KYBERNETIKA — VOLUME 36 (2000), NUMBER 6, PAGES 707 - 720

DECLARATIVE AND PROCEDURAL SEMANTICS OF FUZZY SIMILARITY BASED UNIFICATION

PETER VOJTÁŠ

In this paper we argue that for fuzzy unification we need a procedural and declarative semantics (as opposed to the two valued case, where declarative semantics is hidden in the requirement that unified terms are syntactically – letter by letter – identical). We present an extension of the syntactic model of unification to allow near matches, defined using a similarity relation. We work in Hájek's fuzzy logic in narrow sense. We base our semantics on a formal model of fuzzy logic programming extended by fuzzy similarities and axioms of predicate calculus with equality. Rules are many valued implications and not Horn clauses. We prove soundness and completeness of fuzzy similarity based unification.

1. INTRODUCTION

The aim of this paper is to build a formal model for fuzzy unification containing both declarative part and procedural part and to prove it's soundness and completeness. It is an extension of Hájek's fuzzy logic in narrow sense (cf. [7]) and the author's work on fuzzy logic programming (cf. [17]) with crisp unification to include partial rather than exact matches in unification, and an examination of theoretical consequences. We stress here that in our model rules of a fuzzy logic program are implications and not Horn clauses (which in many valued case makes a difference when calculating the truth valued of the whole rule along the complexity of formula using truth values functions of connectives – we work in an truth extensional logic of P. Hájek, cf. [7]).

It is quite usual in mathematical logic to have clearly defined syntactical(dealing with proofs) part of formal model and the semantical(dealing with truth and/or satisfaction) part. In application of logic to computer science (mainly in logic programming) a different terminology developed, we speak about declarative part of the formal model (corresponding to truth and satisfaction) and procedural part more focused on algorithmic aspect of finding proofs (automated deduction).

Unification is an important part of procedural semantics of many formal models, because it helps to unify (that is to make syntactically – letter by letter – equal) and unified (identical) statements can be handled identically. There was no need to specify a declarative and procedural part of a formal model of unification because on the declarative part there was the requirement of being syntactically equal (and hence equal also from the point of view of truth, satisfaction). Procedural part of unification was well developed – namely different unification algorithms (cf. [11, 14]).

This is no more case when considering fuzzy unification. Here we desperately need a declarative and procedural part of a formal model of fuzzy unification. Fuzzy unification typically deals with the requirement of unifying different objects.

For illustration imagine a hotel name "Salaš" in Slovak can in Polish read as "Szalas", in Hungarian "Szálás" and in English (omitting the "check sign" from Slovak) as "Salas". Does the query using "Salaš" unify with database fact about "Szalas", do subgoal about "Szálás" and "Salas" fit together?

A common sense intuition suggests, that without knowing any additional fact, knowledge, the system itself cannot glue together information about objects with different syntactical form.

	correctness = soundness ⟵	
classical logic	semantical consequence	modus ponens proofs
logic programming	$Q\theta$ semantical consequence of P	resolution
unification	syntactical equality	unification algorithm
fuzzy logic programming	truth degree of $Q\theta$ in models of P at least x	backward modus ponens
fuzzy unification	?? new declarative model of fuzzy unification	??new fuzzy unification algorithm
FLP integrated with fuzzy unification	??integrated declarative model	??integrated procedural model
	⟶ completeness	

So the main question is how to describe this additional information. In this paper we chose the way of having additional information about fuzzy similarities of different objects and using axioms of equality to transfer properties between these objects. There are many other different sources of similarities between objects, we mentioned here only one possible concerning some middle european language peculiarities but we can imagine other syntactical, linguistical and conceptual sources (see e. g. [9]). The main aim of this paper is not to give practical advice how to detect and handle similarities in practical applications but to give a formal model for both declarative and procedural part of similarity based fuzzy unification.

In analogy with some programming paradigm we depict the role of procedural and declarative semantics of fuzzy unification schematically in above table (first column

denotes the formal model, the second its declarative part and the last column the procedural (syntactical part), two arrows denote the direction of implications in soundness and completeness).

Moreover, we depicted that there is a natural requirement that between procedural and declarative part of a formal model there should be some connection. We would like to have our proofs (algorithms) are proving true statements (soundness). This is a minimal requirement for a deductive system having any sense. More difficult (and not always fulfilable) is the requirement of the completeness, namely whenever we have a valid statement, we would like to know our system will find it (in a recursive or recursively enumerable sense).

We base our approach on the theory of fuzzy logic programming with crisp unification. We recall definitions of declarative and procedural semantics of fuzzy logic programming and show how soundness(correctness) and completeness of it, especially the fixpoint theorem and the minimal model obtained by the iteration of the T_P operator can give a base for sound and complete model of fuzzy unification.

