P. The consequence transformation $NT_P : F(B_P) \rightarrow F(B_P)$ is defined as

$$NT_P(X) = \{(A, \alpha_A)\} \cup X, \tag{1}$$

where

$$(A \leftarrow A_1,\ldots,A_n; \beta; I) \in ground(P), (|A_i|, \alpha_{Ai}) \in X, (1 \leq i \leq n);$$

$$\alpha_A = max(0, min\{\gamma \mid I(\alpha_{body}, \gamma) \geq \beta\}).$$

$|A_i|$ 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 $\neg A_i$, if A_i is negative) and $\alpha_{body} = min(\alpha_{A1}, \ldots, \alpha_{An})$.

It can be proved that this transformation has a fixed point. To prove it, let us define the powers of a transformation:

For any $T : F(B_P) \rightarrow F(B_P)$ transformation let

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

and let

$$T_1 = T(T_0)$$

$$T_n = T(T_{n-1})$$

...

T_ω = least upper bound of $\{ T_n \mid n < \omega \}$ if ω is a limit ordinal.

An ordering relation can be defined over $F(B_P)$. For $G; H : B_P \to [0; 1]$; $G \leq H$ iff $(\forall d \in B_P)$ $G(d) \leq H(d)$. It easily can be seen that $L = (F(B_P), \leq)$ is a complete lattice.

It is clear that NT_P is inflationary transformation over L, and if P is negation-free, then NT_P is monotone as well. (A transformation T is inflationary if $X \leq T(X)$ for every $X \in L$ and it is monotone if $T(X) \leq T(Y)$ if $X \leq Y$).

In (Ceri et al 1990) it is shown that an inflationary transformation over a complete lattice has a fixed point and if it is monotone then it has a least fixed point (Loyd 1990). Therefore NT_P has a fixed point, i.e. there exists an $X \in F(B_P)$ for which $NT_P(X) = X$. If P is positive, then X is the least fixed point. (That is for any $Z=T(Z) : X \leq Z$.)

The fixed point of the transformation will be denoted by $lfp(NT_P)$. It can be shown (Achs 1995) that this fixed point is a model of P.

Theorem 1. $lfp(NT_P)$ is a model of P.

Proof In $ground(P)$ there are rules in the next forms:

a/ $(A \leftarrow; \beta; I)$.
b/ $(A \leftarrow A_1, ..., A_n; \beta; I); (A, \alpha_A) \in lfp(NT_P)$ and $(|A_i|, \alpha_{A_i}) \in lfp(NT_P)$, $1 \leq i \leq n$.
c/ $(A \leftarrow A_1, ..., A_n; \beta; I); \exists i : (|A_i|, \alpha_{A_i}) \notin lfp(NT_P)$.

In the case of a, b because of the construction of α_A the condition $I(\alpha_{body}, \alpha_A) \geq \beta$ is realized. In the case of c because of the construction of T_0 A_i is not negative, that is A_i is not among the facts, so $\alpha_{A_i} = 0$, therefore $\alpha_{body} = 0$, so $I(\alpha_{body}, \alpha_A) = 1 \geq \beta$, namely $lfp(NT_P)$ is a model.

Moreover, the next proposition is true as well.

Proposition 1. For negation-free fDATALOG program P $lfp(NT_P)$ is the least model.

Proof In the case of a positive Datalog program, the least fixed point is the least model (Ceri et al 1990, Ullman 1988). In the case of fuzzy Datalog, according to the definition of the consequence transformation, the level of the rule's head is the least value satisfying the criterion of modelness. The application of the transformations may arise only one problem. A lower level would be ordered to the same rule's head, but according to the definitions we should accept the higher value. But such a case can arise only in the case of programs containing any negation. Therefore the proposition is true.

According to the above statements, the meaning of the programs can be defined by this fixed point:

Definition 3. $lfp(NT_P)$ is the nondeterministic semantics of fDATALOG P program.

To compute the level of rule-heads, we need the concept of uncertainty level function.

Definition 4. The uncertainty-level function is:

$$f(I, \alpha, \beta) = min(\{ \gamma \mid I(\alpha, \gamma) \geq \beta \}).$$

According to this function the level of a rule-head is: $\alpha_{head} = f(I, \alpha_{body}, \beta)$.

It is an extremely important question whether the fixed-point algorithm terminates or not. It depends on the feature of uncertainty level function:

Proposition 2. If $f(I, \alpha, \beta) \leq \alpha$ for $\forall \alpha \in [0; 1]$ then the fixed point algorithm terminates.

Proof As P is finite, therefore in the fixed point there are only finite many ground atoms. The only problem may occur with the level of recursive predicates, but according to the above property of the uncertainty-level function, the level of the rule's head cannot be greater than any former one, so this algorithm must terminate.

In former papers (Achs 1995, Achs 2006) several implications were detailed (the operators are detailed in (Dubois et al, 1991)), for now three are chosen from these. The values of their uncertainty-level functions can be easily computed. They are the following:

Gödel	$I_G(\alpha,\gamma) = \begin{cases} 1 & \alpha \leq \gamma \\ \gamma & otherwise \end{cases}$	$f(I_G, \alpha, \beta) = min(\alpha, \beta)$
Lukasiewicz	$I_L(\alpha,\gamma) = \begin{cases} 1 & \alpha \leq \gamma \\ 1-\alpha+\gamma & otherwise \end{cases}$	$f(I_L, \alpha, \beta) = max(0, \alpha + \beta - 1)$
Kleene-Dienes	$I_K(\alpha,\gamma) = max(1-\alpha, \gamma)$	$f(I_K, \alpha, \beta) = \begin{cases} 0 & \alpha + \beta \leq 1 \\ \beta & \alpha + \beta > 1 \end{cases}$

It is obvious that I_G and I_L satisfy the condition of Proposition 2, and it is easy to see that in the case of I_K the fixed point algorithm terminates as well. (Among the operators of (Dubois et al, 1991) there is one for which the algorithm does not terminate and one for which the uncertainty-level function does not exists.)

Example 1. Let us consider the next program:

$(p(a), 0.8).$
$(r(b), 0.6).$
$q(x, y) \leftarrow p(x), r(y); 0.7; I_G.$
$q(x, y) \leftarrow q(y, x); 0.9; I_L.$
$s(x) \leftarrow q(x, y); 0.7; I_K.$

Then $T_0 = \{(p(a), 0.8), (r(b), 0.6)\}$ and the computed atoms are:

$(q(a, b), min(min(0.8, 0.6), 0.7) = 0.6);$
$(q(b, a), max(0, 0.6 + 0.9 - 1) = 0.5);$

$$(s(a), \begin{cases} 0 & 0.6 + 0.7 \leq 1 \\ 0.7 & 0.6 + 0.7 > 1 \end{cases} = 0.7);$$

$$(s(b), \begin{cases} 0 & 0.5 + 0.7 \leq 1 \\ 0.7 & 0.5 + 0.7 > 1 \end{cases} = 0.7);$$

So $lfp(NT_P) = \{(p(a), 0.8), (r(b), 0.6), (q(a, b), 0.6), (q(b, a), 0.5), (s(a), 0.7), (s(b), 0.7)\}$.

As the next examples show, there some problems would arise if the program had any negation.

Example 2. Look at the next one-rule program:

$$p(a) \leftarrow \neg q(b); I_G; 0.7.$$

This program has no least model, only two minimal ones:

$$M_1 = \{(p(a), 0.7)\} \text{ and } M_2 = \{(q(b), 1)\}.$$

(The result of the above fixed point algorithm is M_1.)

Example 3. This example shows that there is a difference between the fixed points.

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

The result depends on the evaluation order. If it is 1., 2., 3., 4., then

$$lfp(NT_P) = \{(r(a), 0.8), (p(a), 0.6), (q(a), 0.5)\},$$

while in the order 1., 3., 2., 4.

$$lfp(NT_P) = \{(r(a), 0.8), (p(a), 0.5), (q(a), 0.5)\}.$$

According to the above examples, in the case of programs containing negation there are problems with the model's minimality. However, the nondeterministic semantics – $lfp(NT_P)$ – is minimal under certain conditions. These conditions are referred to as stratification. Stratification gives an evaluating sequence in which the literals are evaluated before negating them.

2.1.2 Stratified fuzzy datalog

To stratify a program, it is necessary to define the concept of dependency graph. This is a directed graph, whose nodes are the predicates of P. There is an arc from predicate p to predicate q if there is a rule whose body contains p or $\neg p$ and whose head predicate is q. A program is recursive, if its dependency graph has one or more cycles. A program is stratified if whenever there is a rule with head predicate p and a negated body literal $\neg q$, there is no path in the dependency graph from p to q.

The stratification of a program P is a partition of the predicate symbols of P into subsets $P_1,..., P_n$ such that the following conditions are satisfied:

a/ if $p \in P_i$ and $q \in P_j$ and there is an edge from q to p then $i \geq j$;

b/ if $p \in P_i$ and $q \in P_j$ and there is a rule with the head p whose body contains $\neg q$, then $i > j$.

Stratification specifies an order of evaluation. The rules whose head-predicates are in P_1 are evaluated first, then those whose head-predicates are in P_2 and so on. The sets $P_1,..., P_n$ are

called the strata of the stratification. A program P is called stratified if and only if it admits stratification. There is a very simple method for finding stratification for a stratified program in (Ceri et al 1990, Ullman 1988). Because this algorithm groups the predicates of the program, this is suitable for the fDATALOG programs as well.

The first of the following Datalog programs is not stratified, the other one has more distinct stratifications.

Example 4. Consider the one-rule program:

$$p(x) \leftarrow \neg p(x).$$

This is not stratified.

The next program has more stratification (Abiteboul et al 1995):

1. $s(x) \leftarrow r_1(x), \neg r(x).$
2. $t(x) \leftarrow r_2(x), \neg r(x).$
3. $u(x) \leftarrow r_3(x), \neg t(x).$
4. $v(x) \leftarrow r_4(x), \neg s(x), \neg u(x).$

The program has five distinct stratifications, namely:

{1.}, {2.}, {3.}, {4.}
{2.}, {1.}, {3.}, {4.}
{2.}, {3.}, {1.}, {4.}
{1., 2.}, {3.}, {4.}
{2.}, {1., 3.}, {4.}

These lead to five different ways of reading the program. As will be seen later, each of them yields the same semantics.

Let P be a stratified fDATALOG program with stratification $P_1,..., P_n$. Let P_i^* denote the set of all rules of P corresponding to stratum P_i , that is the set of all rules whose head-predicate is in P_i. Let

$$L_1 = lfp(NT_{P_1^*}),$$

where the starting point of the computation is T_0 defined earlier.

$$L_2 = lfp\ (NT_{P_2^*}),$$

where the starting point of the computing is L_1.

$$L_n = lfp\ (NT_{P_n^*}),$$

where the starting point is L_{n-1}.

