A Declarative Semantics for a Fuzzy Logic Language Managing Similarities and Truth Degrees*

Pascual Julián-Iranzo
Department of Technologies and Information Systems
University of Castilla-La Mancha
13071 Ciudad Real (Spain)
Pascual.Julian@uclm.es

Ginés Moreno
Department of Computing Systems
University of Castilla-La Mancha
02071 Albacete (Spain)
Gines.Moreno@uclm.es

Jaime Penabad
Department of Mathematics
University of Castilla-La Mancha
02071 Albacete (Spain)
Jaime.Penabad@uclm.es

Carlos Vázquez
Department of Computing Systems
University of Castilla-La Mancha
02071 Albacete (Spain)
Carlos.Vazquez@uclm.es

This work proposes a declarative semantics based on a fuzzy variant of the classical notion of least Herbrand model for the so-called FASILL language (acronym of "Fuzzy Aggregators and Similarity Into a Logic Language") which has being recently designed and implemented in our research group for coping with implicit/explicit truth degree annotations, a great variety of connectives and unification by similarity.

Keywords: Fuzzy Logic Programming, Similarity Relations, Herbrand Model

1 Introduction

The challenging research area of *Fuzzy Logic Programming* is devoted to introducing *fuzzy logic* concepts into *logic programming* in order to explicitly deals with uncertainty in a natural way. It has provided an extensive variety of PROLOG dialects along the last three decades. *Fuzzy logic languages* can be classified (among other criteria) regarding the emphasis they assign to fuzzifying the original unification/resolution mechanisms of PROLOG. Whereas some approaches are able to cope with similarity/proximity relations at unification time [1, 7, 24, 26], other ones extend their operational principles (maintaining syntactic unification) for managing a wide variety of fuzzy connectives and truth degrees on rules/goals beyond the simpler case of *true* or *false* [13, 15, 21].

The first line of integration, where the syntactic unification algorithm is extended with the ability of managing similarity/proximity relations, is of special relevance for this work. Similarity/proximity relations associate the elements of a set with a certain approximation degree and serve for weakening the notion of equality and, hence, to deal with vague information. With respect to this line, the related work can be summarized as follows:

In [26], an extension of the declarative paradigm of classic logic programming is proposed by considering similarity-based computations (the weakening of the equality notion is managed by means of fuzzy similarity relations for dealing with vague information) which allows to perform approximate inferences. Also, it describes the notion of fuzzy least Herbrand model and proves the equivalence with the

*Work supported by the EU (FEDER), and the Spanish MINECO Ministry (*Ministerio de Economía y Competitividad*) under grant TIN2013-45732-C4-2-P.

Submitted to:
PROLE 2015

fixpoint semantics of logic programs with similarity. Moreover, a graded notion of logical consequence can be considered and the operational semantics is designed by introducing a modified version of SLD resolution, by using a generalized notion of most general unifier that provides a numeric value which gives a measure of the exploited approximation.

In [26] (work that extends the precedent [25]) an operational semantics and a fixpoint semantics are defined and related, as well as a fuzzy extension of the least Herbrand model is given. Also in [2], that proposes the logic programming language Likelog (*LIKEness in LOGic*) which relies on similarity too, an operational semantics and a fix-point semantics are defined.

A more general notion called proximity relation was introduced in [5] by omitting the transitivity axiom. The Bousi∼Prolog language [11] is a fuzzy logic programming language with an operational semantics which is an adaptation of the SLD resolution, incorporating a fuzzy unification algorithm based on proximity relations.

A different generalization of similarity-based logic programming is the SQLP scheme (see *S*imilarity-based reasoning in qualified logic programming, [3]), designed as an innovative extension of the QLP scheme (Quantitative Logic Programs of [6]), in which the authors show that the similarity-based logic programing approach presented in [26] can be reduced to Qualified Logic Programing in the QLP(D) scheme introduced in [23], which supports logic programming with attenuated program clauses over a parametrically given domain D. The SQCLP scheme is a notable extension of [6, 13] which supports qualification values (elements of a domain of qualification), proximity relations and notions coming from CLP. In this framework, the authors present a declarative semantics for SQCLP that is based on observables, providing fixpoint and proof-theoretical characterizations of least program models.