The fact that the theory of fuzzy unification is developed inside the realm of fuzzy logic programming is very important for later integration of fuzzy similarity based unification and fuzzy logic programming deduction.

Developing the declarative semantics of a programming language enables us to reason about programs and their meaning without having to execute the code on a computer. A human must be able to understand each piece of code (clause) in a mathematical and/or natural language sense. Logic is well suited to this, as symbols can be mapped to objects in the real world and the connectives in the two valued logic are fairly well understood. Here we have to stress that fuzzy logic offers a comparative notion of truth (see [7]). So having the truth value of formula to be e. g. 0.8 does not mean anything fixed in the real world. It just states that the truth value is not 1 nor 0 and that it is much closer to 1 than to 0. It becomes more expressive in comparison with truth values of other formulas. The role of many valued connectives is to preserve this comparativeness of truth in more complex statements.

2. FUZZY LOGIC PROGRAMMING WITH CRISP UNIFICATION

We will need the full theory of fuzzy logic programing with crisp unification,

1. first because of the realm inside which we define procedural and declarative semantics of our similarity based unification and
2. second because only after working inside this theory we can give a simpler model of fuzzy similarity based unification.
3. third, because even if axioms of equality have form

$$(p(x_1,\ldots,x_n)\ \&_P\ x_1 =_1 y_1\ \&_P\ \ldots\ \&_P\ x_n =_n y_n \longrightarrow p(y_1,\ldots,y_n),1)\,.$$

with truth value 1 the fuzzy degrees of similarities $=_1,\ldots,=_n$ introduce truth values of subgoal in general smaller than one and we need a model of deduction with fuzzy theories.

4. Note, we couple similarities together and with truth value of $p(x_1, \ldots, x_n)$ via product, because it is independent of validity of $p(x_1, \ldots, x_n)$ whether there is or not a need for the use of similarities in the answer caused e. g. by inexact translation, misprints, misspelling or some other reason a similarity has to be used.

A standard reference for fuzzy logic is the book of Petr Hájek [7], development of fuzzy logic programming with lot of applications and heuristics and almost no formal models is reviewed in [4], theory and a formal model of fuzzy logic programming based on rules as implications which we repeat here in a abbreviated form was developed in [17] and [18]. For similarity based approach we refer to [13] and [4]. Another approach to quantitative deduction was started by a work of M. van Emden [15] and culminated in works of M. Kifer and V. S. Subrahmanian [8] and A. Dekhtyar and V. S. Subrahmanian [5] within development of annotated and hybrid logic programming. The main difference is that we work with exptensional logic assigning truth values to implications (rules) and they calculate truth value of the head from truth value of the body, without speaking about truth value of their rules. Although there is much to be said about connections of our approach and annotated/hybrid logic programming, it is out of the scope of the paper.

2.1. Language

Our language is a language of possible domains of applications of fuzzy similarity based unification, namely many sorted predicate language with or without function symbols. The set of all attributes is denoted by $\mathcal{A}$. For each sort of variables (attribute) $A \in \mathcal{A}$ there is the set $\mathcal{C}^A$ of constant symbols of that sort (these are names of elements of domain of attribute A). These are domains where our similarities act.

A predicate $p(A_1, \ldots, A_n)$ is defined by its sorts of variables (attributes). For each attribute A (and in some cases also for each predicate) we have a similarity relation (fuzzy equivalence) $s_A(\mathcal{C}^A, \mathcal{C}^A)$.

Function symbols have arity defined by a list of attributes for input and an attribute for the result $f(\mathcal{A}_1, \ldots, \mathcal{A}_n) = \mathcal{A}$.

To capture different interdependences between predicates our language can have several many valued connectives:

conjunctions $\&_1, \&_2, \ldots, \&_k$,
disjunctions $\vee_1, \vee_2, \ldots, \vee_l$,
implications $\rightarrow_1, \rightarrow_2, \ldots, \rightarrow_m$ and
aggregations $@_1, @_2, \ldots, @_n$.