In other words: the least fixed point - L_1 - corresponding to the first stratum of P is computed at first. Once this fixed point has been computed, we can take a step to the next strata.

By induction it will be shown that L_n is a minimal model of P. For this purpose, we need the next definition and lemma.

Definition 5. An fDATALOG program P is semi-positive if its negated predicates are solely facts.

Lemma 1. A semi-positive program P has a minimal model: $L = lfp(NT_P)$.

Proof. A semi-positive program is almost the same as a positive one, because if p is a negated predicate of a rule-body, then it can be replaced by the fact $q = \neg p$. As p is a fact predicate, therefore the uncertainty level of q may be easily calculated. So the negation can be eliminated from the program and this program has a least fixed point which is the least model.

According to the lemma, L_1 is the least fixed point for P_1^*. Generally $L_{i-1} \cup P_i^*$ is semi-positive, because according to the stratification each negative literal of the i-th strata belongs to a predicate of a lower level strata. So L_i is the least fixed point for P_i^*, which is minimal model for the given stratification. Therefore the next theorem is true:

Theorem 2. If P is a stratified fDATALOG program then L_n is a minimal model of P.

This means that evaluating the rules in the order of stratification, the least fixed point of the program's nondeterministic transformation is the minimal model of the program as well. So:

Proposition 3. For stratified fDATALOG program P, there is an evaluation sequence, in which $lfp(NT_P)$ is a minimal model of P.

As shown in Example 4, a program can have more then one stratification. Will the different stratifications yield the same semantics? Fortunately, the answer is yes. (Ceri et al 1990) declares, (Abiteboul et al 1995) proves the theorem, according to which for stratified Datalog programs the resulting minimal model is independent of the actual stratification. That is, two stratifications of a Datalog program yield the same semantics on all inputs. As the order of stratification depends only on the predicates of the program and it is not influenced by the uncertainty levels, therefore this theorem is true in the case of fDATALOG programs as well.

Theorem 3. Let P be a stratifiable fDATALOG program. The least fixed point according to an arbitrary order of stratification is a unique minimal model of the program.

Example 5. In Example 3. the right stratified order is 1., 3., 2., 4.; so the least fixed point of the program is: $lfp(NT_P) = \{(r(a), 0.8), (p(a), 0.5), (q(a), 0.5)\}$.

2.2 Multivalued datalog

In fuzzy theory, uncertainty is measured by a single value between zero and one, and negation can be calculated as its complement to 1. However, human beings sometimes hesitate expressing these values, that is, there may be some hesitation degree. This illuminates a well-known psychological fact that linguistic negation does not always correspond to the logical one. Based on this observation, as a generalization of fuzzy sets, the concept of intuitionistic fuzzy sets was introduced and developed by Atanassov in 1983 and later (Atanassov 1983, 1999, Atanassov & Gargov 1989). In the next paragraphs some possible multivalued extensions of Datalog will be discussed.

2.2.1 Intuitionistic- and interval-valued extensions of datalog

In intuitionistic fuzzy systems (IFS) and interval-valued systems (IVS) the uncertainty is represented by two values, $\underline{\mu} = (\mu_1, \mu_2)$ instead of a single one. In the intuitionistic case the two elements must satisfy the condition $\mu_1+\mu_2 \leq 1$, while in the interval-valued case the condition is $\mu_1 \leq \mu_2$. If $\underline{\mu} = (\mu_1, \mu_2)$ belonging to a predicate p is an IFS level, then p is definitely true on level μ_1 and definitely false on level μ_2, while in IVS the truth value is between μ_1 and μ_2. It is obvious that the relation $\mu'_1 = \mu_1$, $\mu'_2 = 1 - \mu_2$ creates a mutual connection between the two systems. (The equivalence of IVS and IFS was stated first in (Atanassov & Gargov 1989).)

The fixed point theory of programming is based on the theory of lattices. So does the theory of fuzzy Datalog as well, which is based on the lattice of fuzzy sets. The extension of the programs into an intuitionistic and interval-valued direction needs the extension of lattices as well.

Definition 6. L_F and L_V are lattices of IFS and IVS respectively, where

$$L_F = \{(x_1, x_2) \in [0,1]^2 \mid x_1+x_2 \leq 1\}, (x_1,x_2) \leq_F (y_1,y_2) \Leftrightarrow x_1 \leq y_1 \text{ and } x_2 \geq y_2,$$

$$L_V = \{(x_1, x_2) \in [0,1]^2 \mid x_1 \leq x_2\}, (x_1,x_2) \leq_V (y_1,y_2) \Leftrightarrow x_1 \leq y_1 \text{ and } x_2 \leq y_2$$

It can be proved that both L_F and L_V are complete lattices (Cornelis et al 2004), so it can be the base of intuitionistic Datalog (ifDATALOG) and interval-valued Datalog (ivDATALOG) as well. (If the distinction is not important, both of them will be denoted by iDATALOG.)

The so called i-extended DATALOG is defined on these lattices, and the necessary concepts are generalizations of the ones presented in Definition 1 and Definition 2. Let us continue to denote by B_P the Herbrand base of the program P, and let $FV(B_P)$ the set of all IFS or IVS sets over B_P.

Definition 7. The i-extended Datalog program (iDATALOG) is a finite set of safe iDATALOG rules r; $\underline{\beta}$; $\underline{I}_{FV}$;

- the i-extended consequence transformation $iNT_P : FV(B_P) \rightarrow FV(B_P)$ is formally the same as NT_P in (1) except:

$$\underline{\alpha}_A = max_{FV} (\underline{0}_{FV}, min_{FV}\{\underline{\gamma} \mid \underline{I}_{FV}(\underline{\alpha}_{body}, \underline{\gamma}) \geq_{FV} \underline{\beta}\});$$

- the i-extended uncertainty-level function is

$$\underline{f}(\underline{I}_{FV}, \underline{\alpha}, \underline{\beta}) = min_{FV}(\{\underline{\gamma} \mid \underline{I}_{FV}(\underline{\alpha}, \underline{\gamma}) \geq_{FV} \underline{\beta}\}),$$

where $\underline{\alpha}$, $\underline{\beta}$, $\underline{\gamma}$ are elements of L_F, L_V respectively, $\underline{I}_{FV} = \underline{I}_F$ or $\underline{I}_V$ is an implication of L_F or L_V; $max_{FV} = max_F$ or max_V; $min_{FV} = min_F$ or min_V are the max or min operator of L_F or L_V; $\underline{0}_{FV}$ is $\underline{0}_F = (0,1)$ or $\underline{0}_V = (0,0)$ and $\geq_{FV}$ is $\geq_F$ or $\geq_V$.

As iNT_P is inflationary transformation over the complete lattices L_F or L_V, thus according to (Ceri et al 1990) it has an inflationary fixed point denoted by $lfp(iNT_P)$. If P is positive (without negation), iNT_P is a monotone transformation, so $lfp(iNT_P)$ is the least fixed point.

The next question is whether this fixed point is a model of P. The fixed point is an interpretation of P, which is a model, if for each

$$A \leftarrow A_1, \ldots, A_n; \underline{\beta}; \underline{I}_{FV} \in ground(P), \underline{I}_{FV}(\underline{\alpha}, \underline{\gamma}) \geq_{FV} \underline{\beta}.$$

Similarly to the proof of Theorem 1, it can easily be proved that this fixed point is a model of the program. For negation-free iDATALOG this is the least model of the program. (Achs 2010).

In fDATALOG a fact can be negated by completing its membership degree to 1. In iDATALOG the uncertainty level of a negated fact can be computed according to negators. A negator on L_F or L_V is a decreasing mapping ordering $\underline{0}_{FV}$ and $\underline{1}_{FV}$ together (Cornelis et al 2004). The applied negators are relevant for the computational meaning of a program, but they have no influence on the stratification. So for a stratified iDATALOG program P there is an evaluation sequence in which $lfp(iNT_P)$ is a unique minimal model of P. Therefore $lfp(iNT_P)$ can be regarded as the semantics of iDATALOG.

After defining the syntax and semantics of extended fuzzy Datalog, it is necessary to examine the properties of possible implication operators and the extended uncertainty-level functions. A number of intuitionistic implications are discussed in (Cornelis et al 2004, Atanassov 2005, 2006) and other papers, four of them are the extensions of the above three fuzzy implication operators. Now these operators will be presented and completed by the suitable interval-value operators and the uncertainty-level functions. The computations will not be shown here, only the starting points and results are presented.

The coordinates of intuitionistic and interval-valued implication operators can be determined by each other. The uncertainty-level functions can be computed according to the applied implication. The connection between $\underline{I}_F$ and $\underline{I}_V$ and the extended versions of uncertainty-level functions are given below: For

$$\underline{I}_V(\underline{\alpha},\underline{\gamma}) = (I_{V1}, I_{V2});$$

$$I_{V1}=I_{F1}(\underline{\alpha}',\underline{\gamma}); \qquad \underline{\alpha}'=(\alpha_1,1-\alpha_2);$$

$$I_{V2}=1-I_{F2}(\underline{\alpha}',\underline{\gamma}'); \qquad \underline{\gamma}'=(\gamma_1,1-\gamma_2).$$

$$\underline{f}\,(\underline{I}_F, \underline{\alpha}, \underline{\beta}) = (min\,(\{\,\gamma_1 \mid I_{F1}\,(\underline{\alpha}, \underline{\gamma}) \geq \beta_1\,\}), max\,(\{\,\gamma_2 \mid I_{F2}\,(\underline{\alpha}, \underline{\gamma}) \leq \beta_2\,\}))$$

$$\underline{f}\,(\underline{I}_V, \underline{\alpha}, \underline{\beta}) = (min\,(\{\,\gamma_1 \mid I_{V1}\,(\underline{\alpha}, \underline{\gamma}) \geq \beta_1\,\}), min\,(\{\,\gamma_2 \mid I_{V2}\,(\underline{\alpha}, \underline{\gamma}) \geq \beta_2\,\}))$$

The studied operators and the related uncertainty-level functions are the following:

2.2.1.1 Extension of Kleene-dienes implication

One possible extension of Kleene-Dienes implication for IFS is:

$$\underline{I}_{FK}(\underline{\alpha}, \underline{\gamma}) = (max(\alpha_2, \gamma_1), min(\alpha_1, \gamma_2))$$