Ending this section, it is important to say that our research group has been involved both on the development of similarity-based logic programming systems and those that extend the resolution principle, as reveals the design of the Bousi∼Prolog language[1] [11, 12, 24], where clauses cohabit with similarity/proximity equations, and the development of the FLOPER system,[2] which manages fuzzy programs composed by rules richer than clauses [16, 20]. Our unifying approach is somehow inspired by [4], but in our framework we admit a wider set of connectives inside the body of programs rules. In this paper, we propose the declarative semantics of de FASSIL language, whose operational semantics has been recently embedded into the FLOPER system.

Following the same scheme of [11, 10], this paper introduces the declarative semantics of the FASILL language, since it was proposed as a pending task in [8] where we initially presented the operational semantics of the new language.

The structure of this paper is as follows. Firstly, in Sections 2 and 3 we formally define and illustrate both the syntax and operational semantics of the FASILL language, initially presented in [8]. Next, Section 4 details its declarative semantics by introducing the concept of Herbrand model and least Herbrand model of a FASILL program. Finally, in Section 5 we present our conclusions and future research lines.

2 The FASILL language

FASILL is a first order language built upon a signature Σ, that contains the elements of a countably infinite set of variables $\mathscr{V}$, function symbols, and predicate symbols with an associated arity –usually expressed as pairs f/n or p/n where n represents its arity–, the implication symbol ($\leftarrow$), and a wide set of others connectives ς. The language combines the elements of Σ as terms, atoms, rules, and formulas.

[1] Two different programming environments for Bousi∼Prolog are available at `http://dectau.uclm.es/bousi/`.

[2] The tool is freely accessible from the Web site `http://dectau.uclm.es/floper/`.

A *constant c* is a function symbol with arity zero. A *term* is a variable, a constant or a function symbol f/n applied to n terms $t_1,\ldots,t_n$, and is denoted as $f(t_1,\ldots,t_n)$. We allow values of a lattice L as part of the signature Σ. Therefore, a well-formed formula can be either:

- r, if $r \in L$,
- $p(t_1,\ldots,t_n)$, if $t_1,\ldots,t_n$ are terms and p/n is an n-ary predicate. This formula is called *atom*. Particularly, atoms containing no variables are called *ground atoms*, and atoms built from nullary predicates are called *propositional variables*,
- $\varsigma(\mathscr{F}_1,\ldots,\mathscr{F}_n)$, if $\mathscr{F}_1,\ldots,\mathscr{F}_n$ are well-formed formulas and ς is an n-ary connective with truth function $\dot{\varsigma} : L^n \to L$.

Definition 2.1 (Complete lattice)**.** *A complete lattice is a partially ordered set $(L,\leq)$ such that every subset S of L has infimum and supremum elements. Then, it is a bounded lattice, i.e., it has bottom and top elements, denoted by $\bot$ and $\top$, respectively. L is said to be the carrier set of the lattice, and $\leq$ its ordering relation.*

The language is equipped with a set of *connectives* ς[3] interpreted on the lattice, including

- aggregators denoted by @, whose truth functions $\dot{@}$ fulfill the boundary condition: $\dot{@}(\top,\top) = \top$, $\dot{@}(\bot,\bot) = \bot$, and monotonicity: $(x_1,y_1) \leq (x_2,y_2) \Rightarrow \dot{@}(x_1,y_1) \leq \dot{@}(x_2,y_2)$.
- t-norms and t-conorms [22] (also named conjunctions and disjunctions, that we denote by & and $|$, respectively) whose truth functions fulfill the following properties:
  - · Commutative: $\dot{\&}(x,y) = \dot{\&}(y,x)$ $\quad$ $\dot{|}(x,y) = \dot{|}(y,x)$
  - · Associative: $\dot{\&}(x,\dot{\&}(y,z)) = \dot{\&}(\dot{\&}(x,y),z)$ $\quad$ $\dot{|}(x,\dot{|}(y,z)) = \dot{|}(\dot{|}(x,y),z)$
  - · Identity element: $\dot{\&}(x,\top) = x$ $\quad$ $\dot{|}(x,\bot) = x$
  - · Monotonicity in each argument:

$$z \leq t \Rightarrow \begin{cases} \dot{\&}(z,y) \leq \dot{\&}(t,y) & \dot{\&}(x,z) \leq \dot{\&}(x,t) \\ \dot{|}(z,y) \leq \dot{|}(t,y) & \dot{|}(x,z) \leq \dot{|}(x,t) \end{cases}$$

Example 1. *In this paper we use the lattice $([0,1],\leq)$, where $\leq$ is the usual ordering relation on real numbers, and three sets of connectives corresponding to the fuzzy logics of Gödel, Łukasiewicz and Product, defined in Figure 1, where labels* `L`*,* `G` *and* `P` *mean respectively* Łukasiewicz logic*,* Gödel logic*, and* product logic *(with different capabilities for modeling* pessimistic*,* optimistic*, and* realistic scenarios*).*

$\dot{\&}_{\mathtt{P}}(x,y) \triangleq x * y$	$\dot{\|}_{\mathtt{P}}(x,y) \triangleq x + y - xy$	*Product*
$\dot{\&}_{\mathtt{G}}(x,y) \triangleq \min(x,y)$	$\dot{\|}_{\mathtt{G}}(x,y) \triangleq max(x,y)$	*Gödel*
$\dot{\&}_{\mathtt{L}}(x,y) \triangleq \max(0,x+y-1)$	$\dot{\|}_{\mathtt{L}}(x,y) \triangleq \min(x+y,1)$	*Łukasiewicz*

Figure 1: Conjunctions and disjunctions in $[0,1]$ for *Product*, *Łukasiewicz*, and *Gödel* fuzzy logics

It is possible to include also other connectives. For instance, the arithmetical average, defined by connective $@_{aver}$ (with truth function $\dot{@}_{aver}(x,y) \triangleq \frac{x+y}{2}$), that is a stated, easy to understand connective that does not belong to a known logic. Connectives with arities different from 2 can also be used, like the $@_{very}$ aggregation, defined by $\dot{@}_{very}(x) \triangleq x^2$, that is a unary connective.

[3] Here, the connectives are binary operations but we usually generalize them with an arbitrary number of arguments, that is, with truth function $\dot{\varsigma} : L^n \to L$.

Definition 2.2 (Similarity relation)**.** *Given a domain $\mathscr{U}$ and a lattice L with a fixed t-norm $\wedge$, a similarity relation $\mathscr{R}$ is a fuzzy binary relation on $\mathscr{U}$, that is a fuzzy subset on $\mathscr{U} \times \mathscr{U}$ (namely, a mapping $\mathscr{R} : \mathscr{U} \times \mathscr{U} \to L$), such that fulfils the following properties:*[4]

- *Reflexive:* $\mathscr{R}(x,x) = \top, \forall x \in \mathscr{U}$,
- *Symmetric:* $\mathscr{R}(x,y) = \mathscr{R}(y,x), \forall x,y \in \mathscr{U}$,
- *Transitive:* $\mathscr{R}(x,z) \geq \mathscr{R}(x,y) \wedge \mathscr{R}(y,z), \forall x,y,z \in \mathscr{U}$.