Specific examples of many valued connectives are Łukasiewicz connectives, Goedel connectives, connectives of product logic and other t-norm based connectives (see e. g. [7]). As already mentioned in the introduction, fuzzy logic offers a comparative notion of truth without assigning any special meaning to the fact that the truth value is e. g. 0.8. On the other side we need a rich variety of connectives to preserve this comparative notion of truth among more complex statements. Although it is not probability it resembles some features of it (maybe due to a probabilistic nature of how humans assign meaning to linguistical object?). To illustrate this let us recall

from [19]: "... $\max(0, x+y-1)$ is the minimal measure of a intersection of events with probability x and y, independent sets meet in a set with measure xy and $\min(x, y)$ represents the intersection of overlapping events. In this sense (loosely speaking) we can understand the scale of all possible many-valued connectives representing the scale of notions, for example:

Łukasiewicz – pessimist, minimal lost, competition, contradictory, expert vs. laymen, enemies

Product – realist, moderate outcome, tolerance, independent, two laymen, both selfish

Goedel – optimist, maximal profit, cooperation, affirmative, two experts, friends

V. S. Subrahmanian and A. Dekhtyar [5] use another linguistic description: Łukasiewicz – ignorance, Goedel – positive (negative) correlation, product – independence.

Nevertheless, in front of this big variety of connectives, we will argue that similarities are coupled via the product conjunction.

The syntactical level is not touched by many valuedness of truth functions and fuzzy structures. Nevertheless, because we have several different connectives one cannot expect associativity and commutativity between them and hence parentheses should be used more consequently and traditional abbreviations in notation do not apply here. The only we will need is that connectives in equality axioms coupling similarities in different domains are associative (in our example above this was the product conjunction).

2.2. Truth values and structures

We work with the set of truth values being the set of reals from the unit interval $[0, 1]$, though we can imagine also a comfortable work with a residuated lattice (or even a substantial weakening of it). For a connective c the corresponding truth value function will be denoted by c^* (i. e. a dot over the very connective).

A conjunction $\&$ has a truth value function a conjunctor $\&^* : [0, 1]^2 \longrightarrow [0, 1]$ and for disjunction $\vee$ a disjunctor $\vee^* : [0, 1]^2 \longrightarrow [0, 1]$ which are both assumed to extend the respective two valued connective and are monotone in both coordinates. Truth function for an n-ary aggregation operator $@^* : [0, 1]^n \longrightarrow [0, 1]$ is monotone and should fulfill $@^*(1, 1, \ldots, 1) = 1$ and $@^*(0, 0, \ldots, 0) = 0$ (note that both conjunctors and disjunctors are special cases of aggregation operators). Truth function for an implication $\rightarrow$ is an implicator $\rightarrow^*: [0, 1]^2 \longrightarrow [0, 1]$ which is nonincreasing in the first and nondecreasing in the second coordinate and extends the two valued implication.

In the language of similarities and equality theory there is no negation, it is therefore enough to deal with logic programs without negation. Herbrand universe U_L^A consists of all ground terms of type A. Having function symbols we are going to interpret them crisp. Herbrand base B_L consists of all ground atoms. Note that so far there is no fuzziness.

An n-ary predicate symbol $p(A_1, \ldots, A_n)$ will be interpreted as a fuzzy subset of

$U_L^{A_1} \times U_L^{A_2} \times \ldots \times U_L^{A_n}$, i.e. as a mapping

$$p^J : U_L^{A_1} \times U_L^{A_2} \times \ldots \times U_L^{A_n} \longrightarrow [0,1].$$

Gluing together all fuzzy predicates we can interpret them all at once by a mapping

$$J : B_L \to [0,1].$$

There is a one to one correspondence between this two ways of representing a structure of the language.

Let $J : B_L \to [0,1]$ be a fuzzy interpretation of our language. The truth value for ground atoms $p \in B_L$ is defined to be $J(p)$. For arbitrary formula φ and an evaluation of all sorts of variables $e^A : Var^A \to U_L^A$ the truth value $J(\varphi)[e]$ is calculated along the complexity of formulas using truth evaluation of connectives c^*. The universal quantifier acts as $J((\forall)\varphi)[e] = \inf\{J(\varphi)[e'] : e' =_x e\}$ where $e' =_x e$ means that e' can differ from e only at x. Finally let the truth value of a formula φ under an interpretation J be the same as that of its generalization and, it does not depend on evaluation:

$$J(\varphi) = J(\forall\varphi) = \inf\{J(\varphi)[e] : e \text{ arbitrary}\}.$$

2.3. Fuzzy theories with equality axioms

For purposes of this paper a fuzzy theory is a partial mapping T assigning formulas numbers from $(0,1]$. Partiality of the mapping T we understand as of being defined constantly zero outside of the domain $dom(T)$. A single axiom we often denote as $(\varphi, T(\varphi))$, e.g. $(\varphi, 0.5), (\psi, 0.9)$.