The appropriate computed elements are the following:

$$\underline{I}_{VK}(\underline{\alpha}, \underline{\gamma}) = (max(1-\alpha_2, \gamma_1), max(1-\alpha_1, \gamma_2))$$

$$f_1(\underline{I}_{FK}, \underline{\alpha}, \underline{\beta}) = \begin{cases} 0 & \alpha_2 \geq \beta_1 \\ \beta_1 & \text{otherwise} \end{cases} \qquad f_2(\underline{I}_{FK}, \underline{\alpha}, \underline{\beta}) = \begin{cases} 1 & \alpha_1 \leq \beta_2 \\ \beta_2 & \text{otherwise} \end{cases}$$

$$f_1(\underline{I}_{VK}, \underline{\alpha}, \underline{\beta}) = \begin{cases} 0 & 1-\alpha_2 \geq \beta_1 \\ \beta_1 & \text{otherwise} \end{cases} \qquad f_2(\underline{I}_{VK}, \underline{\alpha}, \underline{\beta}) = \begin{cases} 0 & 1-\alpha_1 \geq \beta_2 \\ \beta_2 & \text{otherwise} \end{cases}$$

2.2.1.2 Extension of Lukasiewicz implication

One possible extension of Lukasiewicz implication for IFS is:

$$\underline{I}_{FL}(\underline{\alpha}, \underline{\gamma}) = (\min(\alpha_2, \gamma_1), \min(\alpha_1, \gamma_2))$$

The appropriate computed elements are as follows:

$$\underline{I}_{VL}(\underline{\alpha}, \underline{\gamma}) = (\max(1, \alpha_2 + \gamma_1), \max(0, \alpha_1 + \gamma_2 - 1))$$

$$f_1(\underline{I}_{FL}, \underline{\alpha}, \underline{\beta}) = \min(1-\alpha_2, \max(0, \beta_1-\alpha_2))$$

$$f_2(\underline{I}_{FL}, \underline{\alpha}, \underline{\beta}) = \max(1-\alpha_1, \min(1, 1-\alpha_1+\beta_2))$$

$$f_1(\underline{I}_{VL}, \alpha, \beta) = \max(0, \alpha_2+\beta_1-1)$$

$$f_2(\underline{I}_{VL}, \alpha, \beta) = \max(0, \alpha_1+\beta_2-1)$$

2.2.1.3 Extensions of Gödel implication

There are several alternative extensions of Gödel implication, two of them are presented here:

$$\underline{I}_{FG1}(\underline{\alpha}, \underline{\gamma}) = \begin{cases} (1,0) & \alpha_1 \leq \gamma_1 \\ (\gamma_1, 0) & \alpha_1 > \gamma_1, \alpha_2 \geq \gamma_2 \\ (\gamma_1, \gamma_2) & \alpha_1 > \gamma_1, \alpha_2 < \gamma_2 \end{cases} \qquad \underline{I}_{FG2}(\underline{\alpha}, \underline{\gamma}) = \begin{cases} (1,0) & \alpha_1 \leq \gamma_1, \alpha_2 \geq \gamma_2 \\ (\gamma_1, \gamma_2) & \text{otherwise} \end{cases}$$

The appropriate computed elements are:

$$\underline{I}_{VG1}(\underline{\alpha}, \underline{\gamma}) = \begin{cases} (1,1) & \alpha_1 \leq \gamma_1 \\ (\gamma_1, 1) & \alpha_1 > \gamma_1, \alpha_2 \geq \gamma_2 \\ (\gamma_1, \gamma_2) & \alpha_1 > \gamma_1, \alpha_2 < \gamma_2 \end{cases} \qquad \underline{I}_{VG2}(\underline{\alpha}, \underline{\gamma}) = \begin{cases} (1,1) & \alpha_1 \leq \gamma_1, \alpha_2 \leq \gamma_2 \\ (\gamma_1, \gamma_2) & \text{otherwise} \end{cases}$$

$$f_1(\underline{I}_{FG1}, \underline{\alpha}, \underline{\beta}) = \min(\alpha_1, \beta_1) \qquad f_2(\underline{I}_{FG1}, \underline{\alpha}, \underline{\beta}) = \begin{cases} 1 & \alpha_1 \leq \beta_2 \\ \max(\alpha_2, \beta_2) & \text{otherwise} \end{cases}$$

$$f_1(\underline{I}_{VG1}, \underline{\alpha}, \underline{\beta}) = \min(\alpha_1, \beta_1) \qquad f_2(\underline{I}_{VG1}, \underline{\alpha}, \underline{\beta}) = \begin{cases} 0 & \alpha_1 \leq \beta_2 \\ \min(\alpha_2, \beta_2) & \text{otherwise} \end{cases}$$

$$f_1(\underline{I}_{FG2}, \underline{\alpha}, \underline{\beta}) = \min(\alpha_1, \beta_1) \qquad f_2(\underline{I}_{FG2}, \underline{\alpha}, \underline{\beta}) = \max(\alpha_2, \beta_2)$$

$$f_1(\underline{I}_{VG2}, \underline{\alpha}, \underline{\beta}) = \min(\alpha_1, \beta_1) \qquad f_2(\underline{I}_{VG2}, \underline{\alpha}, \underline{\beta}) = \min(\alpha_2, \beta_2)$$

An important question is whether the resulting degrees satisfy the conditions referring to IFS and IVS respectively. Unfortunately, for implications other than G_2, the answer is "no",

or rather "not in all cases". For example, in the case of the Kleene-Dienes and the Lukasiewicz intuitionistic operators the levels of a rule-head satisfy the condition of intuitionism only if the sum of the levels of the rule-body is at least as large as the sum of the levels of the rule. That is, the solution is inside the scope of IFS, if the level of the rule-body is less "intuitionistic" than the level of the rule. In the case of the first Gödel operator, the solution is inside the scope of IFS only if the level of the rule-body is more certain than the level of the rule (Achs 2010). However, for the second Gödel operator the next proposition can easily be proven:

Proposition 4. For $\underline{\alpha} = (\alpha_1, \alpha_2)$, $\underline{\beta} = (\beta_1, \beta_2)$

$$if\ \alpha_1+\alpha_2 \le 1,\ \beta_1+\beta_2 \le 1\ then\ f_1(\underline{I}_{FG2}, \underline{\alpha}, \underline{\beta}) + f_2(\underline{I}_{FG2}, \underline{\alpha}, \underline{\beta}) \le 1$$

$$if\ \alpha_1 \le \alpha_2,\ \beta_1 \le \beta_2\ then\ f_1(\underline{I}_{VG2}, \underline{\alpha}, \underline{\beta}) \le f_2(\underline{I}_{VG2}, \underline{\alpha}, \underline{\beta})$$

Similarly to Proposition 2, it can be seen that the fixed-point algorithm terminates if $\underline{f}(\underline{I}_{FV}, \underline{\alpha}, \underline{\beta}) \le_{FV} \underline{\alpha}$ for each $\underline{\alpha} \in L_{FV}$ (Achs 2010). G2 satisfies this condition, so:

Proposition 5. In the case of G2 operator the fixed-point algorithm terminates.

2.2.2 Bipolar extension of datalog

The intuitive background of intuitionistic levels is some psychological perception. Experiments have shown that when making decisions people deal with positive and negative facts in different ways (Dubois et al 2000, 2005). Continuing this idea, it can be stated that there would be differences not only in the scaling of truth values, but in the way of concluding as well. This means that in a way similar to the facts, positive and negative inferences can be separated. The idea of bipolar Datalog is based on the above observation: two kinds of ordinary fuzzy implications are used for positive and negative deduction, namely, a pair of consequence transformations is defined instead of a single one. Since in the original transformations lower bounds are used with degrees of uncertainty, therefore starting from IFS or IVS facts, the resulting degrees will be lower bounds of membership and non-membership respectively, instead of the upper bound for non-membership. However, if each non-membership value μ is transformed into membership value μ′ = 1 – μ, then both members of head-level can be deduced similarly. So the appropriate concepts are as follows.

Definition 8. The bipolar Datalog program (bDATALOG) defined on L_F or L_V is a finite set of safe bDATALOG rules $r; (\beta_1, \beta_2); (I_1, I_2)$.

The elements of the bipolar nondeterministic consequence transformation $bNT_P = (NT_{P1}, NT_{P2})$ are similar to NT_P in (1) except in NT_{P2} the level of rule's head is:

$$\alpha'_2 = max_{FV2}\ (0,\ min_{FV2}\ \{\gamma'_2 \mid I_2(\alpha'_{body2},\ \gamma'_2) \ge \beta'_2\});$$

where $\alpha'_{body2} = \min_{FV2}(\alpha'_{A_1 2}, \ldots, \alpha'_{A_n 2})$.

The uncertainty-level function is: $\underline{f} = (f_1, f_2)$ where

$$f_1 = min_{FV2}\ (\{\gamma_1 \mid I_1(\alpha_1,\ \gamma_1) \ge \beta_1\});$$

$$f_2 = 1 - min_{FV2}(\{1 - \gamma_2 \mid I_2(1 - \alpha_2, 1 - \gamma_2) \geq 1 - \beta_2\}).$$

It is evident that applying the transformation $\mu'_1 = \mu_1$, $\mu'_2 = 1 - \mu_2$, for all IFS levels of the program, the above definition can be applied to IVS degrees as well. As a simple computation can show, contrary to the results of iDATALOG, the resulting degrees for most variants of bipolar Datalog satisfy the conditions referring to IFS:

Proposition 5. For $\underline{\alpha} = (\alpha_1, \alpha_2)$, $\underline{\beta} = (\beta_1, \beta_2)$ and for implication-pairs $\underline{I} = (I_G, I_G)$; $\underline{I} = (I_L, I_L)$; $\underline{I} = (I_L, I_G)$; $\underline{I} = (I_K, I_K)$; $\underline{I} = (I_L, I_K)$;

$$\text{if } \alpha_1 + \alpha_2 \leq 1, \beta_1 + \beta_2 \leq 1 \text{ then } f_1(I_1, \underline{\alpha}, \underline{\beta}) + f_2(I_2, \underline{\alpha}, \underline{\beta}) \leq 1$$

From the construction of bipolar consequence transformations follows:

Proposition 6. The nondeterministic bipolar consequence transformation has a least fixed point, which is a model of program P in the following sense: for each $A \leftarrow A_1,\ldots,A_n; \underline{\beta}; \underline{I} \in ground(P)$

$$I(\alpha_{body1}, \alpha_1) \geq \beta_1; I(\alpha'_{body2}, \alpha'_2) \geq \beta'_2$$

As the termination of the consequence transformations based on these three implication operators was proven in the case of fDATALOG (Achs 2006) and since this property does not change in bipolar case, the bipolar consequence transformations terminate as well.