Certainly, we are interested in fuzzy binary relations on a syntactic domain. We primarily define similarities on the symbols of a signature, Σ, of a first order language. This makes possible to treat as indistinguishable two syntactic symbols which are related by a similarity relation $\mathscr{R}$. Moreover, a similarity relation $\mathscr{R}$ on the alphabet of a first order language can be extended to terms by structural induction in the usual way [26]. That is, the extension, $\hat{\mathscr{R}}$, of a similarity relation $\mathscr{R}$ is defined as:

1. let x be a variable, $\hat{\mathscr{R}}(x,x) = \mathscr{R}(x,x) = 1$,
2. let f and g be two n-ary function symbols and let $t_1, \ldots, t_n$, $s_1, \ldots, s_n$ be terms,
   $\hat{\mathscr{R}}(f(t_1,\ldots,t_n), g(s_1,\ldots,s_n)) = \mathscr{R}(f,g) \wedge (\bigwedge_{i=1}^{n} \hat{\mathscr{R}}(t_i,s_i))$
3. otherwise, the approximation degree of two terms is zero.

Analogously for atomic formulas. In this work conditional formulas of the form $\mathscr{C} \equiv A \leftarrow \mathscr{B}$, where A is an atom, have a special relevance (see below). For this kind of formulas we use a different and more restrictive notion of similarity than the one defined in [26]. The idea is that a conditional formula $\mathscr{C}$ is similar to another conditional formula $\mathscr{C}'$ if their heads are similar but maintain the same body. Hence, given $\mathscr{C} : A \leftarrow \mathscr{B}$ and $\mathscr{C}' : A' \leftarrow \mathscr{B}'$, $\hat{\mathscr{R}}(\mathscr{C},\mathscr{C}') = \hat{\mathscr{R}}(A,A')$ if $\mathscr{B} = \mathscr{B}'$; otherwise $\hat{\mathscr{R}}(\mathscr{C},\mathscr{C}') = 0$. That is, a conditional formula $\mathscr{C}$ is similar to another conditional formula $\mathscr{C}'$ in the same degree that their heads provided that they have the same body.

Note that, in the sequel, we shall not make a notational distintion between the relation $\mathscr{R}$ and its extension $\hat{\mathscr{R}}$.

Example 2. *A similarity relation $\mathscr{R}$ on $\mathscr{U} = \{vanguardist, elegant, metro, taxi, bus\}$ is defined by the following matrix:*

$\mathscr{R}$	*vanguardist*	*elegant*	*metro*	*taxi*	*bus*
vanguardist	*1*	*0.6*	*0*	*0*	*0*
elegant	*0.6*	*1*	*0*	*0*	*0*
metro	*0*	*0*	*1*	*0.4*	*0.5*
taxi	*0*	*0*	*0.4*	*1*	*0.4*
bus	*0*	*0*	*0.5*	*0.4*	*1*

It is easy to check that $\mathscr{R}$ fulfills the reflexive, symmetric, and transitive properties. Particularly, using the Gödel *conjunction as the t-norm $\wedge$, we have that:* $\mathscr{R}(taxi, metro) \geq \mathscr{R}(metro, bus) \wedge \mathscr{R}(bus, taxi) = 0.5 \wedge 0.4$.

[4] For convenience, $\mathscr{R}(x,y)$, also denoted $x\mathscr{R}y$, refers to both the syntactic expression (that symbolizes that the elements $x,y \in \mathscr{U}$ are related by $\mathscr{R}$) and the membership degree $\mu_{\mathscr{R}}(x,y)$, i.e., the affinity degree of the pair $(x,y) \in \mathscr{U} \times \mathscr{U}$ with the verbal predicate (or fuzzy predicate) $\mathscr{R}$.

Furthermore, the extension $\hat{\mathscr{R}}$ of $\mathscr{R}$ determines that the terms $elegant(taxi)$ and $vanguardist(metro)$[5] *are similar, since: $\hat{\mathscr{R}}(elegant(taxi), vanguardist(metro)) = \mathscr{R}(elegant, vanguardist) \wedge \hat{\mathscr{R}}(taxi, metro) = 0.6 \wedge \mathscr{R}(taxi, metro) = 0.6 \wedge 0.4 = 0.4$.*