For purposes of this paper we have to extend axioms of the theory T by axioms describing properties of fuzzy similarities and equality axioms. We have a choice whether to assume the similarity is defined on the whole Herbrand universe for an attribute U_T^A or only on constants and then extended to the whole base by assuming axioms of equality also for function symbols (nevertheless in many real world applications there are no function symbols needed).

Definition 1. For a fuzzy theory T let T_E denotes a fuzzy theory consisting of axioms of T and for every predicate, similarity on a domain of a attribute and a function symbol in our language of following axioms (with truth value 1):

$(s(x,x),1)$
$(s(x,y) \longrightarrow_L s(y,x),1)$
$(s(x,y)\ \&_P\ s(y,z) \longrightarrow_L s(x,z),1)$
$(p(x_1,\ldots,x_n)\ \&_P\ s_1(x_1,y_1)\ \&_P\ \ldots\ \&_P\ s_n(x_n,y_n) \longrightarrow_L p(y_1,\ldots,y_n),1)$.
$(s_1(x_1,y_1)\ \&_P\ \ldots\ \&_P\ s_n(x_n,y_n) \longrightarrow_L s(f(x_1,\ldots,x_n),f(y_1,\ldots,y_n)),1)$

An interpretation J is a model of a theory T if for all formulas $\varphi \in dom(T)$ we have

$$J(\varphi) \geq T(\varphi).$$

2.4. Many valued modus ponens

Starting point of our procedural semantics is the many valued modus ponens which syntactically looks like

$$\frac{(B,x),(B \to A,y)}{(A,\mathcal{C}_{\to}(x,y))}$$

where $\mathcal{C}_{\to}$ is a conjunctor residual to the implicator $\to^{\cdot}$. Note that this conjunctor $\mathcal{C}_{\to}$ evaluating modus ponens with $\to$ need not be a truth value function of any conjunctor in our language.

Agreement 1. For the purposes of this paper and to have easier work with axioms of equality which have truth value 1, we assume our conjunctors evaluating modus ponens fulfill additional property

$$(\forall x)\,(\mathcal{C}(x,1) = x).$$

According to W. Pedrycz [12] and S. Gottwald [6] we can define following properties of functions of two real variables

$\Phi 2(\mathcal{C},\mathcal{I})$ iff $(\forall x)\,(\forall y)\mathcal{C}(x,\mathcal{I}(x,y)) \leq y$,

$\Phi 3(\mathcal{C},\mathcal{I})$ iff $(\forall x)\,(\forall y)\mathcal{I}(x,\mathcal{C}(x,y)) \geq y$.

Note also that $\Phi 2(\mathcal{C},\mathcal{I})$ implies that many valued modus ponens is a sound deduction rule, namely whenever $\mathcal{I} = \to^{\cdot}$ and $J(B) \geq x$ and $J(B \to A) = \mathcal{I}(J(B), J(A)) \geq y$ then

$$\mathcal{C}(x,y) \leq \mathcal{C}(J(B), J(B \to A)) = \mathcal{C}(J(B), \mathcal{I}(J(B), J(A))) \leq J(A).$$

2.5. Declarative semantics of fuzzy logic programming

A formula B is called a body if it is built from atoms using arbitrary conjunctions, disjunctions and aggregations.

A rule is a formula of form $A \leftarrow B$; where B is a body and A is an atom. An atom is also called a fact (typically an element of a Herbrand base B_L). A fuzzy logic program P is a fuzzy theory

$$P : Formulas \longrightarrow [0,1]$$

such that $dom(P) = P^{-1}(0,1]$ is finite and consists of rules and facts.

Let us stress again that rules are implications and not Horn clauses and truth value of a rule is the truth value of the implication. It was necessary to make this distinction because in many valued logic in general $A \leftarrow B$ is not equivalent to $A \vee \neg B$. Moreover in two valued logic two rules with same head $A \leftarrow B$ and $A \leftarrow C$ are equivalent to $A \leftarrow (B \vee C)$. Here in manyvalued logic having two rules equipped with truth values $(A \leftarrow B, x)$, $(A \leftarrow C, y)$ it can correspond to $(A \leftarrow (B \vee_G C), z_{x,y})$ depending on which many valued implications we use in rules. But it can quite well happen that B and C aggregate through an implication to the truth value of A in a way which is neither an conjunction nor a disjunction (in general an aggregation

operator) and that is why we consider such a general form of the body of the rule $(A \leftarrow @(B,C),z)$ (we do not use negation here).

A query (or a goal) is again an atom (positive) intended as a question A? prompting the system (we do not have refutation here).

Substitutions are crisp substitutions in the sense of two valued theory of logic programming.

