



ELSEVIER

Fuzzy Sets and Systems 124 (2001) 361–370

FUZZY
sets and systems

www.elsevier.com/locate/fss

Fuzzy logic programming [☆]

Peter Vojtáš^{a,b,*}

^aDepartment of Computer Science, Faculty of Science, P.J. Šafárik University, Jesenná 5, 041 54 Košice, Slovak Republic

^bInstitute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2, 180 00 Praha, Czech Republic

Received May 2000; received in revised form January 2001

Abstract

In this paper we consider the theory of fuzzy logic programming without negation. Our results cover logical systems with a wide variety of connectives ranging from t-norm and conorms, through conjunctors and disjunctors and their residuals to aggregation operators. Rules of our programs are many valued implications. We emphasize, that in contrast to other approaches, our logic is truth functional, i.e. according to P. Hájek, we work in fuzzy logic in narrow sense. We prove the soundness and the completeness of our formal model. We deal with applications to threshold computation, abduction, fuzzy unification based on similarity. We show that fuzzy unification based on similarities has applications to fuzzy databases and flexible querying. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Fuzzy logic programming; Fuzzy unification; Abduction; Databases; Querying; Similarities; Threshold computation

1. Introduction

1.1. Pavelka completeness and a great variety of connectives

The aim of this paper is to build a formal model for fuzzy logic programming without negation. We

work in a truth functional fuzzy logic in narrow sense (according to Hájek [7]). This is the main difference with other approaches (annotated, signed, hybrid, possibilistic, probabilistic logical systems). Our rules are many valued implications (IF–THEN rule based fuzzy systems). Truth value of the formula is evaluated along the complexity of formula using truth functions of connectives.

Pavelka [15] showed that arbitrary fuzzy theory over his logic (which is in fact Łukasiewicz logic with truth constants) the infimum of truth degree of a formula φ in all models of the theory equals supremum of degrees of graded proofs of φ in that theory and also showed that this does not hold in full generality for fuzzy theories over other logical systems. (A

[☆] Research supported by grants VEGA 1/7557/20 and GAČR 201/00/1489.

* Corresponding author. Department of Computer Science, Faculty of Science, P.J. Šafárik University, Jesenná 5, 041 54 Košice, Slovak Republic.

E-mail addresses: vojtas@kosice.up.js.sk, vojtas@cs.cas.cz (P. Vojtáš).

presentation of Pavelka's results together with a simplification of his system can be found in [7].)

We prove a Pavelka type result for a wide variety of logical systems, restricting to simple theories namely fuzzy logic programs without negation.

We consider the system with a widest possibility of many valued connectives. This enables our system to cover a great variety of applications where connectives are learned from data. Especially, it enables to describe different user environments and/or stereotypes.

Our motivating example was inspired by a hotel reservation system presented by Naito et al. [15]. They described a system of rules with connectives (aggregation operators) which is learned by a neural network. Input of the neural network is user's preferences on a sample of hotels. Output are parameters of a parametrized family of t -norms and t -conorms and a parameter of a convex combination of them.

Earlier, Gupta and Qi [6] wrote: "Theoretical and experimental studies have indicated that some t -operators work better in some situations, especially in the context of decision making process than min and max operators. In fact the choice of an operation is always a matter of context, and it mostly depends on the real world problem which is to be modelled. It is appropriate, therefore to use the general concept of t -operators in the modeling of decision making process, so as to provide more options and flexibility for the selection of t -operators that may be better suited for a given problem."

1.2. Example

In our example, we describe the knowledge base using logic programming notation. The rules are true implications. We work in a fuzzy logic where the connectives have many valued truth functions. @ is an aggregation operator (e.g. arithmetic mean, weighted sum or a monotone function learned from data). Aggregation operators are very useful in such systems because they enable us to describe increasing fulfilment of positive witnesses to user requirements.

```
hotel_reservation(Business_Location, Time,
                 Hotel) ←P
@((near_to(Business_Location, Hotel),
  cost_reasonable(Hotel, Time),
  building_is_fine(Hotel))). with truth value = 0.8
```

```
near_to(Business_Location, Hotel) ←P
location(Hotel, Hotel_Location) &G
distance(Hotel_Location, Business_Location,
         Distance) &G
close(Distance). with truth value = 1.
```

Rules for `cost_reasonable` and `building_is_fine` are defined similarly. Here `location`, `distance`, `price` and `building` are crisp relational data. `Close`, `reasonable_price` and `age_reasonable` are fuzzy relations. They can look like

$$close(x) = \max\left(0, 1 - \frac{x}{50}\right),$$

$$reasonable_price(x) = \max\left(0, 1 - \frac{x}{1000}\right),$$

$$age_reasonable(x) = \max\left(0, 1 - \frac{2000 - x}{15}\right).$$

As mentioned above, the aggregation can be $@(x, y, z) = (x + y + z)/3$ (which equally aggregates quality of distance, price and building) or $@(x, y, z) = (3x + y + 2z)/6$ (which gives the highest preference for distance and then the second important is the building quality while the price is least significant). The fuzzy connectives are $\&_G(x, y) = \min(x, y)$ and $\rightarrow(x, y) = \min(1, y/x)$. Note that the first rule is equipped with a truth value 0.8.