Definition 2.3 (Rule and goal)**.** *A rule has the form $A \leftarrow \mathscr{B}$, where A is an atomic formula called head and $\mathscr{B}$, called body, is a well-formed formula (ultimately built from atomic formulas $B_1, \ldots, B_n$, truth values of L, and connectives).* [6] *In particular, when the body of a rule is $r \in L$ (an element of lattice L), this rule is called fact and can be written as $A \leftarrow r$ (or simply A if $r = \top$). A goal is a body submitted as a query to the system.*

Definition 2.4 (Program)**.** *A* FASILL *program (or simply program) is a tuple $\langle \Pi, \mathscr{R}, L \rangle$ where Π is a set of rules, $\mathscr{R}$ is a similarity relation between the elements of Σ, and L is a complete lattice.*

Example 3. *The set of rules Π given below, the similarity relation $\mathscr{R}$ of Example 2, and lattice $L = ([0,1], \leq)$ of Example 1, form a program $\mathscr{P} = \langle \Pi, \mathscr{R}, L \rangle$.*

$$\Pi = \begin{cases} R_1: & vanguardist(hydropolis) & \leftarrow 0.9 \\ R_2: & elegant(ritz) & \leftarrow 0.8 \\ R_3: & close(hydropolis, taxi) & \leftarrow 0.7 \\ R_4: & good\_hotel(x) & \leftarrow @_{aver}(elegant(x), @_{very}(close(x, metro))) \end{cases}$$

3 Operational Semantics of FASILL

Rules in a FASILL program have the same role as clauses in PROLOG (or MALP [15, 10, 19]) programs, that is, stating that a certain predicate relates some terms (the *head*) if some conditions (the *body*) hold.

As a logic language, FASILL inherits the concepts of substitution, unifier, and most general unifier (*mgu*). Some of them are extended to cope with similarities. Concretely, following the line of Bousi∼Prolog [11], the most general unifier is replaced by the concept of *weak most general unifier* (w.m.g.u.) and a weak unification algorithm is introduced to compute it. Roughly speaking, the *weak unification algorithm* states that two *expressions* (i.e, terms or atomic formulas) $f(t_1, \ldots, t_n)$ and $g(s_1, \ldots, s_n)$ weakly unify if the root symbols f and g are close with a certain degree (i.e. $\mathscr{R}(f, g) = r > \bot$) and each of their arguments t_i and s_i weakly unify. Therefore, there is a weak unifier for two expressions even if the symbols at their roots are not syntactically equals ($f \not\equiv g$).

More technically, the weak unification algorithm we are using is a reformulation/extension of the one which appears in [26] for arbitrary complete lattices. We formalize it as a transition system supported by a similarity-based unification relation "$\Rightarrow$". The unification of the expressions $\mathscr{E}_1$ and $\mathscr{E}_2$ is obtained by a state transformation sequence starting from an initial state $\langle G \equiv \{\mathscr{E}_1 \approx \mathscr{E}_2\}, id, \alpha_0 \rangle$, where id is the identity substitution and $\alpha_0 = \top$ is the supreme of $(L, \leq)$: $\langle G, id, \alpha_0 \rangle \Rightarrow \langle G_1, \theta_1, \alpha_1 \rangle \Rightarrow \cdots \Rightarrow \langle G_n, \theta_n, \alpha_n \rangle$. When the final state $\langle G_n, \theta_n, \alpha_n \rangle$, with $G_n = \emptyset$, is reached (i.e., the equations in the initial state have been solved), the expressions $\mathscr{E}_1$ and $\mathscr{E}_2$ are unifiable by similarity with w.m.g.u. θ_n and *unification degree* α_n. Therefore, the final state $\langle \emptyset, \theta_n, \alpha_n \rangle$ signals out the unification success. On the other hand, when expressions $\mathscr{E}_1$ and $\mathscr{E}_2$ are not unifiable, the state transformation sequence ends with failure (i.e., $G_n = Fail$).

[5] Note that $elegant(taxi)$ and $vanguardist(metro)$ are 1-ary predicates, whereas that $taxi, metro$ are terms with arity 0.