A pair $(x;\theta)$ consisting of a real number $0 < x \leq 1$ and a substitution θ is a *correct answer* for a program P and a query A? if for arbitrary interpretation $J : B_L \to [0,1]$ which is a models of P we have $J(\forall(A\theta)) \geq x$.

2.6. Procedural semantics of fuzzy logic programming

Procedural semantics is based on a "backward" usage of modus ponens:

$$\begin{array}{cc} (B,x),(B \to A,y) & ?A \\ \hline (A,\mathcal{C}_{\to}(x,y)) & \mathcal{C}_{\to}(B,y) \\ & \mathcal{C}_{\to}(x,y) \end{array}$$

Namely, facing a query $?A$ we have to select a rule with the head unifiable with A. If the selected rule is $(B \to A',y)$ and θ_1 is a crisp mgu of A and A' we already know a part of solution which consists of part of substitution and part of formula for the truth value of the answer, namely we know that truth value will be $\mathcal{C}_{\to}(TV(B),y)$, where $TV(B)$ should be computed. Finding a fact (B',x), B' unifiable with $B\theta_1$ via θ_2 we know that computed answer will consist of substitution $\theta_1\theta_2$ and truth value $\mathcal{C}_{\to}(x,y)$

We define *admissible rules* as follows:

Rule **1.** From $((XA_mY);\vartheta)$ infer $((XC(B,q)Y)\theta;\vartheta \circ \theta)$ if

1. A_m is an atom (called the selected atom)
2. θ is an mgu of A_m and A
3. $P(B \to A) = q$ and B is a (nonempty body).

Rule **2.** From (XA_mY) infer $(X0Y)$ (this is a rule to handle situation when in a disjunction or aggregation an argument is missing. We have argued before why we have to put all possible witnesses for the head into the body coupled by an aggregation operator. This computational rule is to ensure us that the computation will not fail, when a part of witness is missing).

Rule **3.** From $((XA_mY);\vartheta)$ infer $((XrY)\theta;\vartheta \circ \theta)$ if

1. A_m is an atom
2. θ is an mgu of A_m and A
3. $P(A) = r$ (i.e. A is a fact).

Rule **4.** If the word does not contain any predicate symbols rewrite all connectives (&'s and $\vee$'s) to $\&^*$, $\vee^*$. As this word contains only some additional $\mathcal{C}$'s and reals evaluate it (of course the substitution remains untouched). Having in mind the efficiency of our computational model, we restrict ourselves to theories assigning truth values rational numbers and to connectives preserving rationality of truth values.

Let P be a program and A is a goal. A pair $(r;\theta)$ consisting of a (rational) number r and a substitutions θ is said to be a *computed answer* if there is a sequence $G_0,\ldots,G_n$ such that

1. $G_0 = (A, \mathrm{id})$
2. every G_{i+1} is inferred from G_i by one of admissible rules (between lines we do not forget the usual Prolog renaming of variables along derivation).
3. $G_n = (r;\theta')$ and $\theta = \theta'$ restricted to variables of A.

Note that if there are multiple proof path for a particular query and substitution, it means all of them give a compted answer and having soundness the truth value in models is above all of them. So only the proof path with the greatest truth value counts. This gives a place for considerations on threshold computations and proof strategies, but it is out of the scope of the paper.

Example. Assume we know a crisp EDB fact $p(a_1,\ldots,a_n)$ with truth value $TV = 1$ and we would like to know the answer to query $p(b_1,\ldots,b_n)$ (recall motivation from the introduction), a_i's and b_i's are some constants from respective domains. Using equality axiom

$$(p(x_1,\ldots,x_n)\ \&_P\ s_1(x_1,y_1)\ \&_P\ \ldots\ \&_P\ s_n(x_n,y_n) \longrightarrow p(y_1,\ldots,y_n),1)$$

and above procedural semantics, we get the truth value of the answer equal to

$$\mathcal{C}_{\rightarrow}\,(p(a_1,\ldots,a_n)\ \&_P\ s_1(a_1,b_1)\ \&_P\ \ldots\ \&_P\ s_n(a_n,b_n),1) =$$

using our agreement about conjunctors evaluating modus ponens to

$$s_1(a_1,b_1) * \ldots * s_n(a_n,b_n)$$

which can for instance in case $s_1(a_1,b_1) = 0.9$, $s_2(a_2,b_2) = 0.8$ and $s_3(a_3,b_3) = s_n(a_n,b_n) = 1$ be equal to 0.72. Hence having an information about two objects with different but similar names, leads to answer with certain truth value.

Please note that variables in two parts of the equality axiom are disjoint, hence the substitution consists of two parts, one on the query side and one on the side of knowledge from fuzzy logic program (head of a rule or a fact).