1.3. Why do we need a formal model?

The main task of this paper is to build a procedural and declarative semantics for fuzzy logic programming and to prove their soundness and completeness. In our opinion it is important to have a sound and complete semantics because it enables us to compare results of computation with real world data. If there exists a difference between computation and real world data, we know that the reason is not in the system itself but in the description of the real world situation. In this case we should tune either connectives, fuzzy predicates or the rule base. Of course, we have to take into account that the system might not be truth functional. In this case it is not appropriate for a description using logic. Thus fuzzy logic programming is not the only solution but we try to extend its applicability.

Moreover, we can extend the system by threshold computation. It will be sound and complete.

Using this we can define the fuzzy logic programming abduction problem and show its soundness and completeness.

Axioms describing properties of fuzzy similarities and properties of predicate calculus with equality are fuzzy IF–THEN rules. When we incorporate them in our rule base we achieve a system which has fuzzy unification based on similarities.

Moreover, we develop the full fuzzy analogy of fixpoint theory which has consequences for fuzzy databases.

We also mention applications for a theory of fuzzy resolution which use knowledge bases consisting of clauses (not implications).

All this can be done without any additional experimental work just on the base of our formal model for fuzzy logic programming.

2. Truth functional fuzzy logic (in narrow sense)

2.1. Language

Our language is a many sorted predicate language with or without function symbols (we identify sorts of variables and attributes here). 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). A predicate $p(A_1, \dots, A_n)$ 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 on the domain \mathcal{C}^A . Similarly, function symbols have arity defined by a list of attributes for input and an attribute for the output.

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

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

The syntactical level is not touched by the many valuedness of truth functions and fuzzy structures. Nevertheless, since we have several different connec-

tives 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.

2.2. Truth values and structures

The set of truth values in this paper is the set of real numbers from the unit interval $[0, 1]$. For a connective c the corresponding truth value function will be denoted by c^\cdot (i.e. a dot over the very connective). c^\cdot is according to the arity of the connective a function from $[0, 1]^{ar(c)}$ into $[0, 1]$ (usually preserving rationality).

A truth function for a conjunction $\&$ is a conjunctor $\&^\cdot: [0, 1]^2 \rightarrow [0, 1]$ and for disjunction \vee a disjunctive $\vee^\cdot: [0, 1]^2 \rightarrow [0, 1]$ which are assumed to extend the respective two valued connective and are monotone in both coordinates (no associativity, symmetry or boundary conditions are assumed). This enables us to work with approximations of connectives and/or with connectives learned from data. The truth function for an n -ary aggregation operator $@^\cdot: [0, 1]^n \rightarrow [0, 1]$ should be a nonmonotone and fulfill $@^\cdot(1, 1, \dots, 1) = 1$ and $@^\cdot(0, 0, \dots, 0) = 0$. The truth function for an implication \rightarrow is an implicator $\rightarrow^\cdot: [0, 1]^2 \rightarrow [0, 1]$ which is nonincreasing in the first and nondecreasing in the second coordinate and extends the two valued implication.

Recall some classical connectives

The Łukasiewicz connectives:

$$\&_L^\cdot(x, y) = \max(0, x + y - 1),$$

$$\rightarrow_L^\cdot(x, y) = \min(1, 1 - x + y),$$

$$\vee_L^\cdot(x, y) = \min(1, x + y).$$

The Gödel intuitionistic connectives:

$$\&_G^\cdot(x, y) = \min(x, y),$$

$$\rightarrow_G^\cdot(x, y) = y \text{ if } x > y \text{ else } 1,$$

$$\vee_G^\cdot(x, y) = \max(x, y).$$

The product logic:

$$\&_p^\cdot(x, y) = x \cdot y,$$

$$\rightarrow_p^\cdot(x, y) = \min(1, y/x),$$

$$\vee_p^\cdot(x, y) = x + y - x \cdot y.$$

Since we are dealing with logic programs without negation, we skip here arbitrary interpretations and we base our declarative semantics only on fuzzy Herbrand interpretations. We follow closely Lloyd’s presentation and even notation [10]. Herbrand universe of sort A , denoted by U_L^A , consists of all ground terms of type A . Having function symbols we are going to interpret them as crisp (although in many database applications where the knowledge base is in a normal form, there are no function symbols at all). Herbrand base B_L consists of all ground atoms. Note that so far there is no fuzziness.

An n -ary predicate symbol $p(A_1, \dots, A_n)$ will be interpreted as a fuzzy subset of $U_L^{A_1} \times U_L^{A_2} \times \dots \times U_L^{A_n}$, i.e. as a mapping from

$$p^f : U_L^{A_1} \times U_L^{A_2} \times \dots \times U_L^{A_n} \rightarrow [0, 1].$$

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

$$f : B_L \rightarrow [0, 1].$$

There is a one-to-one correspondence between these two ways of representing a structure of the language. Namely, having the value $p^f(t_1, \dots, t_n)$ for some constant terms in appropriate domains, we can define $f_{p^f}(p(t_1, \dots, t_n)) = p^f(t_1, \dots, t_n)$ and vice versa $p_{f^f}^f$. Note that $p(t_1, \dots, t_n)$ is a syntactical object, i.e. a word in the alphabet of our language.