[6] In order to subsume the syntactic conventions of MALP, in our programs we also admit *weighted rules* with shape "$A \leftarrow_i \mathscr{B}$ *with* v", which are internally treated as "$A \leftarrow (v \&_i \mathscr{B})$" (this transformation preserves the meaning of rules as proved in [17]).

The *similarity-based unification relation*, " $\Rightarrow$", is defined as the smallest relation derived by the following set of transition rules (where $\mathscr{V}ar(t)$ denotes the set of variables of a given term t)

$$\frac{\langle\{f(t_1,\ldots,t_n)\approx g(s_1,\ldots,s_n)\}\cup E,\theta,r_1\rangle \quad \text{if} \quad \mathscr{R}(f,g)=r_2>\bot}{\langle\{t_1\approx s_1,\ldots,t_n\approx s_n\}\cup E,\theta,r_1\wedge r_2\rangle} \stackrel{\text{RULE 1}}{\Longrightarrow}$$

$$\frac{\langle\{X\approx X\}\cup E,\theta,r_1\rangle}{\langle E,\theta,r_1\rangle} \stackrel{\text{RULE 2}}{\Longrightarrow} \qquad \frac{\langle\{X\approx t\}\cup E,\theta,r_1\rangle \quad \text{if} \quad X\notin\mathscr{V}ar(t)}{\langle(E)\{X/t\},\theta\{X/t\},r_1\rangle} \stackrel{\text{RULE 3}}{\Longrightarrow}$$

$$\frac{\langle\{t\approx X\}\cup E,\theta,r_1\rangle}{\langle\{X\approx t\}\cup E,\theta,r_1\rangle} \stackrel{\text{RULE 4}}{\Longrightarrow} \qquad \frac{\langle\{X\approx t\}\cup E,\theta,r_1\rangle \quad \text{if} \quad X\in\mathscr{V}ar(t)}{\langle Fail,\theta,r_1\rangle} \stackrel{\text{RULE 5}}{\Longrightarrow}$$

$$\frac{\langle\{f(t_1,\ldots,t_n)\approx g(s_1,\ldots,s_n)\}\cup E,\theta,r_1\rangle \quad \text{if} \quad \mathscr{R}(f,g)=\bot}{\langle Fail,\theta,r_1\rangle} \stackrel{\text{RULE 6}}{\Longrightarrow}$$

Rule 1 decomposes two expressions and annotates the relation between the function (or predicate) symbols at their root. The second rule eliminates spurious information and the fourth rule interchanges the position of the symbols to be handled by other rules. The third and fifth rules perform an occur check of variable X in a term t. In case of success, it generates a substitution $\{X/t\}$; otherwise the algorithm ends with failure. It can also end with failure if the relation between function (or predicate) symbols in $\mathscr{R}$ is $\bot$, as stated by Rule 6.

Usually, given two expressions $\mathscr{E}_1$ and $\mathscr{E}_2$, if there is a successful transition sequence, $\langle\{\mathscr{E}_1\approx\mathscr{E}_2\},id,\top\rangle\Rightarrow^\star\langle\emptyset,\theta,r\rangle$, then we write that $wmgu(\mathscr{E}_1,\mathscr{E}_2)=\langle\theta,r\rangle$, being θ the *weak most general unifier* of $\mathscr{E}_1$ and $\mathscr{E}_2$, and r is their *unification degree*.

Finally note that, in general, a w.m.g.u. of two expressions $\mathscr{E}_1$ and $\mathscr{E}_2$ is not unique [26]. Certainly, the weak unification algorithm only computes a representative of a w.m.g.u. class, in the sense that, if $\theta=\{x_1/t_1,\ldots,x_n/t_n\}$ is a w.m.g.u., with degree β, then, by definition, any substitution $\theta'=\{x_1/s_1,\ldots,x_n/s_n\}$, satisfying $\mathscr{R}(s_i,t_i)>\bot$, for any $1\leq i\leq n$, is also a w.m.g.u. with approximation degree $\beta'=\beta\wedge(\bigwedge_{i=1}^n\mathscr{R}(s_i,t_i))$, where "$\wedge$" is a selected t-norm. However, observe that the w.m.g.u. representative computed by the weak unification algorithm is one with an approximation degree equal or greater than any other w.m.g.u. As in the case of the classical syntactic unification algorithm, our algorithm always terminates returning a success or a failure.