The bipolar extension of Datalog has no influence on the stratification, so the propositions detailed in the case of stratified fDATALOG programs are true in the case of bipolar fuzzy Datalog programs as well, that is, for a stratified bDATALOG program P, there is an evaluation sequence, in which $lfp(bNT_P)$ is a unique minimal model of P.

Example 6. Consider the next IFS program:

$$(p(a), (0.7, 0.2)).$$
$$(q(b), (0.65, 0.3)).$$
$$(r(a, b), (0.7, 0.2)).$$
$$r(x, y) \leftarrow p(x), q(y); (0.75, 0.2); \underline{I}.$$

Let $\underline{I} = \underline{I}_{FG_2}$, then according to the rule r(a,b) is inferred and uncertainty can be computed as follows: $\underline{\alpha}_{body} = \min_F((0.7, 0.2), (0.65, 0.3)) = (0.65, 0.3)$, $\underline{f}_1(\underline{I}_{FG_2}, \underline{\alpha}, \underline{\beta}) = \min(\alpha_1, \beta_1) = \min(0.65, 0.75) = 0.65$, $f_2(\underline{I}_{FG_2}, \underline{\alpha}, \underline{\beta}) = \max(\alpha_2, \beta_2) = 0.3$, that is, the level of the rule's head is (0.65, 0.3). There is a fact for r(a,b) as well, so the resulting level is the union of the level of the rule's head and the level of the fact: $\max_F((0.65, 0.3), (0.7, 0.2)) = (0.7, 0.2)$. So the fixed point of the program is:

$$\{(p(a), (0.7, 0.2)), (q(b), (0.65, 0.3)), (r(a, b), (0.7, 0.2))\}.$$

Let us see the bipolar evaluation of the program. Let $\underline{I} = (I_L, I_G)$. That is let the first element of the uncertainty level be computed according to the Lukasiewicz operator and the second one according to the Gödel operator. The Lukasiewicz operator defines the uncertainty level function $f(I_L, \alpha, \beta) = max(0, \alpha + \beta - 1)$. Then $\alpha_{body1} = min(0.7, 0.65) = 0.65$, $\alpha'_{body2} = min(1-0.3, 1-0.2) = 0.7$; $f_1(I_L, \alpha_1, \beta_1) = max(0, \alpha_1+\beta_1-1) = 0.65+0.75-1 = 0.4$, $f_2(I_G, \alpha'_2, \beta'_2) = 1 -$

$min(\alpha'_2, 1-\beta_2) = 1 - min(0.7, 0.8) = 0.3$. So the uncertainty level of rule's head is (0.4, 0.3). Considering the other level of r*(a,b)*, its resulting level is $(max(0.4, 0.7), 1-max(1-0.3, 1-0.2)) = (0.7, 0.2)$, so the fixed point is:

$$\{(p(a), (0.7, 0.2)), (q(b), (0.65, 0.3)), (r(a,b), (0.7, 0.2))\}.$$

Example 7. Consider the next (recursive) IVS program:

$(p(a, b), (0.7, 0.8)).$
$(p(a, c), (0.8, 0.9)).$
$(p(b, d), (0.75, 0.8)).$
$(p(d, e), (0.9, 0.95)).$
$q(x, y) \leftarrow p(x, y); (0.85, 0.95); \underline{I}_1.$
$q(x, y) \leftarrow p(x, z), q(z, y); (0.8, 0.9); \underline{I}_2.$

Let $\underline{I}_1 = \underline{I}_{VG2}$, $\underline{I}_2 = \underline{I}_{VL}$, that is the appropriate uncertainty level functions are:

$$f_1(\underline{I}_{VG2}, \underline{\alpha}, \underline{\beta}) = min(\alpha_1, \beta_1) \qquad f_2(\underline{I}_{VG2}, \underline{\alpha}, \underline{\beta}) = min(\alpha_2, \beta_2)$$

$$f_1(\underline{I}_{VL}, \underline{\alpha}, \underline{\beta}) = max(0, \alpha_2 + \beta_1 - 1) \qquad f_2(\underline{I}_{VL}, \underline{\alpha}, \underline{\beta}) = max(0, \alpha_1 + \beta_2 - 1)$$

Before showing the fixed point algorithm, two computations are set out. According to the first rule for $q(x, y)$ the uncertainty level of $q(a, b)$ is: $\underline{f}(\underline{I}_{VG2}, \underline{\alpha}, \underline{\beta}) = (min(\alpha_1, \beta_1), min(\alpha_2, \beta_2)) = (min(0.7, 0.85), min(0.8, 0.95)) = (0.7, 0.8)$.

According to the second rule, the uncertainty of $q(a, d)$ can be computed in this way:

The body of the rule is: $p(a, b), q(b, d)$, so $\alpha_{body} = min((0.7, 0.8),(0.75, 0.8)) = (0.7, 0.8)$; $\underline{f}(\underline{I}_{VL}, \underline{\alpha}, \underline{\beta}) = (max(0, \alpha_2 + \beta_1 - 1), max(0, \alpha_1 + \beta_2 - 1) = (max(0, 0.8 + 0.8-1), max(0, 0.7 + 0.9 - 1)) = (0.6, 0.6)$.

The steps of fixed point algorithm are:

$X_0 = \{(p(a, b), (0.7, 0.8)), (p(a, c), (0.8, 0.9)), (p(b, d), (0.75, 0.8)), (p(d, e), (0.9, 0.95))\}$
$X_1 = X_0 \cup \{ (q(a, b), (0.7, 0.8)), (q(a, c), (0.8, 0.9)), (q(b, d), (0.75, 0.8)), (q(d, e), (0.85, 0.95)), (q(a, d), (0.6, 0.6)), (q(b, e), (0.6, 0.65)) \}$
$X_2 = X_1 \cup \{ (q(a, e), (0.45, 0.5)) \}$

X_2 is fixed point, so it is the result of the program.

As fuzzy Datalog is a special kind of its multivalued extensions, so further on both fDATALOG and any of above extensions will be called multivalued Datalog (mDATALOG).

3. Multivalued knowledge-base

The facts of an mDATALOG program can be regarded as any kind of lexical knowledge including uncertainty as well, and from this knowledge other facts can be deduced according to the rules. Therefore a multivalued Datalog program is suitable to be the deduction mechanism of a knowledge base. Sometimes, however, it is not enough for getting answer to a question. For example, if there are rules describing the options of loving a good composer, and there is a fact declaring that Vivaldi is a good composer, what is the

possible answer to the question inquiring about liking Bach? Getting an answer needs the use of synonyms and similarities. For handling this kind of information, our model includes a background knowledge module.

3.1 Background knowledge

Some "synonyms" and "similarities" will be defined between the potential predicates and between the potential constants of a given problem, so it can be examined in a larger context. More precisely, proximity relations will be defined on the sets of the program's predicates and terms. These relations will serve as the basis for the background knowledge.

Definition 9. A multivalued proximity on a domain D is an IFS or IVS valued relation $\underline{R}_{FV_D}$: $D \times D \rightarrow [\underline{0}_{FV}, \underline{1}_{FV}]$ which satisfies the following properties:

$$\underline{R}_{F_D}(x, y) = \underline{\lambda}_F(x, y) = (\lambda_1, \lambda_2), \qquad \lambda_1 + \lambda_2 \leq 1$$
$$\underline{R}_{V_D}(x, y) = \underline{\lambda}_V(x, y) = (\lambda_1, \lambda_2), \qquad 0 \leq \lambda_1 \leq \lambda_2 \leq 1$$
$$\underline{R}_{FV_D}(x, y) = \underline{1}_{FV} \qquad \forall x \in D \qquad \text{(reflexivity)}$$
$$\underline{R}_{FV_D}(x, y) = R_{FV_D}(y, x) \quad \forall x \in D \quad \text{(symmetry)}.$$

A proximity is similarity if it is transitive, that is

$$\underline{R}_{FV_D}(x, z) \geq_{FV} min_{FV}(\underline{R}_{FV_D}(x, y), \underline{R}_{FV_D}(y, x)) \quad \forall x, y, z \in D.$$

In the case of similarity, equivalence classifications can be defined over D. The effect of this classification is perhaps a simpler or more effective algorithm, but in many cases the requirement of similarity is a too strict constraint. Therefore this chapter deals only with the more general proximity.

Background knowledge consists of the "synonyms" of each terms and each predicates of the program. The "synonyms" of any element form the proximity set of the element, and all of the proximity sets compose the background knowledge. More precisely:

Definition 10. Let $d \in D$ any element of domain D. The proximity set of d is an IFS or IVS subset over D:

$$\mathcal{R}_{FV_d} = \{(d_1, \underline{\lambda}_{FV_1}), (d_2, \underline{\lambda}_{FV_2}), \ldots (d_n, \underline{\lambda}_{FV_n})\},$$

where $d_i \in D$ *and* $\underline{\lambda}_{FV_i} = \underline{R}_{FV_D}(d, d_i)$ for $i = 1, \ldots, n$.

Definition 11. Let G be any set of ground terms and S any set of predicate symbols. Let $\underline{RG}_{FV_G}$ and $\underline{RS}_{FV_S}$ be any proximity over G and S respectively. The background knowledge is the set of proximity sets:

$$Bk = \{\mathcal{RG}_{FV_g} \mid g \in G\} \cup \{\mathcal{RS}_{FV_s} \mid s \in S\}$$

3.2 Computing uncertainties

Up to now, the deduction mechanism and the background knowledge of a multivalued knowledge-base have been defined. Now, the question remains: how can the two parts be connected to each other? How can we find the "synonyms"? For example, if it is known that

Ann loves the music of Bach very much (*(love(Ann, Bach), (0.9, 0.95))*) and the concept of love is similar to the concept of like ($\underline{RS}_{VS}$ *("love", "like")* = *(0.8, 0.9)*) and the music of Bach is more or less similar to the music of Vivaldi ($\underline{RG}_{VG}$ *(Bach, Vivaldi)* = *(0.7, 0.75)*) then how strongly can be stated that Ann likes Vivaldi, that is, what is the uncertainty of the predicate *like(Ann, Vivaldi)*?

To solve this problem, the concept of proximity-based uncertainty function will be introduced. According to this function, the uncertainty levels of "synonyms" can be computed from the levels of original fact 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 should be equal or less than the original level or than the proximity values. Furthermore, this function is required to increase monotonically. This function will be ordered to each atom of a program.

Let p be a predicate symbol with n arguments, then p/n is called the functor of the atom, characterized by this predicate symbol.