Let $f : B_L \rightarrow [0, 1]$ be a fuzzy interpretation of our language. The truth value for ground atoms $p \in B_L$ is defined to be $f(p)$. For arbitrary formula φ and an evaluation of all sorts of variables $e^A : Var^A \rightarrow U_L^A$ the truth value $f(\varphi)[e]$ is calculated along the complexity of formulas using truth value functions of connectives c^* . The universal quantifier acts as $f((\forall x)\varphi)[e] = \inf\{f(\varphi)[e'] : e' =_x e\}$ 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 f be the same as that of its generalization and not depend on evaluation (here $\forall\varphi$ means universal quantification of all variables with free occurrence in φ):

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

2.3. Fuzzy theories

Developing a fuzzy logic system we should be more careful and to distinguish between syntactical and se-

mantical part of the system (some stereotypes from two valued case do not apply here). Especially, we will have no logical axioms. This is caused by the fact that we work in a very general situation with almost arbitrary connectives and we would like our computed query answers to apply to real world situation. So there is no time to look for 1-tautologies to extend the capabilities of our system. We work only with theories describing the application domain.

For the purposes of this paper a fuzzy theory is a partial mapping P assigning to formulas rational numbers from $(0, 1]$. We do not depend on this assumption on rational numbers, from the point of view of complexity it is quite reasonable. We understand that partiality of the mapping P as being defined constantly zero outside of the domain $dom(P)$. A single axiom is often denoted as $(\varphi, P(\varphi))$, i.e. the ordered tuple—the formula and the (axiomatically assigned) truth value (e.g. by an expert, by a learning procedure, etc.).

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

$$f(\varphi) \geq P(\varphi).$$

This means that the truth value assigned by the axiom is understood as a lower bound of truth values in structures which are models.

2.4. Many valued modus ponens

For our application we have to decide what are the reasonable implications and how do we evaluate logic programming computation with uncertain rules. Our starting point is the many valued modus ponens (see e.g. [7,15]), which syntactically looks like

$$\frac{(B, x), (B \rightarrow A, y)}{(A, mp_{\rightarrow}(x, y))}$$

where mp_{\rightarrow} is a function calculating the truth value of the answer.

The soundness of many valued modus ponens semantically means that whenever f is an interpretation such that $f(B) \geq x$ (i.e. f is a model of (B, x)) and $f(B \rightarrow A) = \rightarrow^*(f(B), f(A)) \geq y$, (i.e. f is a model of $(B \rightarrow A, y)$) then $f(A) \geq mp_{\rightarrow}(x, y)$, i.e. the truth value of A in all models of mentioned two axioms is guaranteed to be at least the computed degree.

Even without knowing what properties \rightarrow^* should fulfill, we argue that mp_{\rightarrow} should fulfill $mp_{\rightarrow}(0, 0) =$

$mp_{\rightarrow}(0, 1) = mp_{\rightarrow}(1, 0) = 0$ (because if one of the premises of modus ponens is false then a nonzero truth value of the answer cannot be guaranteed). Similarly, $mp_{\rightarrow}(1, 1) = 1$, because we would like our system to extend the two valued modus ponens.

Moreover, if we know premises with higher truth value then the truth value of conclusion should not be smaller. That is mp_{\rightarrow} should be nondecreasing in both coordinates.

These are exactly the properties of a conjunctor. Denote mp_{\rightarrow} by $\mathcal{C}_{\rightarrow}$. Note that this conjunctor $\mathcal{C}_{\rightarrow}$, evaluating modus ponens with \rightarrow need not be a truth value function of any conjunction in our language. Moreover, since we do not assume commutativity (symmetry) of conjunctors \mathcal{C} evaluating modus ponens and to be sure that in proofs we do not confuse position of variables denoting truth value of the fact and the rule, let us make the following agreement:

In the many valued modus ponens

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

the position of variables of the conjunctor \mathcal{C} , which is used to estimate the lower bound for the truth value of the head of the implication, has fixed meaning and are intended as:

the first variable of $\mathcal{C}(x, \cdot)$ is the truth value of the fact entering the modus ponens (here (B, x)),

the second variable of $\mathcal{C}(\cdot, y)$ is the truth value of the rule entering the modus ponens (here $(B \rightarrow A, y)$), schematically:

$$\mathcal{C}_{\rightarrow}(\text{Truth_value_of_the_fact}, \text{Truth_value_of_the_rule}).$$

Position/order of variables of \mathcal{I} is also fixed and is intended as

$$\mathcal{I}(\text{Truth_value_of_the_body}, \text{Truth_value_of_the_head}).$$

Assertion $MP(\mathcal{C}, \mathcal{I})$ means: \mathcal{C} is a conjunctor, \mathcal{I} is an implicator and $\rightarrow = \mathcal{I}$; then from $f(B) \geq x$ and $f(B \rightarrow A) = \mathcal{I}(f(B), f(A)) \geq y$ follows that $f(A) \geq \mathcal{C}(x, y)$.