Next, we illustrate the weak unification process in the following example.

Example 4. *Consider the lattice $L=([0,1],\leq)$ of Example 1 and the relation $\mathscr{R}$ of Example 2. Given terms $elegant(taxi)$ and $vanguardist(metro)$, the following weak unification process it is possible:*

$$\langle\{elegant(taxi)\approx vanguardist(metro)\},id,1\rangle \stackrel{Rule\,1}{\Longrightarrow}\langle\{taxi\approx metro\},id,0.6\rangle \stackrel{Rule\,1}{\Longrightarrow}$$
$$\langle\{\},id,0.6\wedge 0.4\rangle=\langle\{\},id,0.4\rangle$$

Also it is possible to unify the terms $elegant(taxi)$ and $vanguardist(x)$, since:

$$\langle\{elegant(taxi)\approx vanguardist(x)\},id,1\rangle \stackrel{Rule\,1}{\Longrightarrow}\langle\{taxi\approx x\},id,0.6\rangle \stackrel{Rule\,4}{\Longrightarrow}$$
$$\langle\{x\approx taxi\},id,0.6\rangle \stackrel{Rule\,3}{\Longrightarrow}\langle\{\},\{x/taxi\},0.6\rangle$$

and the substitution $\{x/taxi\}$ is their w.m.g.u. with unification degree 0.6.