Definition 12. A proximity-based uncertainty function of p/n is:

$$\underline{\varphi}_p(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \ldots, \underline{\lambda}_n) : (\underline{0}_{FV}, \underline{1}_{FV}]^{n+2} \rightarrow [\underline{0}_{FV}, \underline{1}_{FV}]$$

where

$$\underline{\varphi}_p(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \ldots, \lambda_n) \leq min_{FV}(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \ldots, \underline{\lambda}_n)$$

$$\underline{\varphi}_p(\underline{\alpha}, \underline{1}_{FV}, \underline{1}_{FV}, \ldots, \underline{1}_{FV}) = \underline{\alpha}$$

and $\underline{\varphi}_p(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \ldots, \underline{\lambda}_n)$ is monotonically increasing in each argument.

Any triangular norm obeys the above constraints so they are appropriate proximity-based uncertainty functions.

Example 8. Let *(p(a), (0.7, 0.2))* be an *IFS* fact and $\underline{RS}_{FS}(p, q)$ = *(0.8, 0.1)*, $\underline{RG}_{FG}(a, b)$ = *(0.7, 0.3)* and $\underline{\varphi}_p(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1) = \min_F(\underline{\alpha}, \underline{\lambda} \cdot \underline{\lambda}_1)$.

(In IFS the product is defined as: $\underline{\mu} \cdot \underline{\lambda} = (\mu_1 \cdot \lambda_1, 1-(1-\mu_2)\cdot(1-\lambda_2))$.)

The uncertainty levels of *p(b)*, *q(a)* and *q(b)* are:

(p(b), (min((0.7, 0.2), ((1, 0)·(0.7, 0.3))) = (min(0.7, 0.7), max(0.2, 0.3)) = (0.7, 0.3));

(q(a), (min((0.7, 0.2), ((0.8, 0.1)·(1, 0))) = (min(0.7, 0.8), max(0.2, 0.1)) = (0.7, 0.2));

(q(b), (min((0.7, 0.2), ((0.8, 0.1)·(0.7, 0.3))) = (min(0.7, 0.56), max(0.2, 0.37)) = (0.56, 0.37));

Example 9. Let *(love(Ann, Bach), (0.9, 0.95)* be an *IVS fact* and $\underline{RS}_{VS}$*("love", "like")* = *(0.8, 0.9)*, $\underline{RG}_{VG}$ *(Bach, Vivaldi)* = *(0.7, 0.75))* and $\underline{\varphi}_{ove}(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \underline{\lambda}_2) = min_V(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1 \cdot \underline{\lambda}_2)$. Then

(love(Ann, Vivaldi), (min(0.9, 1, 1·0.7), min(0.95, 1, 1·0.75) = (0.7, 0.75));

(like(Ann, Bach), (min(0.9, 0.8, 1·1), min(0.95, 0.9, 1·1) = (0.8, 0.9));

(like(Ann, Vivaldi), (min(0.9, 0.8, 1·0.7), min(0.95, 0.9, 1·0.75) = (0.7, 0.75));

As the above examples show, the levels of "synonyms" can be computed according to proximity-based uncertainty functions. To determine all direct or indirect conclusions of the facts and rules of a program, a proximity based uncertainty function has to be ordered to each predicate of the program. The set of these functions will be the decoding-set of the program.

Definition 13. Let P be a multivalued Datalog program, and F_P be the set of the program's functors. The decoding-set of P is: $\Phi_P = \{\underline{\varphi}_p(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \ldots, \underline{\lambda}_n) \mid \forall p/n \in F_P\}$.

3.3 Deduction with background knowledge

The original deducing mechanism makes conclusions according to the rules of the program, but from now on the background knowledge must be considered as well. So the original mechanism has to be modified. This modified deduction consists of two alternating parts: starting from the facts, their "synonyms" are determined, then applying the suitable rules, other facts are derived, followed by their "synonyms" determined, and again the rules are applied, etc. To define it in a precise manner the concept of modified consecution transformation will be introduced.

The consequence transformation of a mDATALOG P program is defined over the set of all multivalued sets of P's Herbrand base, that is, over $FV(B_P)$. To define the modified transformation's domain, let us extend P's Herbrand universe with all possible ground terms of the background knowledge, obtaining the so called modified Herbrand universe, $modH_P$. The modified Herbrand base, $modB_P$ is the set of all ground atoms, whose predicate symbols occur in $P \cup Bk$ and whose arguments are elements of $modH_P$.

However, it is possible, that there are some special predicates in P, which have no alternatives, even if their arguments have "synonyms". These predicates are named as bound predicates. For such predicates, the modified Herbrand base only includes atoms that are present in the original Herbrand base.

Definition 14. The modified consequence transformation $modNT_P : FV(modB_P) \rightarrow FV(modB_P)$ is defined as

$$modNT_P(X) = \{(q(s_1,\ldots,s_n),\ \underline{\varphi}_p(\underline{\alpha}_p, \underline{\lambda}_q, \underline{\lambda}_{s_1}, \ldots, \underline{\lambda}_{s_n}))\ \mid$$

$$(q, \underline{\lambda}_q) \in \underline{RS}_{FV_p};\ (s_i, \underline{\lambda}_{s_i}) \in \underline{RG}_{FVt_i},\ 1 \le i \le n\} \cup X,$$

where

$$(p(t_1,\ldots,t_n) \leftarrow A_1,\ldots,A_k;\ \underline{\beta};\ \underline{I}) \in ground(P),$$

$$(|A_i|, \underline{\alpha}_{A_i}) \in X,\ 1 \le i \le k \qquad (|A_i| \text{ is the kernel of } A_i)$$

and $\underline{\alpha}_p$ is computed according to the actual extension of (1).

This transformation is inflationary over $FV(modB_P)$ and it is monotone if P is positive. So, according to (Ceri et al 1990) it has a least fixed point. If P is positive, this is the least fixed point. This fixed point is a model of P, but because $lfp(NT_P) \subseteq lfp(mod\ NT_P)$, it is not a minimal one (Achs 2010).

The modifying algorithm is irrelevant to the evaluation sequence, so stratification can be applied with the same condition. That is, the modified consequence transformation has a least fixed point in the case of stratified programs as well. This transformation makes connections between an mDATALOG program, the background knowledge and the decoding-set of the program. So these four components can form a knowledge-base. However, there should be other transformations connecting the three other parts with each other, therefore the universal concept of a multivalued knowledge-base can be defined with an arbitrary deduction algorithm:

Definition 15. A multivalued knowledge-base (*mKB*) is a quadruple

$$mKB = (P, Bk, \Phi_P, dA(P, Bk, \Phi_P)),$$

where P is a multivalued Datalog program, Bk is a background knowledge, Φ_P is a decoding-set of P and dA is any deduction algorithm connecting the three other part with each other. The least fixed point of the deduction algorithm is called the consequence of the knowledge-base, denoted by

$$C(Bk, P, \Phi_P, dA) = lfp(dA(P, Bk, \Phi_P)).$$

Because the actual deduction algorithm is the modified consequence transformation, now the consequence is

$$C(Bk, P, \Phi_P, dA) = lfp(modNT_P).$$

Note: If it is important to underline that there are bound predicates in P, then *mKB* can be denoted by $mKB = (P, Bp, Bk, \Phi_P, dA(P, Bp, Bk, \Phi_P))$, where Bp is the set of bound predicates.

Example 10. Let us suppose that an internet agent's job is to send a message to its clients if the cinema (*C*) presents a film, which its clients like. The agent knows that people generally go (*go*) to the cinema if they can pay (*cp*) for the ticket and are interested in (*in*) the film presented (*pr*) in the cinema. It also knows that people usually want to see (*ws*) a film if they like (*li*) its actor (*ac*). Moreover it knows that Paul (P) has enough money (*hm*) and he enjoys (*en*) Chaplin (*Ch*) very much. In the cinema, a film of Stan and Pan (*SP*) is presented. Should the agent inform Paul about this film? How much will he want to go to the cinema?

This situation can be modelled, for example, by the following multivalued knowledge-base.

Let the IVS valued mDATALOG program and the background knowledge be as follows

$$go(x, C) \leftarrow pr(C, f), in(x, f), cp(x); (0.85, 0.95); I_{VL}. \quad (R1)$$

$$ws(f, x) \leftarrow ac(f, y), li(x, y); (0.8, 0.85); I_{VG_2}. \quad (R2)$$

$$(hm(P), (0.75, 0.8)).$$
$$(en(P, Ch), (0.9, 0.95)).$$

$$(pr(C, Film), (1,1)).$$
$$(ac(Film, SP), (1,1)).$$

According to their roll, *pr(C, Film)* and *ac(Film, SP)* have no alternatives.

Let the proximities be:

$$\underline{R}_V(in, ws) = (0.7, 0.8). \qquad \underline{R}_V(Ch, SP) = (0.8, 0.9).$$

$$\underline{R}_V(li, en) = (0.8, 0.9).$$

$$\underline{R}_V(cp, hm) = (0.9, 0.95).$$

According to the connecting algorithm, it is enough to consider only the proximity-based uncertainty functions of head-predicates. Let these functions be the minimum function:

$$\underline{\varphi}_{go}(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \underline{\lambda}_2) = \underline{\varphi}_{ws}(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \underline{\lambda}_2) = \underline{\varphi}_{en}(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \underline{\lambda}_2) = \underline{\varphi}_{pr}(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \underline{\lambda}_2) = \underline{\varphi}_{ac}(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \underline{\lambda}_2) := \min_V(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \underline{\lambda}_2),$$

$$\underline{\varphi}_{im}(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1) := \min_V(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1),$$

The modified consequence transformation has the next steps:

$X_0 = \{(hm(P), (0.75, 0.8)), (en(P, Ch), (0.9, 0.95)), (pr(C, Film), (1,1)), (ac(Film, SP), (1,1))\}$

$\Downarrow$ (according to the proximity)

$X_1 = modNT_P(X_0) = X_0 \cup$

$\{(cp(P), \underline{\varphi}_{im}((0.75, 0.8), (0.9, 0.95), (1,1)) = (\min(0.75, 0.9, 1), \min(0.8, 0.95, 1)) = (0.75, 0.8)),$
$(en(P, SP), \underline{\varphi}_{en}((0.9, 0.95), (1,1), (1,1), (0.85, 0.9)) = (0.85, 0.9)),$
$(li(P, Ch), \underline{\varphi}_{en}((0.9, 0.95), (0.8, 0.9), (1,1), (1,1)) = (0.8, 0.9)),$
$(li(P, SP), \underline{\varphi}_{en}((0.9, 0.95), (0.8, 0.9), (1,1), (0.85, 0.9)) = (0.8, 0.9))\}$

$\Downarrow$ (applying the rules – only (R2) can be applied)

$X_2 = modNT_P(X_1) = X_1 \cup$

$\{(ws(Film, P), \underline{f}(\underline{I}_{VG2}, \underline{\alpha}, \underline{\beta}) = \underline{f}(\underline{I}_{VG2}, \min_V((1,1), (0.8, 0.9)), (0.8, 0.85)) =$
$\min_V((0.8, 0.9),(0.8, 0.85)) = (0.8, 0.85))\}$

$\Downarrow$ (according to the proximity)

$X_3 = modNT_P(X_2) = X_2 \cup$

$\{(in(Film, P), \underline{\varphi}_{ws}((0.8, 0.85), (0.7, 0.8), (1,1), (1,1)) = (0.7, 0.8))\}$

$\Downarrow$ (applying the rules – (R1) can be applied)

$X_4 = modNT_P(X_3) = X_3 \cup$

$\{(go(P, C), \underline{f}(\underline{I}_{VL}, \underline{\alpha}, \underline{\beta}) = \underline{f}(\underline{I}_{VL}, \min_V((1,1), (0.7, 0.8), (0.75, 0.8)), (0.85, 0.95)) =$
$(\max(0, 0.8 + 0.85 - 1), \max(0, 0.7 + 0.95 - 1)) = (0.65, 0.65))\}$

X_4 is a fixed point, so it is the consequence of the knowledgebase.