We use product conjunctor in handling similarities because it is independent of each other in which attribute there is a partial match, and we model independence by a product conjunction. On the other side in bodies of rules we can use several different connectives.

2.7. Soundness of our semantics

Theorem 1. Every computed answer for a many valued definite fuzzy logic program P and A is a correct answer.

Proof. Is in [17] for propositional case and t-norms based connectives and in full generality in [18], and in principle it follows the classical proof of Lloyd [10] and uses soundness of many valued modus ponens (see Subsection 2.2). □

3. FIXPOINT THEORY AND COMPLETENESS

Consider the lattice

$$F_{P_E} = \{J : J \text{ is a mapping from } B_{P_E} \text{ into } [0,1]\}$$

with coordinatewise lattice ordering.

Definition 2. We define the operator $T_{P_E} : F_{P_E} \longrightarrow F_{P_E}$ by $T_{P_E}(J)(A) = \sup\{r :$ there is a rule $A \leftarrow_i B$ which is a ground instance of some $C \in dom(P_E)$ and $r = \mathcal{C}_i(J(B), P_E(C))\}$

3.1. The fixpoint theorem

Theorem 2. Assume for all implications there is a conjunctor fulfilling $\Phi 2 - 3(\mathcal{C}_i, \rightarrow_i^{\cdot})$. Moreover assume all $\mathcal{C}_i$'s, $\&_i^{\cdot}$'s, $\vee_i^{\cdot}$'s and $@_i$'s are lower semicontinuous. Then

1. T_{P_E} is continuous (i. e. T_{P_E} preserves joins of upward directed sets of interpretations).
2. J is a model of P_E iff $T_{P_E}(J) \leq J$ (hence the minimal fixpoint of T_{P_E} is a model of P_E).

Proof. Is in [17] for propositional case and t-norms based connectives and in full generality in [18]. □

Hence fixpoint theorem works even without any further assumptions on conjunctors (definitely they must not be commutative and associative).

3.2. Completeness of fuzzy definite programs

Theorem 3. Assume for all implications there is conjunctor with $\Phi 2 - 3(\mathcal{C}_i, \mathcal{I}_i)$ and all $\mathcal{C}_i$'s, $\&_i^{\cdot}$'s and $\vee_i^{\cdot}$'s are lower semicontinuous. Then for every correct answer (x, Θ) for P_E and A and for every $\epsilon > 0$ there is a computed answer (r, ϑ) for P_E and A such that $x - \epsilon < r$ and $\Theta = \vartheta\gamma$ (for some γ).

Proof. Is in [17] for propositional case and t-norms based connectives and in full generality in [18]. □

4. SEMANTICS FOR FUZZY SIMILARITY BASED UNIFICATION

Let us recall that in equality axioms we couple similarities with the product conjunction because truth value of the predicate has nothing to do with the need and/or necessity of application of the equality axiom.

Moreover let us recall that conjunctors evaluating modus ponens fulfill $\mathcal{C}(x,1)=x$.

Further let us assume that all interpretations J fulfill the requirement that all $\{J(s_A) : A \in \mathcal{A}\}$ are fixed fuzzy similarities on the whole Herbrand universe U_L^A. This means that in the iteration of the T_{P_E} operator we need not to use axioms of similarity and the axiom of equality for function symbols because they cannot increase the truth degree of any $J(s)$.

Next theorem shows that one step iteration of T_P with equality axiom for predicate symbol suffices, immediate additional application of equality axiom cannot increase the truth value of an atom (of course merging multiple use of equality axiom with the use of other rules can make difference).

Theorem 4. Assume J is an interpretation fulfilling above agreements. Consider

$$J : B_L \longrightarrow [0,1]$$

as a fuzzy logic program, consisting only of ground atoms (facts). Let J_E be the fuzzy theory J extended by similarity axioms and equality axioms. Then

$$T^1_{J_E}(J) \text{ is the fixpoint and hence the minimal model of } J_E.$$

Proof. As said above, we assume all $J(s)$'s are fixed fuzzy similarities, fuzzy theory J consists of axioms $(p, J(p))$. So neither applying similarity axioms nor axioms of type $J(p)$ cannot increase the truth value of ground atoms from the Herbrand base given by J (now considered as interpretation, as a subject of the iteration).

So the only we have to use are equality axioms of predicate calculus with predicates, and namely their repeated use in the T_{P_E} iteration.