According to Pedrycz [16] and Gottwald [5] we can define the following properties of functions of two real variables:

$\Phi 1(\mathcal{I})$ iff \mathcal{I} is nonincreasing in the first and nondecreasing in the second coordinate (note, every implicator fulfills this),

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

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

Moreover, having \mathcal{I} we define

$$\mathcal{C}_{\mathcal{I}}(f, r) = \inf\{h: \mathcal{I}(f, h) \geq r\}.$$

First note that names of variables are mnemonic (according to the above agreement on the position and intended meaning of variables): f as the truth value of the fact coming to MP (in \mathcal{I} it plays the role of body), r as the truth value of the rule and h as the truth value of the head. Note also that $\mathcal{C}_{\mathcal{I}}$ is always a conjunctor. $\Phi 2(\mathcal{C}_{\mathcal{I}}, \mathcal{I})$ always holds true. If \mathcal{I} is right continuous in the second variable, then $\Phi 3(\mathcal{C}_{\mathcal{I}}, \mathcal{I})$ holds. For classical connectives we have $\mathcal{C}_{\rightarrow_L} = \&_L$, $\mathcal{C}_{\rightarrow_G} = \&_G$ and $\mathcal{C}_{\rightarrow_P} = \&_P$.

Analogously for \mathcal{C} define

$$\mathcal{I}^{\mathcal{C}}(b, h) = \sup\{r: \mathcal{C}(b, r) \leq h\}.$$

Note that $\mathcal{I}^{\mathcal{C}}$ is an implicator, $\Phi 3(\mathcal{C}, \mathcal{I}^{\mathcal{C}})$ always holds and if \mathcal{C} is left continuous in the second (rule typed) coordinate, then $\Phi 2(\mathcal{C}, \mathcal{I}^{\mathcal{C}})$ is also true. Note $\mathcal{I}^{\&_L} = \rightarrow_L$, $\mathcal{I}^{\&_G} = \rightarrow_G$ and $\mathcal{I}^{\&_P} = \rightarrow_P$.

Moreover, note that whenever $\mathcal{C} \leq \mathcal{C}_{\mathcal{I}}$ then $\Phi 2(\mathcal{C}, \mathcal{I})$ and $\mathcal{I} \geq \mathcal{I}^{\mathcal{C}}$ imply $\Phi 3(\mathcal{C}, \mathcal{I})$.

Note also that $\Phi 2(\mathcal{C}, \mathcal{I})$ implies $MP(\mathcal{C}, \mathcal{I})$, namely whenever $\mathcal{I} = \rightarrow^*$ and $f(B) \geq x$ and $f(B \rightarrow A) = \mathcal{I}(f(B), f(A)) \geq y$ then

$$\begin{aligned} \mathcal{C}(x, y) &\leq \mathcal{C}(f(B), f(B \rightarrow A)) \\ &= \mathcal{C}(f(B), \mathcal{I}(f(B), f(A))) \leq f(A). \end{aligned}$$

$\mathcal{C}_{\mathcal{I}}$ is the largest sound evaluation of modus ponens. As a corollary of completeness we will get that $\mathcal{C} \leq \mathcal{C}_{\mathcal{I}}$ and $\mathcal{I} \geq \mathcal{I}^{\mathcal{C}}$ imply $\mathcal{C} = \mathcal{C}_{\mathcal{I}}$.

3. Semantics of fuzzy logic programming

3.1. Declarative semantics

A formula B is called a body if it is built from atoms using arbitrary conjunctions, disjunctions and aggregations. For the purposes of description of procedural semantics we will use for a while prefix notation in the

body. Typically, a body looks like

$$\vee([\&_1(B_1, @_2(B_2, B_3)), [\&_3(C_1, @_4(C_2, C_3))]).$$

Warning: The usual Prolog notation does not apply. A comma “,” in the body does not denote a conjunction, it only separates inputs of a connective when written in a prefix notation.

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

$$P : \text{Formulas} \rightarrow [0, 1] \cap Q$$

such that $\text{dom}(P) = P^{-1}(0, 1]$ is finite and consists of rules and facts. A query (or a goal) is again an atom (positive) intended as a question $A?$ prompting the system (we do not have refutation here).

For the beginning, substitutions are crisp substitutions.

Definition 1. 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 $f : B_L \rightarrow [0, 1]$, which is a model of P , we have $f(\forall(A\theta)) \geq x$.

3.2. Procedural semantics

First, we define admissible rules which act on tuples of words in the alphabet L_P^R and substitutions. L_P^R is the disjoint union of alphabets of the language of the program $\text{dom}(P)$ enlarged by names for $\&$, \vee , $@$ and reals. Moreover, for every implication \rightarrow we have in L_P^R a conjunctor \mathcal{C} , intended for evaluating modus ponens with \rightarrow . Note that neither of $\mathcal{I}, \rightarrow, \rightarrow'$ is in L_P^R .