According to this result, the agent will know that the message can be sent, because Paul will probably enjoy Stan and Pan at a likelihood level of 85-90% (level (0.85, 0.9)), and there is a good chance (65%) that Paul will go to the cinema.

4. Evaluating algorithms

The fixed point-query is a bottom-up evaluation algorithm, which may involve many superfluous calculations. However, very often, only a particular question is of interest and the answer to this question needs to be searched. If a goal (query) is specified together with the multivalued knowledge-base, it is not necessary to evaluate all facts and rules, and it is

enough to consider only a part of them. A goal is a pair $(q(t_1, t_2, \ldots, t_n); \underline{\alpha})$, where $q(t_1, t_2, \ldots, t_n)$ is an atom, $\underline{\alpha}$ is the fuzzy, the intuitionistic, the interval-valued or the bipolar level of the atom. q may contain variables, and its levels may be known or unknown values. The evaluation algorithm gives answer to this query.

In standard Datalog, the most common approach in top-down direction is called query – sub-query framework. A goal, together with a program, determines a query. Literals in the body of any one of the rules defining the goal predicate are sub-goals of the given goal. Thus, a sub-goal together with the program yields a sub-query. In order to answer the query, each goal is expanded in a list of sub-goals, which are recursively expanded in their turn. That is, considering a goal, all rule-heads unifying with the goal are selected and the literals of the rule-body are the new sub-goals of given goal, which are evaluated one by one in an arbitrary order. This procedure continues until the facts have been obtained.

The situation is the same with a multivalued knowledge-base as well, but in this case the algorithm is completed with the computation of the unification levels. However, it is possible that such rules do not exist, but perhaps they do exist for the synonyms. For example, the goal is to know if Ann likes Bach, but there are rules only for describing the options of loving somebody and there are facts only about Vivaldi. In such cases, the synonyms are used. Therefore, the algorithm has to consider the proximities and has to compute the uncertainty levels. It is a bidirectional evaluation: firstly, the uncertainty-free rules and the proximities are evaluated in a top-down manner, obtaining the required starting facts, then the computing of uncertainties is executed in the opposite direction, that is, according to the fixed-point algorithm.

4.1 Evaluation of a general knowledge-base

The uncertainty levels are not required in the top-down part of the evaluation, so, this part of the algorithm can be based on the concept of classical substitution and unification (Ceri et al 1990, Ullman 1988, etc.) However other kinds of substitutions may be necessary as well: to substitute some predicate p or term t with their proximity sets RS_{FV_p} and RG_{FV_t}, and to substitute some proximity sets with their members.

From now on, for the sake of a simpler terminology, the terms "goal", "rule" and "fact" will refer to 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 facts, and uncertainty can be computed moving towards the direction of the root. This tree is built up by a periodic change of three kinds of steps: a proximity-based unification, a rule-based unification and a splitting step.

Proximity-based unification unifies the predicate symbols of sub-goals and the members of their proximity sets. Rule-based unification unifies the sub-goals with the head of suitable rules, and continues the evaluating by the bodies of these rules. The splitting step splits the rule-body into sub-goals if the body contains more literals or splits a literal of proximity sets into literals of the suitable ground atoms.

During the construction process, the edges are labelled by necessary information for computing the uncertainties. 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, …)* is in AND connection, and the others are in OR connection. The step after depth *3k (k = 0, 1, …)* is a proximity-based unification, after depth *3k+1 (k=0, 1, …)* is a rule-based unification and after depth *3k+2 (k=0,1,…)* is a splitting step. In detail:

If the atom $p(t_1, t_2, \ldots, t_n)$ is in depth *3k (k = 0, 1, …)*, then the successor nodes let be all possible $p'(t_1, t_2, \ldots, t_n)$, where $p' \in RS_{FV_p}$. The edges starting from these nodes are labelled by the proximity-based uncertainty functions $\varphi_{p'}$.

If the atom *L* is in depth *3k+1 (k=0, 1, …)*, 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 $M \leftarrow M_1,\ldots,M_n$, *(n>0)* is unifiable with *L*, then the successor of *L* will be $M_1\theta,\ldots,M_n\theta$, where θ is the most general unification of *L* and *M*. The edges starting from these nodes are labelled by the uncertainty functions belonging to the implication operator of the applied rules and by the uncertainty level of the rule.

If *n=0*, that is, in the program there is any fact with the predicate symbol of *L*, then the successors will be the unified facts. If $L = p(t_1, t_2, \ldots, t_n)$ and in the program there is any fact with predicate symbol *p*, then let the successor nodes be all possible $p(t'_1, t'_2,\ldots, t'_n)$, where $t'_i = t_i$ if t_i is a ground term or $t'_i = RG_{FV_{ti}\theta}$ if t_i is a variable and θ is a suitable unification. The edges starting from these nodes are not labelled.

According to the previous paragraph, there are three kinds of nodes in depth *3k+2 (k=0,1,…)*: a unified body of a rule; a unified fact the arguments of with are ordinary ground terms or proximity sets; or the symbol NO.

In the first case, the successors are the members of the body. They are in AND connection. The connected edges will not be labelled, but because of the AND connection, during the computation, the minimum value of the successor's levels will be regarded.

In the second case, the successors are the so called facts-sets. This means, that if the node is $p(t_1, t_2,\ldots, t_n)$, where t_i is a ground term or a proximity set, then the facts-set is the set of all possible $p(t'_1, t'_2,\ldots, t'_n)$, where $t'_i \in RG_{FV_{ti}}$. The edges starting from these nodes are labelled by the proximity-based uncertainty functions φ_p.

The facts-set has a further successor, the symbol YES.

The NO-node has no successor.

A solution can be achieved in the graph along the path ending in the symbol YES. According to the unification algorithm, one of the literals that are located at the parent node of YES can also be found among the original facts of the program. Knowing its uncertainty and using the proximity-based uncertainty function of the label leading to this facts-set, the uncertainty of other members of the facts-set can be computed. However, it is not necessary to compute all of them, only those ones, which are appropriate for the pattern of the literal being in the parent node of facts-set. This means, that if the literal is $p(t_1, t_2,\ldots, t_n)$, and t_i is a ground term, then it is enough to consider only $p(t'_1, t'_2,\ldots, t_i,\ldots, t'_n)$ from the facts-set, but if t_i is a proximity set, then it is necessary to deal with all $p(t'_1, t'_2,\ldots, t'_i,\ldots, t'_n)$, where $t'_i \in RG_{FV_{ti}}$.

In this way each starting fact can be appointed. Then a solution can be determined by connecting the suitable unifications and computing in succession the uncertainties according to the labels of edges in the path from the symbol YES to the root of the graph. The union of these solutions is the answer to the given query.

From the construction of searching graph follows:

Proposition 7. For a given goal and in the case of finite evaluation graph, the top-down evaluation terminates and gives the same answer as the fixed point query.

Example 11. Consider the IFS program of Example 6, and let it be completed by proximities and proximity-based uncertainty functions

$(p(a), (0.7, 0.2))$.

$(q(b), (0.65, 0.3))$.

$(r(a, b), (0.7, 0.2))$.

$r(x, y) \leftarrow p(x), q(y); (0.75, 0.2); \underline{I}_{FG2}$.

$\underline{R}_F (p, p_1) = (0.7, 0.1)$. $\quad \varphi_p(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1) := \underline{\alpha} \cdot \underline{\lambda} \cdot \underline{\lambda}_1$.

$\underline{R}_F (q, q_1) = (0.8, 0.1)$. $\quad \varphi_q(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1) := min_F (\underline{\alpha}, \underline{\lambda} \cdot \underline{\lambda}_1)$.

$\underline{R}_F (r, r_1) = (0.75, 0.2)$. $\quad \varphi_r(\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \underline{\lambda}_2) := min_F (\underline{\alpha}, \underline{\lambda}, \underline{\lambda}_1, \underline{\lambda}_2)$.

$\underline{R}_F (a, a_1) = (0.8, 0.1)$.

$\underline{R}_F (b, b_1) = (0.6, 0.3)$.

Let the goal be: $(r_1(a_1,x); \underline{\alpha})$, where x is a variable. Then the evaluation graph is on the next page.

The three facts-sets can easily be seen in the graph. From the first one, the uncertainty of $r(a_1, b)$ and $r(a_1, b_1)$ can be computed.