Assume $p(a_1,\ldots,a_n)$ and $p(b_1,\ldots,b_n)$ are two ground atoms and that an contribution to the value of

$$T^2_{J_E}(J)\,(p(b_1,\ldots,b_n))$$

looks like

$$T^1_{J_E}(J)\,(p(c_1,\ldots,c_n)) * J(s(b_1,c_1)) * \ldots * J(s(b_n,c_n)).$$

Every contribution to T^1 is of the form $J(p(a_1,\ldots,a_n)) * \Pi_{i=1}^n J(s(a_i,c_i))$ and this together with above expression gives

$$J(p(a_1,\ldots,a_n)) * \Pi_{i=1}^n J(s(b_i,c_i)) * \Pi_{i=1}^n J(s(a_i,c_i))$$

but this is due to the transitivity of similarities less than or equal to

$$J(p(a_1,\ldots,a_n)) * \Pi_{i=1}^n J(s(b_i,a_i)) \leq T^1_{J_E}(J)\,(p(b_1,\ldots,b_n))$$

hence the second iteration cannot increase the truth value of the atom $p(b_1, \ldots, b_n)$, and hence $T^1_{J_E}(J)$ is the fixpoint. □

Note again (as argued before) that handling with similarities is coupled by product conjunction and in the rest of calculation (in bodies of rules) there can be other connectives.

To define the declarative semantics of fuzzy similarity based unification (FSBU) we have first to distinguish between the query side of the unification and between the program side (namely the head or fact entering the unification). Because the unification has to do with semantical consequence and this is one side oriented requirement, and hence we have to distinguish them.

Assume we have two atomary formulas, both variants of the same predicate. Let us denote

$$A^q \text{ the query variant, and}$$

$$A^{h/f} \text{ the head/fact variant of the very predicate.}$$

We are going to define the declarative semantics of fuzzy unification in terms of semantical consequence. It is more or less our choice we want to make our definition of fuzzy unification independent of program in which it is used and depending only on actual values of similarities used.

We can show that it really does not matter whether we work in an equality extension of an empty theory $\emptyset_E$ or it is true in all models of arbitrary theory P.

The only what counts is that

1. values of similarities are fixed, and
2. we have axioms of equality of predicate calculus.

Definition 3. Assume A^q and $A^{h/f}$ are as above and $\{J(s_A) : A \in \mathcal{A}\}$ are fixed similarities.

A triple $(\theta^q, \theta^{h/f}, x)$ consisting of two crisp substitutions and a real number is a *correct fuzzy similarity based unification (FSBU)* of A^q and $A^{h/f}$ wrt fixed similarities $\{J(s_A) : A \in \mathcal{A}\}$, if for every fuzzy logic program P and every model K of P_E which agrees with J on similarities, the truth value of

$$A^q\theta^q \longleftarrow_P A^{h/f}\theta^{h/f} \text{ is at least } x.$$

Now let us rewrite axioms of equality distinguishing the the "query" and "head/fact" side as follows

$$\Big(p^{h/f}(x_1, \ldots, x_n) \,\&_P\, x_1 =_1 y_1 \,\&_P\, \ldots \,\&_P\, x_n =_n y_n \longrightarrow p^q(y_1, \ldots, y_n), 1\Big)$$

in order in our definition of procedural semantics we can define the query part of the substitution and the head/fact part of the substitution.

Definition 4. Assume A^q and $A^{h/f}$ are as above and $\{J(s_A) : A \in \mathcal{A}\}$ are fixed similarities.

The the triple $(\theta^q, \theta^{h/f}, x)$ consisting of two crisp substitutions and a real number is a *computed fuzzy similarity based unification (FSBU)* if $(\theta^q\theta^{h/f}, x)$ is a computed answer for the query A^q wrt the fuzzy program $P_E(J, A^{h/f})$ consisting of all fuzzy facts

$$\{J(s_A) : A \in \mathcal{A}\}$$

and two more axioms

$$\left(p^{h/f}(x_1, \ldots, x_n)\ \&_P\ x_1 =_1 y_1\ \&_P\ \ldots\ \&_P\ x_n =_n y_n \longrightarrow p^q(y_1, \ldots, y_n), 1\right)$$

and

$$(A^{h/f}, 1).$$

Note that J. F. Baldwin in [2] and [3] uses implicit rules to perform unification between fuzzy sets, in a manner similar to this definition.

The advantage of this definitions is that they use well developed theory of fuzzy logic programming (with crisp unification) and hence can be easily integrated into the formal model of fuzzy logic programming with fuzzy similarity based unification. Moreover they offer a solution to the problem of deduction with different names of objects as mentioned in the beginning.