Definition 2. We define *admissible rules* as follows:

Rule 1: From $((XA_mY); \vartheta)$ infer $((X\mathcal{C}(B, q)Y)\theta; \vartheta \circ \theta)$ if

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

Rule 2: From (XA_mY) infer $(X\emptyset Y)$ (this is a rule to handle situation when in a disjunction or aggregation an argument 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).

Definition 3. Let P be a program and A a goal. A pair $(r; \theta)$ consisting of a (rational) number r and a substitution θ is said to be a *computed answer* if there is a sequence G_0, \dots, G_n such that

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

Example of a computation: Consider a program P with rules (in propositional logic; the unification and substitution stuff is the same as in classical Prolog). Connectives are in prefix notation:

$$A \leftarrow \vee([\&_1(B_1, @_2(B_2, B_3)),$$

$$[\&_3(C_1, @_4(C_2, C_3))]).cf = a,$$

$$B_1 \leftarrow_1 \&_3(D_1, D_2).cf = b_1,$$

$$C_2 \leftarrow_2 @_2(E_1, E_2).cf = c_2$$

and the database

$$B_2.cf = b_2, \quad B_3.cf = b_3, \quad C_1.cf = c_1,$$

$$C_3.cf = c_3, \quad D_1.cf = d_1, \quad D_2.cf = d_2,$$

$$E_1.cf = e_1, \quad E_2.cf = e_2.$$

The question (consultation) $A?$ leads to a computation which resembles classical Prolog, we have to just remember (via a bookkeeping) truth values for the final evaluation of the confidence of the answer (see also

graded proofs of Pavelka [15] and Hájek [7]):

$$\mathcal{C}(a, \vee([\&_1(B_1, @_2(B_2, B_3))], [\&_3(C_1, @_4(C_2, C_3))]))$$

$$\mathcal{C}(a, \vee([\&_1(\mathcal{C}_1(b_1, \&_3(D_1, D_2)), @_2(B_2, B_3))], [\&_3(C_1, @_4(C_2, C_3))]))$$

$$\mathcal{C}(a, \vee([\&_1(\mathcal{C}_1(b_1, \&_3(D_1, D_2)), @_2(B_2, B_3))], [\&_3(C_1, @_4(\mathcal{C}_2(c_2, @_2(E_1, E_2)), C_3))]))$$

...

$$\mathcal{C}(a, \vee^*([\&_1^*(\mathcal{C}_1(b_1, \&_3^*(d_1, d_2)), @_2^*(b_2, b_3))], [\&_3^*(c_1, @_4^*(\mathcal{C}_2(c_2, @_2^*(e_1, e_2)), c_3))]))$$

what is the truth value of the answer YES to the query A ?

3.3. Soundness of our semantics

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

Proof. Is similar to that of [10]. Take a goal A with a computation of length $k + 1$ starting with a rule $P(A \leftarrow B) = r$. Suppose that the result holds for all computed answers due to computations of length $\leq k$. For each atom D from the body B there is a computation of length $\leq k$, hence computed answer $d \leq f(D)$ in every model of P . But then $f(B) \geq b$, where b is the computed answer for the whole body. This is because conjunctions, disjunctions and aggregations are monotone in both coordinates. Hence, f being a model of P means $f(A \leftarrow B) \geq P(A \leftarrow B) = r$ and by the soundness of modus ponens we get

$$f(A) \geq \mathcal{C}(b, r). \quad \square$$

4. Fixpoint theory and completeness

Consider the lattice

$$F_P = \{f: f \text{ is a mapping from } B_P \text{ into } [0, 1]\}$$

with coordinatewise lattice ordering.

Definition 4. We define the operator $T_P: F_P \rightarrow F_P$ by $T_P(f)(A) = \sup\{r: \text{there is a rule } A \leftarrow_i B \text{ which is a ground instance of some } C \in \text{dom}(P) \text{ and } r = \mathcal{C}_i(f(B), P(C))\}$.

4.1. The fixpoint theorem

Theorem 2. *Assume that all implications fulfill $\Phi 1 - 3(\mathcal{C}_i, \rightarrow_i)$. Moreover, assume that all \mathcal{C}_i 's, $\&_i$'s, \vee_i 's and $@_i$'s are lower semicontinuous. Then*

1. T_P is continuous (i.e. T_P preserves joins of upward directed sets of interpretations).
2. f is a model of P iff $T_P(f) \leq f$ (hence the minimal fixpoint of T_P is a model of P).

Proof. (1) The continuity of T_P is straightforward, using monotonicity and lower continuity of all connectives in the body and conjunctors evaluating modus ponens.