As $(r(a, b), (0.7, 0.2))$ is a known member of the set, so knowing this uncertainty, the proximity-based function and the proximities of knowledge base, the uncertainties can be determined:

$$r (a_1, b), (min_F ((0.7, 0.2), (1, 0), (0.8, 0.1), (1,0))= (0.7, 0.2)).$$

$$r (a_1, b_1), (min_F ((0.7, 0.2), (1, 0), (0.8, 0.1), (0.6,0.3))= (0.6, 0.3)).$$

Applying the next label of the path the uncertainty of $r_1(a_1, b)$ and $r_1(a_1, b_1)$ can be found. They are as follows:

$$(r_1(a_1, b), (0.7, 0.2))$$

$$(r_1(a_1, b_1), (0.6, 0.3))$$

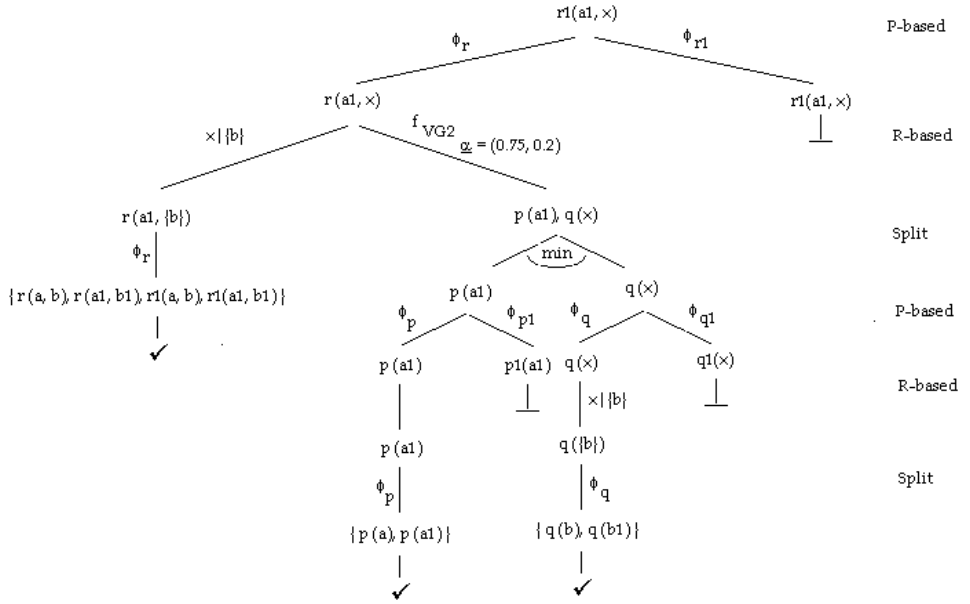


Fig. 1. The evaluation graph of Example 11.

In the second facts-set *(p(a), (0.7, 0.2))* is the known fact, from this *(p(a1), (0.56, 0.28))*.

Similarly *(q(b), (0.65, 0.3))*, *(q(b1), (0.6, 0.3))*. Applying the min, f_{FG2} and $\underline{\varphi}_r$ functions:

$$(r_1(a_1, b), (0.56, 0.3))$$

$$(r_1(a_1, b_1), (0.56, 0.3))$$

As the answer is the union of the different solutions, the final answer is:

$$(r_1(a_1, b), (0.7, 0.2))$$

$$(r_1(a_1, b_1), (0.6, 0.3))$$

4.2 Special evaluation based on multivalued unification

The necessity of bidirectional evaluation is derived from the generality of implications and proximity functions, because their values can be computed only from known arguments, namely in bottom-up manner. However, in special cases, computation can be realized parallel with the evaluation of rules and proximities, so the algorithm could be a more efficient pure top-down evaluation. This is the situation if all of the functions are the minimum function. That is, all implications that are used are the second Gödel implication (G2) and all proximity-based uncertainty functions are the minimum function.

As mentioned earlier, in the case of standard Datalog, the heart of the evaluation algorithm is the unification process. Our special evaluation of multivalued knowledge is based on unification as well, but on multivalued unification. The multivalued unification consists of two parts, one is the alternation of a rule-based-unification and the other is a proximity-based one. Both of them are the extensions of the classical unification algorithm. Now, the splitting step is inside these unifications: evaluating a fact, the last proximity-based unification unifies the fact with its facts-set, and a rule-based unification splits these sets into their members.

4.2.1 Rule-based unification

This unification algorithm is similar to the classical one, that is, the goal can be unified with the body of any one of the rules defining the goal predicate – if the body is not empty. The level of the unification is the level of the rule defining the goal predicate. In that case, a variable can be substituted with other variable or with a constant; a constant can be substituted with itself only. The next sub-goal of the evaluation process will be the first member of the body. It is possible that during the evaluation a variable of a later member is substituted by a proximity set. In such a case, in the course of later evaluation, this proximity set will be substituted with itself.

If the predicate symbol of the goal is the predicate symbol of a fact, its arguments can be substituted as follows:

- The variables of the goal can be substituted with the proximity set of the constants being the corresponding arguments of the fact. (E.g.: if *q(a,b)* is a fact predicate and the goal is *q(x, y)*, then x can be substituted with R_{FV_a} and y with R_{FV_b}.)
- The constants of the goal can be substituted
  - with themselves, if the goal contains any variable or
  - with their proximity set if the goal does not contain any variable.
- The proximity set argument of a goal can be substituted with itself.

The level of the unification is $\underline{1}_{FV}$.

If there is no fact with the same predicate symbol, the unification process fails.

There is a special case of unification: the facts-set of a predicate is unified with its members in the following way:

According to the previous unifications, between the literals of these facts-set there is one from the facts of the program. Knowing its uncertainty and the level of proximities, the uncertainty of other members can be computed. Then these members can be unified respectively:

- with the empty clause, if there is no other literal to evaluate in the parent-node;
- with the remaining part of the body to be evaluated.

The level of this unification is the level of the fitting member, and the former proximity set-substitution of a variable is replaced by the suitable member of this set.

(E.g. : if there is a former $x \mid R_{FV_a}$ substitution for literal *p(x)*, and $R_{FV_a} = \{a, b\}$, then the facts-set of unification is {*p(a), p(b)*}, which is unified with empty clauses . The substitutions for x are $x \mid a$ and $x \mid b$, and the levels are the computed levels of *p(a)* and *p(b)* respectively.)

4.2.2 Proximity-based unification

This unification serves for handling proximities. When these steps are implemented, the following substitutions can be realized:

- A constant or a variable can be substituted with itself only.
- A predicate symbol can be substituted with the elements of its proximity set. The level of the unification is the current proximity value.
- A proximity set can be substituted with itself except in the last step of the evaluation of a literal. In this case, that is if each argument of the literal is a proximity set, the literal can be unified with its facts-set. The level of the unification is $\underline{1}_{FV}$.

4.2.3 The unification algorithm

The evaluation algorithm combines the two kinds of unification. It starts with the proximity based unification and after it is finished, they alternate. The query is successful if the unification algorithm ends with an empty clause or a failure. In the first case the variables get the values defined during the substitutions. If all uncertainties are regarded as a minimum value, the actual level of unification can be computed as the minimum of former levels and when the algorithm reaches the empty clause, its level will be the level of the goal as well.

If the unification algorithm ends with a failure, there is no answer on this path.

If more answers arise during the evaluation, their union will be the resolution of the query.

According to the construction of unifications, the following proposition is true.

Proposition 8. For a given goal, and in the case of a finite evaluation graph, the above top-down evaluation gives the same answer as the fixed point query.

Notes:

- Although this algorithm was described for a knowledge-base based on a negation free program, it is similar in the case of stratified programs; the only difference is the calculation of the uncertainty of the negated sub-goal, but the computing of minimum remains the same.
- With a good depth limit this algorithm is suitable for evaluating recursive programs or infinite graphs as well.

Example 12. Let us consider a part of Example 10, and let it be completed with a new rule and new facts. That is now the internet agent knows that people usually want to see (*ws*) a film if they like (*li*) its actor (*ac*), or they like more or less the subject (*su*) of the film. Moreover, it knows that Paul (P) enjoys (*en*) Chaplin (*Ch*) very much and mostly enjoys the historical films (*H*). In the cinema, a film (*F1*) of Stan and Pan (*SP*) is presented. There are two other films (*F2, F3*). Both films' topics (*to*) are the war (*W*), but in different manner. The first one's central message is the war, the second one play in wartime, but it is only a background. From the former example it is known, that the agent wants to know the interest of Paul. Therefore let our goal be $(in(P,x); \underline{\alpha})$.

Let the IVS valued mDATALOG program and the background knowledge be as follows

$ws(f, x) \leftarrow ac(f, y), li(x, y); (0.8, 0.85); I_{VG_2}.$ *(R1)*

$ws(f, x) \leftarrow su(f, y), li(x, y); (0.75, 0.85); I_{VG_2}.$ *(R2)*

(en(P, Ch), (0.9, 0.95)).
(en(P, H), (0.6, 0.7)).

(ac(F1, SP), (1, 1)).
(to(F2, W), (0.9, 0.95)).
(to(F3, W), (0.55, 0.6)).

According to its roll, *ac(F1, SP)* has no alternatives. Let the other proximities be:

$\underline{R}_V$ (in, ws) = (0.7, 0.8). $\underline{R}_V$ (Ch, SP) = (0.8, 0.9).

$\underline{R}_V$ (li, en) = (0.8, 0.9). $\underline{R}_V$ (H, W) = (0.6, 0.8).

$\underline{R}_V$ (to, su) = (0.9, 1).

Then the evaluation graph is Fig.2.

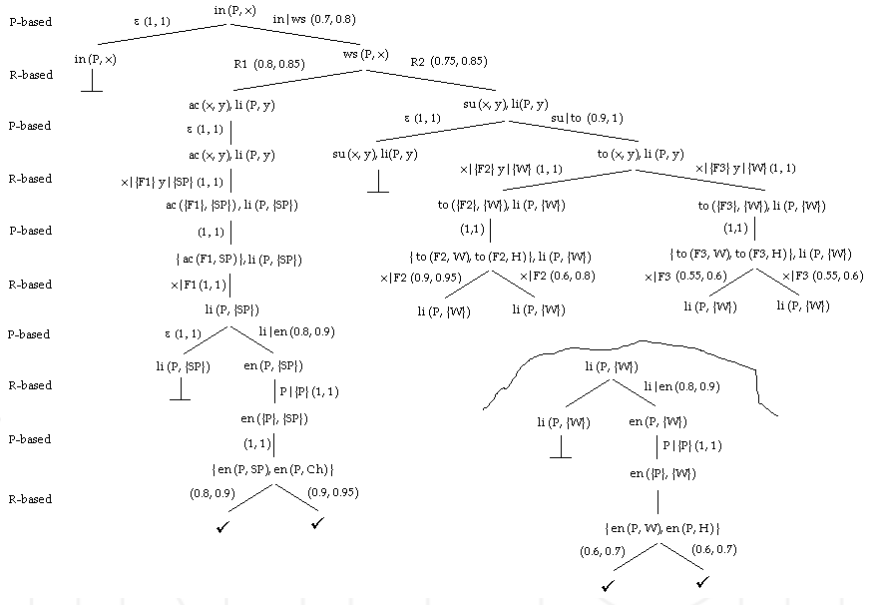


Fig. 2. The evaluation graph of Example 12.

Let each proximity-based uncertainty function be the minimum function.

According to the left path of the graph one can see, that *(in (P, F1), (0.7, 0.8))* because the applied substitution is $x \mid F1$ and the minimum of levels is *(0.7, 0.8)*.