Using results on soundness and completeness of fuzzy logic programming, especially the fixpoint theorem, the minimal model obtained by a one step iteration of the T_P operator, we have the following

Theorem 5. With assumptions as in the above definitions: every computed FSBU is a correct one and every correct FSBU can be computed with a truth value at least as big as that of the correct one and with a substitution more general than that of the correct one.

5. CONCLUSIONS

Besides many other approaches in literature offering deduction with similarity (see e. g. [1, 16] and references there) we based our approach on the requirement of having both procedural and declarative semantics of fuzzy unification and proven soundness and completeness. In the future it would be interesting to have comparison of the computational aspects of all of these approaches.

ACKNOWLEDGEMENT

Research supported by Grants VEGA 1/7557/20 and 1/7555/20.

(Received June 13, 2000.)

REFERENCES

[1] F. Arcelli, F. Formato, and G. Gerla: Similitude-based unification as a foundation of fuzzy logic programming. In: Logic Programming and Soft Computing in AI (T. P. Martin and F. Arcelli Fontana, eds.), Research Studies Press, Wiley, New York 1998.
[2] J. F. Baldwin: Support logic programming. In: Fuzzy Sets – Theory and Applications (A. Jones, ed.), D. Reidel 1986, pp. 133–170.
[3] J. F. Baldwin: The management of fuzzy and probabilistic uncertainties for knowledge based systems. In: Encyclopedia of AI. Second edition (S. A. Shapiro, ed.), Wiley, New York 1992, pp. 528–537
[4] D. Dubois, J. Lang, and H. Prade: Fuzzy sets in approximate reasoning, Part 2: Logical approaches. In: Foundations of Fuzzy Reasoning. Special Memorial Volume; 25 years of fuzzy sets: Attribute to Professor Lotfi Zadeh. First issue (I. B. Turksen, D. Dubois, H. Prade, and R. R. Yager eds.), Fuzzy Sets and Systems *40* (1991), 203–244.
[5] A. Dekhtyar and V. S. Subrahmanian: Hybrid probabilistic programs. J. Logic Programming *43* (2000), 187–250
[6] S. Gottwald: Fuzzy Sets and Fuzzy Logic. Vieweg, Wiesbaden 1993.
[7] P. Hájek: Metamathematics of Fuzzy Logic. Kluwer, Dordrecht 1998.
[8] M. Kifer and V. S. Subrahmanian: Theory of generalized annotated logic programming and its applications. J. Logic Programming *12* (1992), 335–367
[9] P. Kriško, P. Marcinčák, P. Mihók, J. Sabol, and P. Vojtáš: Low retrieval remote querying dialogue with fuzzy conceptual, syntactical and linguistical unification. In: Proc. FQAS'98 Flexible Query Answering Systems (T. Andreasen et al, eds., Lecture Notes in Computer Science 1495), Springer Verlag, Berlin 1998, pp. 215–226.
[10] J. W. Lloyd: Foundations of Logic Programming. Springer Verlag, Berlin 1987.
[11] A. Martelli and U. Montanari: An efficient unification algorithm. ACM Trans. Programming Languages and Systems *4* (1982), 258–282.
[12] W. Pedrycz: Fuzzy Control and Fuzzy Systems. Report 82/14, Dept. Math., Delft Univ. of Technology.
[13] F. E. Petry: Fuzzy Databases – Principles and Applications. Kluwer, Dordrecht 1996.
[14] J. A. Robinson: A machine-oriented logic based on the resolution principle. J. Assoc. Comp. Mach. *12* (1965), 23–41.
[15] E. van Emden: Quantitative deduction and its fixpoint theory. J. Logic Programming *1* (1986), 37–53.
[16] H. E. Virtanen: Linguistic logic programming. In: Logic Programming and Soft Computing (T. P. Martin and F. Arcelli Fontana, eds., Research Press Studies Lim.), Wiley, New York 1998.
[17] P. Vojtáš: Fuzzy reasoning with tunable t-operators. J. Advanced Comp. Intelligence 2 Fuji Press (1998), 121–127.
[18] P. Vojtáš: Fuzzy logic programming. Submitted to Proc. Workshop on Fuzzy Logic at FSTA, for Fuzzy Sets and Systems.
[19] P. Vojtáš: Uncertain reasoning with floating connectives. In: Proc. AIT'96 Artificial Intelligence Techniques Brno (J. Žižka, ed.), Technical University Brno, PC-Dir Publ. 1996, pp. 31–40.

Doc. RNDr. Peter Vojtáš, DrSc., Department of Computer Science, Faculty of Science, P. J. Šafárik University, Mathematical Institute, Slovak Academy of Sciences, Jesenná 5, 041 54 Košice. Slovak Republic.
e-mail: vojtas@kosice.upjs.sk