(2) Denote $\rightarrow_i^* = \mathcal{I}_i$. Assume that f is a model of P . For an $A \leftarrow_i B$ a ground instance of some $C \in \text{dom}(P)$ we would like to show that

$$f(A) \geq \mathcal{C}_i(f(B), P(C)).$$

We have

$$f(A \leftarrow_i B) \geq f(C) \geq P(C),$$

the first inequality holds because $A \leftarrow_i B$ is a ground instance of C , the second, because f is a model of P . By $(\Phi 1 - 2)$ we have

$$\begin{aligned} \mathcal{C}_i(f(B), P(C)) &\leq \mathcal{C}_i(f(B), f(A \leftarrow_i B)) \\ &= \mathcal{C}_i(f(B), \mathcal{I}_i(f(B), f(A))) \\ &\leq f(A). \end{aligned}$$

Now assume $T_P(f) \leq f$. It suffices for all ground instances $A \leftarrow_i B$ of C to show $f(A \leftarrow_i B) \geq P(C)$. But

$$f(A) \geq T_P(f)(A) \geq \mathcal{C}_i(f(B), P(C))$$

gives (by $\Phi 1$ and the above)

$$\begin{aligned} f(A \leftarrow_i B) &= \mathcal{I}_i(f(B), f(A)) \\ &\geq \mathcal{I}_i(f(B), \mathcal{C}_i(f(B), P(C))) \geq P(C). \end{aligned}$$

The last inequality is $\Phi 3$.

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

4.2. Completeness of fuzzy definite programs

Theorem 3. Assume for all implications we have $\Phi_1 - 3(\mathcal{C}_i, \mathcal{I}_i)$ and all \mathcal{C}_i 's, $\&_i$'s and \forall_i 's are lower semicontinuous. Then for every correct answer (x, Θ) for P and A and for every $\varepsilon > 0$ there is a computed answer (r, ϑ) for P and A such that $x - \varepsilon < r$ and $\Theta = \vartheta\gamma$ (for some γ).

Proof. Having the previous theorem the proof is very similar to the classical proof of the completeness of SLD resolution.

The previous theorem practically states the completeness for ground instances of atoms (similarly as in [10]), because the correct answer for a ground query A holds also in the minimal model which is a countable iteration of the T_P operator starting from the 0-interpretation. Now having a positive ε there is an n such that $T_P^n(0)(A) > T_P^\omega(0) - \varepsilon$. Looking for the contributions to the value of $T_P^n(0)(A)$, there is one also greater than $T_P^\omega(0) - \varepsilon$, this is obtained through the application of a rule or a fact, in any case we can trace the computation backwards. The only difference from the classical case is that a zero value of an atom can appear in a nonzero value of a disjunction or aggregation (that is why we need the admissible rule 2).

To extend it for general case we can extend the language by constants c_X for every variable and define the truth value of a formula with c_X as the truth value of the formula with universally quantified variable X , that is as the infimum of truth values on all constants. Such a structure of the language is again a model of the theory P , because where in the theory there was a free variable truth value is computed as though it was universally quantified, hence all values on constants are greater or equal and hence also on c_X . Hence, the validity on ground atoms directly follows from the fixed point theorem and using the lifting lemma and Mgu lemma of [10] we obtain the full result.

5. Applications

5.1. Generalization of van Emden's cut [2]—computations with a threshold

Having a conjunctive \mathcal{C} for the evaluation of modus ponens we define the function $\mathcal{E}_{\mathcal{C}}$ (\mathcal{C} tries to be

mnemonic to suggest this function counts estimates or expectations of the development of a branch in the search tree). The intended meaning of variables is $\mathcal{E}_{\mathcal{C}}(\text{rule}, \text{threshold})$:

$$\mathcal{E}_{\mathcal{C}}(r, t) = \inf\{f: \mathcal{C}(f, r) \geq t\}.$$

As $\mathcal{C}(x, y) \leq \min(x, y)$ the function \mathcal{E} is not defined for values $\text{rule} < \text{threshold}$. Except of $\mathcal{E}(0, 0) = 0$ it fulfills all properties of an implicator and properties dual to Φ 's can be introduced. The most important application of \mathcal{E} is the following: Assume I am interested whether there is an answer to A with truth value not less than some threshold t . Moreover assume that my computation selects a rule with truth value r . Then the new threshold for cut is $\mathcal{E}_{\mathcal{C}}(r, t)$.

The situation is even more interesting if the new threshold is larger than the previous one, i.e. $\mathcal{E}_{\mathcal{C}}(r, t) > t$. This is for instance the case when $\mathcal{C}(t, r) < t$ and \mathcal{C} is right continuous in the first variable. In cases when the iteration \mathcal{C}^n tends to zero we can estimate the depth of the search tree according to the size of the threshold and the highest truth value of a rule (assuming all are below 1).

5.2. Fuzzy logic programming abduction

A fuzzy logic abduction problem was defined in [22] and a computational model similar to the one above was given. The only difference is that the abduction does not fail when there is no fact or head unifiable with a selected atom. Instead of this it generated another constraint to the set of hypotheses and with a linear optimization algorithm finds a cheapest explanation for the abduction observation.