The other paths are only half-drawn, and they are continued in a partial graph, because this part of evaluation is similar in all cases. The only differences are in uncertainty levels.

So, according to these paths the other answers for the query are:

(in (P, F2), (0.6, 0.7)), (in (P, F3), (0.55, 0.6))

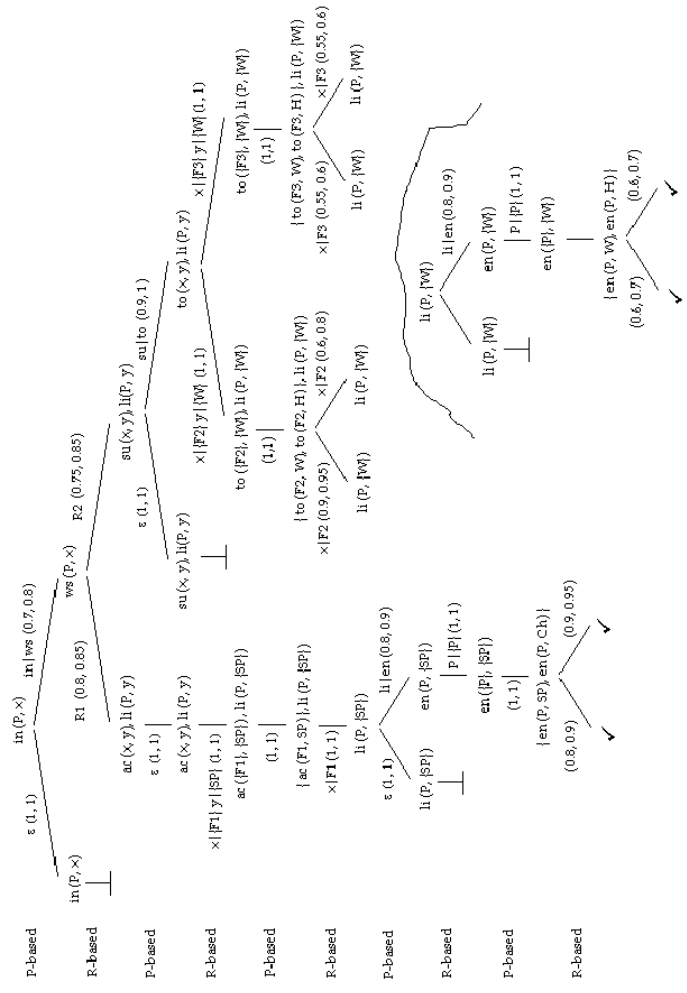


Fig. 3. The enlarged evaluation graph of Example 12.

5. Conclusion

In this chapter, a possible multivalued knowledge-base was presented as a quadruple of background knowledge, a deduction mechanism, a decoding set and an algorithm connecting the background knowledge to the deduction mechanism.

The background knowledge is based on proximity relations between terms and between predicates and it serves as a mechanism handling "synonyms".

The deduction mechanism can be any of the extensions of Datalog. These extensions are the fuzzy Datalog, based on fuzzy logic, the intuitionistic- or interval-value Datalog, based on the suitable logics and bipolar Datalog, which is some kind of coexistence of the former ones.

The semantics of Datalog is a fixed-point semantics, so the algorithm, which connects the two main pillars of the knowledge-base is the generalization of the consequence transformation determining this fixed-point. This transformation is defined on the extended Herbrand base of the knowledge-base, which is generated from the ground terms of knowledge base and its background knowledge.

Applying this transformation it is necessary to compute the uncertainty levels of "synonyms". The proximity-based uncertainty functions can do it, giving uncertainty values from the levels of the original fact and from the proximity values. The set of this kind of functions is the decoding set.

Two possible evaluation strategies were presented as well. One of them evaluates a general knowledgebase with arbitrary proximity-based uncertainty functions and arbitrary implication operators. The other one allows minimum functions only as proximity based uncertainty functions, and the special extension of Gödel operator, but in this case a multivalued unification algorithm can be determined. This strategy is based on the alternating rule-based- and proximity-based unification.

The improvement of this strategy and/or the deduction algorithm and/or the structure of background knowledge is a subject of further investigations.

A well structured multivalued knowledge-base and an efficient evaluating algorithm determining its consequence could be the basis of making decisions based on uncertain information, or it would be useful for handling argumentation or negotiation of agents. An implementation of this model would be an interesting future development as well.

6. References

Abiteboul, S.; Hull, R. and Vianu, V. (1995). *Foundations of Databases*. Addison-Wesley Publishing Company, Reading, Massachusetts.

Achs, A. & Kiss, A. (1995). Fuzzy extension of datalog, *Acta Cybernetica Szeged*, Vol.12, pp. 153-166.

Achs, A. (2006). Models for handling uncertainty, *PhD thesis*, University of Debrecen, 2006.

Achs, A. (2007). From Fuzzy- to Bipolar- Datalog, In: *Proceedings of 5th EUSFLAT Conference*, Ostrava, Czech Republic, pp. 221-227.

Achs, A. (2010). A multivalued knowledge-base model, *Acta Universitatis Sapientiae, Informatica*, Vol.2, No.1, pp. 51-79.

Alsinet, T. & Godo, L. (1998). Fuzzy Unification Degree, In: *Proceedings 2nd Int Workshop on Logic Programming and Soft Computing '98, in conjunction with JICSLP'98,* Manchester, UK, pp. 23-43.

Atanassov, K. (1983). Intuitionistic fuzzy sets, *VII ITKR's Session, Sofia* (deposed in Central Science-Technical Library of Bulgarian Academy of Science, 1697/84).

Atanassov, K. & Gargov, G. (1989). Interval-valued intuitionistic fuzzy sets, *Fuzzy Sets and Systems*, Vol.31, No.3, pp. 343-349.

Atanassov, K. (1994). Remark on intuitionistic fuzzy expert systems. *BUSEFAL*, Vol.59, pp. 71-76.

Atanassov, K. (2005). Intuitionistic fuzzy implications and modus ponens. *Notes on Intuitionistic Fuzzy Sets,* Vol. 11, pp. 1-5.

Atanassov, K. (2006). On some intuitionistic fuzzy implications. *Comptes Rendus de l'Academie Bulgare des Sciences, Tome*, Vol. 59, pp. 19-24.

Atanassov, K. (1999). *Intuitionistic Fuzzy Sets*, Springer-Verlag, Heidelberg

Ceri, S.; Gottlob, G. & Tanca, L. (1990). *Logic Programming and Databases*, Springer-Verlag, Berlin

Cornelis, C.; Deschrijver, G. & Kerre, E.E. (2004). Implication in intuitionistic fuzzy and interval-valued fuzzy set theory: construction, classification, application, *International Journal of Approximate Reasoning*, Vol.35, pp. 55-95.

Dubois, D. & Prade, H. (1991). Fuzzy sets in approximate reasoning, Part 1: Inference with possibility distributions, *Fuzzy Sets and Systems,* Vol.40, pp. 143-202.

Dubois, D.; Hajek, P. & Prade, H. (2000). Knowledge-Driven versus Data-Driven Logics, *Journal of Logic, Language, and Information*, Vol.9, pp. 65-89.

Dubois, D.; Gottwald, S.; Hajek, P.; Kacprzyk, J. & Prade, H. (2005). Terminological difficulties in fuzzy set theory – The case of Intuitionistic Fuzzy Sets, *Fuzzy Sets and Systems*, Vol.15, pp. 485-491.

Formato, F.; Gerla, G. & Sessa, M.I. (2000). Similarity-based Unification, *Fundamenta Informaticae*, Vol.41, pp. 393-414.

Julian-Iranzo, P. & Rubio-Manzano, C. (2009). A declarative semantics for Bousi~Prolog, *PPDP '09: Proceedings of the 11th ACM SIGPLAN conference on Principles and practice of declarative programming*.

Julian-Iranzo, P. & Rubio-Manzano, C. (2010). An Efficient Fuzzy Unification Method and its Implementation into the BousiProlog System, *WCCI2010 IEEE World Congress On Computational Intelligence, Barcelona*.

Lloyd, J.W. (1990). *Foundations of Logic Programming*, Springer-Verlag, Berlin

Medina, J.; Ojeda-Aciego, M. & Vojtas, P. (2004). Similarity-based unification: a multi-adjoint approach, *Fuzzy Sets and Systems*, Vol.146, No.1, pp. 43-62.

Sessa, M. I. (2002). Approximate reasoning by similarity-based SLD resolution, *Theoretical Computer Science*, Vol.275, No.1-2, pp. 389-426.

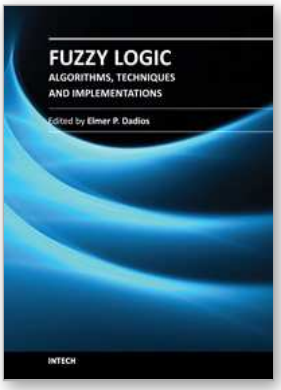
Straccia, U. (2008). Managing Uncertainty and Vagueness in Description Logics, Logic Programs and Description Logic Programs, In: Reasoning Web 2008, C. Baroglio et al. (Eds), pp. 54-103, Springer-Verlag, Berlin

Straccia, U.; Ojeda-Aciego, M. & Damasio, C.V. (2009). On Fixed-points of Multi-valued Functions on Complete Lattices and their Application to Generalized Logic Programs, *SIAM Journal on Computing*, Vol.8, pp. 1881-1911.

Ullman, J.D. (1988). *Principles of Database and Knowledge-base Systems*, Computer Science Press, Rockville

Virtanen, H.E. (1994) Fuzzy Unification, *Proc. of IPMU'94, Paris (France)*, pp. 1147–1152.

Zadeh, L. A. (1975) The concept of a linguistic variable and its application to approximate reasoning (I–II-III), *Information Sciences*, Vol.8, pp. 199-249; 301-357; Vol.9 pp. 43-80.



Fuzzy Logic - Algorithms, Techniques and Implementations
Edited by Prof. Elmer Dadios

ISBN 978-953-51-0393-6
Hard cover, 294 pages
Publisher InTech
Published online 28, March, 2012
Published in print edition March, 2012

Fuzzy Logic is becoming an essential method of solving problems in all domains. It gives tremendous impact on the design of autonomous intelligent systems. The purpose of this book is to introduce Hybrid Algorithms, Techniques, and Implementations of Fuzzy Logic. The book consists of thirteen chapters highlighting models and principles of fuzzy logic and issues on its techniques and implementations. The intended readers of this book are engineers, researchers, and graduate students interested in fuzzy logic systems.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821