5.3. Similarities and the problem of fuzzy unification

The formal model for fuzzy logic programming with crisp unification is a base for a model for fuzzy unification. In the case if we base our fuzzy unification on similarities we can do the following. We express properties of fuzzy similarities and axioms of predicate calculus with equality (which are luckily rules of our type, with truth value 1) as additional rules of our program P . Then the fixpoint theorem and minimal model iteration gives a model of this extended theory. So similarity based fuzzy unification is nothing else

as what the T_P operator does, closure of crisp unification under similarity and equality axioms. Since T_P^ω is also a model of this extended theory, we have a sound and complete model of fuzzy unification (for details see [23]).

5.4. Fuzzy databases and flexible querying

The T_P operator is directly connected with data model which evaluates rules, starting from a crisp EDB, via fuzzy IDB relations. The fuzzy relational algebra was described in [26] and above completeness is guaranteeing the commensurability of logic and fuzzy relational algebra. Moreover, similarity based fuzzy unification is the base of a flexible query answering system (see [25]).

5.5. Fuzzy resolution with true clauses

In [19] we used an observation that the truth function for $\neg A \vee B$ is an implicator to get a sound and complete model of resolution modus ponens (extended to full resolution in [20]). The main point here is that we do not need t-norms and their residuals but our theory developed here works for (almost) arbitrary conjunctions and implicators.

5.6. Fuzzy logic paradox

Many authors noticed the following paradoxical behavior of many valued truth functional logical systems:

Assume A is a proposition with truth values in a structure

$$f(A) = f(\neg A) = \frac{1}{2}.$$

Then the following holds:

$$\begin{aligned} f(A \& A) &= \& \cdot (f(A), f(A)) \\ &= \& \cdot (f(A), f(\neg A)) = f(A \& \neg A), \end{aligned}$$

which is never fulfilled in real world applications. Our solution to this is that it is wrong to describe conjunction of A and A and with $\neg A$ with the same conjunction. Our system offers sound and complete deduction with many connectives, which can be chosen to fit the

real world situation. In the above case it can look like

$$\begin{aligned} \frac{1}{2} &= f(A \&_G A) \\ &= \& \dot{G} (f(A), f(A)) > \& \dot{L} (f(A), f(\neg A)) \\ &= f(A \&_G \neg A) = 0. \end{aligned}$$

6. Conclusions and historical comments

Our reference should first mention the development of many valued logic starting from Łukasiewicz [11] and Goedel [3] dealing with 1-tautologies, surveyed in Gottwald’s book [4]. Works of Zadeh [27,28] have also started investigations in uncertain reasoning based on many valued logic and fuzzy theories (i.e. theories which no more require the truth value of axioms being 1 but they postulate some minimal truth value $0 < TV \leq 1$ required for models). Pavelka [15] introduced a proof procedure for fuzzy theories in the Łukasiewicz logic and proved its completeness. This was generalized by Novák [14] for predicate calculus and substantially simplified by Hájek [7].

An exhaustive survey on fuzzy logic programming before ’91 is in Chap. 4.3 of Dubois et al. [1], most of this are mainly heuristic algorithms which are sometimes based on a truth functional logical system and seldom based on a proof of completeness. In what follows, we refer to results on many valued (fuzzy) logic programming in truth functional logical system. We refer to a system according to which many valued connectives appear in rules. Here a $(\rightarrow, \&)$ -program means definite logic program with rules of the form of implication \rightarrow with body consisting of $\&$ -conjunction of atoms. Soundness and completeness results were proved for $(\rightarrow_L, \&_L, \&_G)$ -programs by Klawon and Kruse [8] and for $(\rightarrow_G, \&_G)$ -programs by Mukaidono and Kikuchi [12]. In [24] Vojtáš and Paulík proved completeness for arbitrary $(\rightarrow, \&)$ -programs with left continuous t-norms under some restrictions about evaluation of modus ponens. In [21] this is proved for propositional fuzzy logic without these restrictions.

In 1986 van Emden [2], not cited in [1], generalizing some results of Shapiro [17], proved under some restrictions soundness and completeness for $(\rightarrow_P, \&_G)$ -programs, though not in a truth functional logic. He introduced, moreover, a fuzzy fixpoint theory and invented a method which allows to cut the search tree es-

timating the truth value for product implication. In the present paper we generalize also van Emden's method of cut. Note that the MYCIN-expert system [18] is based on $(\rightarrow_P, \&_G)$ -rules. van Emden's results were implemented by Li and Liu [9]. This led to other approaches to fuzzy logic programming (anotated, hybrid, probabilistic programs). There are also systems using possibilistic and signed logic. It is out of the scope of this paper to discuss relations of our system to these systems.

In this paper we presented a soundness and completeness proof for fuzzy logic programs without negation and with a wide variety of connectives in a truth functional fuzzy logic in a narrow sense. We mentioned applications in threshold computation, abduction, similarity based fuzzy unification, fuzzy databases, flexible querying and clausal resolution.

References

- [1] D. Dubois, J. Lang, H. Prade, Fuzzy sets in approximate reasoning, Part 2: Logical approaches, in: I.B. Turksen, D. Dubois, H. Prade, R.R. Yager (Eds.), Foundations of Fuzzy Reasoning. Special Memorial Volume; 25 years of fuzzy sets: A tribute to Professor Lotfi Zadeh, First issue; Fuzzy Sets and Systems 40 (1991) 203–244.
- [2] M.H. van Emden, Quantitative deduction and its fixpoint theory, J. Logic Programming 1 (1986) 37–53.
- [3] K. Goedel, Zum intuitionistischen Aussagenkalkuel. Anzeiger Akademie der Wissenschaften Wien, Math.- Naturwissensch. Klasse 69 (1932) 65–66; also in Ergebnisse eines mathematischen Kolloquiums 4 (1933) 40.
- [4] S. Gottwald, Mehrwertige Logik, Akademie Verlag, Berlin, 1988.
- [5] S. Gottwald, Fuzzy Sets and Fuzzy Logic, Vieweg, Braunschweig, 1993.
- [6] M.M. Gupta, J. Qi, Theory of T-norms and fuzzy inference methods, in: M.M. Gupta (Ed.), Fuzzy Logic and Uncertainty Modeling, Special Memorial Volume; 25 years of fuzzy sets: A tribute to Proffesor Lotfi Zadeh, Third issue, Fuzzy Sets and Systems 40 (1991) 431–450.
- [7] P. Hájek, Metamathematics of Fuzzy Logic, Kluwer, Dordrecht, 1998.
- [8] K. Klawonn, K. Kruse, A. Łukasiewicz, logic based Prolog, Mathware Soft Comput. 1 (1994) 5–29.
- [9] D.Y. Li, D.B. Liu, A Fuzzy Prolog Database System, Research Studies Press and Wiley, New York, 1990.
- [10] J.W. Lloyd, Foundation of Logic Programming, Springer, Berlin, 1987.
- [11] J. Łukasiewicz, Selected Works, North-Holland, Amsterdam, 1970.
- [12] M. Mukaidono, H. Kikuchi, Foundations of fuzzy logic programming, in: P.-Z. Wang, K.-F. Loe (Eds.), Between Mind and Computer, Advances in Fuzzy Systems—Applications and Theory, vol. 1, World Scientific Publ., Singapore, pp. 225–244.
- [13] E. Naito, J. Ozawa, I. Hayashi, N. Wakami, A proposal of a fuzzy connective with learning function, in: P. Bosc, J. Kaczprzyk (Eds.), Fuzziness Database Management Systems, Physica-Verlag, Wurzburg, 1995, pp. 345–364.
- [14] V. Novák, On the syntactico-semantic completeness of first-order fuzzy logic I, II. Kybernetika 26 (1990) 26–47, 134–152.
- [15] J. Pavelka, On fuzzy logic I, II, III, Zeitschr. Math. Logik und Grundl. Math. 25 (1979) 45–52, 119–134, 447–464.
- [16] W. Pedrycz, Fuzzy control and fuzzy systems, Report 82/14, Dept. Math., Delft Univ. of Technology.
- [17] E.Y. Shapiro, Programs with uncertainties, in: A. Bundy (Ed.), Proc. 8 IJCAI, W. Kaufmann, Los Altmos, CA, 1983, pp. 529–532.
- [18] E.H. Shortliffe, B.G. Buchanan, A model of inexact reasoning in medicine, Math. Biosci. 23 (1975) 351–379.
- [19] D. Smutná, P. Vojtáš, Fuzzy resolution with residuation of material implication, EUROFUSE-SIC'99, Budapest 1999, pp. 472–476.
- [20] D. Smutná, P. Vojtáš, New connectives for (full) fuzzy resolution, in: P. Sinčák et al. (Eds.), Proc. ISCI, Košice, Physica-Verlag, Heidelberg, 2000, pp. 146–151.
- [21] P. Vojtáš, Fuzzy reasoning with tunable t-operators, J. Adv. Comput. Intelligence 2 (1998) 121–127.
- [22] P. Vojtáš, Fuzzy logic abduction, Proc. Workshop 17 at ECAI'98, Brighton, 1998.
- [23] P. Vojtáš, Declarative and procedural model of fuzzy unification, Kybernetika 36 (2000) 707–720.
- [24] P. Vojtáš, L. Paulík, Soundness and completeness of non-classical extended SLD resolution, in: R. Dyckhoff et al. (Eds.), Proc. ELP'96 Leipzig, Lecture Notes in Computer Science, vol. 1050, Springer, Berlin, 1996, pp. 289–301.
- [25] P. Vojtáš, Z. Fabián, Aggregating similar witness for flexible query answering, in: H.L. Larsen et al., (Eds.), Proc. Flexible Query Answering Systems FQAS 2000, Warszawa, Physica-Verlag, Heidelberg, 2000, pp. 220–229.
- [26] J. Vinař, P. Vojtáš, A formal model for fuzzy knowledge based systems with similarities, Neural Network World 10 (2000) 891–905.
- [27] L.A. Zadeh, Fuzzy sets, Inform. and Control 8 (1965) 338–353.
- [28] L.A. Zadeh, Fuzzy logic and approximate reasoning (in memory of Grigore Moisil), Synthese 30 (1975) 407–428.