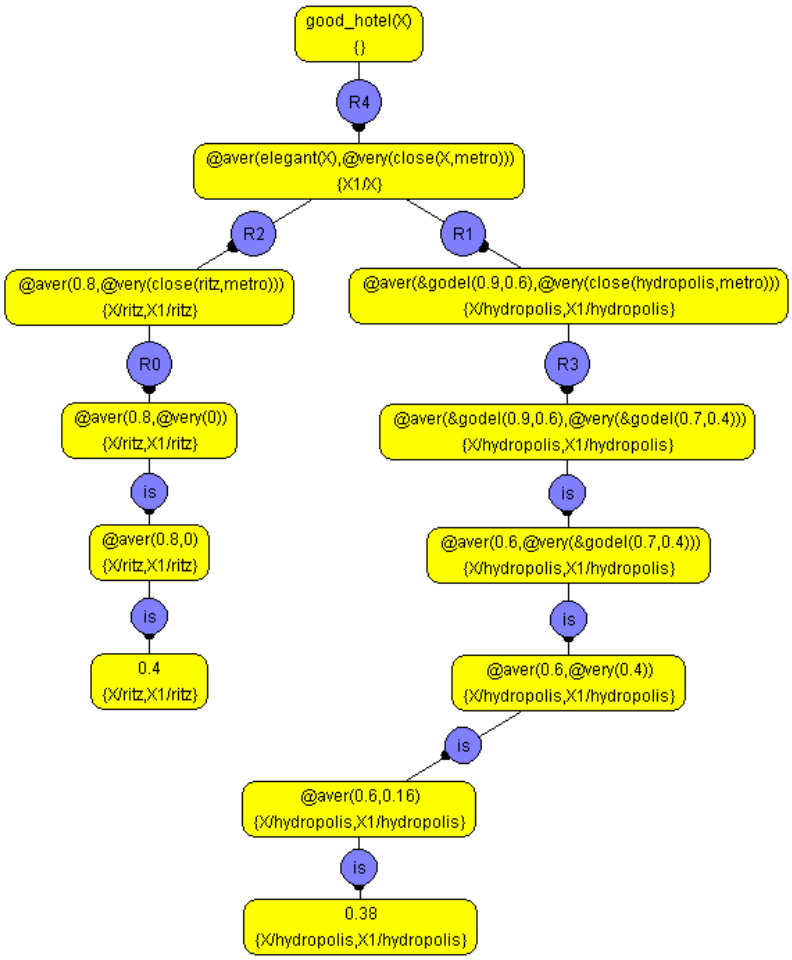


Figure 2: An execution tree as shown by the FLOPER system

In order to describe the procedural semantics of the FASILL language, in the following we denote by $\mathscr{C}[A]$ a formula where A is a sub-expression (usually an atom) which occurs in the –possibly empty– context $\mathscr{C}[]$ whereas $\mathscr{C}[A/A']$ means the replacement of A by A' in the context $\mathscr{C}[]$. Moreover, $\mathscr{V}ar(s)$ denotes the set of distinct variables occurring in the syntactic object s and $\theta[\mathscr{V}ar(s)]$ refers to the substitution obtained from θ by restricting its domain to $\mathscr{V}ar(s)$. In the next definition, we always consider that A is the selected atom in a goal $\mathscr{Q}$ and L is the complete lattice associated to Π.

Definition 3.1 (Computational Step)**.** *Let $\mathscr{Q}$ be a goal and let σ be a substitution. The pair $\langle \mathscr{Q}; \sigma \rangle$ is a* state*. Given a program $\langle \Pi, \mathscr{R}, L \rangle$ and a t-norm $\wedge$ in L, a* computation *is formalized as a state transition system, whose transition relation $\rightsquigarrow$ is the smallest relation satisfying these rules:*

1) Successful step *(denoted as $\overset{SS}{\rightsquigarrow}$):*

$$\frac{\langle \mathscr{Q}[A], \sigma \rangle \quad \textit{if} \quad A' \leftarrow \mathscr{B} \in \Pi \textit{ and } wmgu(A, A') = \langle \theta, r \rangle, r > \bot}{\langle \mathscr{Q}[A/\mathscr{B} \wedge r]\theta, \sigma\theta \rangle} \; \underset{\rightsquigarrow}{SS}$$

2) Failure step *(denoted as $\overset{FS}{\rightsquigarrow}$):*

$$\frac{\langle \mathscr{Q}[A], \sigma \rangle \quad \textit{if} \quad \nexists A' \leftarrow \mathscr{B} \in \Pi \textit{ such that } wmgu(A, A') = \langle \theta, r \rangle, r > \bot}{\langle \mathscr{Q}[A/\bot], \sigma \rangle} \ \underset{\rightsquigarrow}{\text{FS}}$$

3) Interpretive step *(denoted as* $\overset{IS}{\rightsquigarrow}$*):*

$$\frac{\langle \mathscr{Q}[@(r_1, \ldots, r_n)]; \sigma \rangle \quad \textit{if} \quad \dot{@}(r_1, \ldots, r_n) = r_{n+1}}{\langle \mathscr{Q}[@(r_1, \ldots, r_n)/r_{n+1}]; \sigma \rangle} \ \underset{\rightsquigarrow}{\text{IS}}$$

A *derivation* is a sequence of arbitrary lenght $\langle \mathscr{Q}; id \rangle \rightsquigarrow^* \langle \mathscr{Q}'; \sigma \rangle$. As usual, rules are renamed apart. When $\mathscr{Q}' = r \in L$, the state $\langle r; \sigma \rangle$ is called a *fuzzy computed answer* (f.c.a.) for that derivation.

Example 5. *Let* $\mathscr{P} = \langle \Pi, \mathscr{R}, L \rangle$ *be the program from Example 3. It is possible to perform this derivation with fuzzy computed answer* $\langle 0, 4, \{x/ritz\} \rangle$ *for* $\mathscr{P}$ *and goal* $\mathscr{Q} = good\_hotel(x)$*:*

$$\begin{array}{lll}
D: & \langle good_hotel(x), id \rangle & \overset{SS}{\rightsquigarrow}{}^{R4} \\
& \langle @_{aver}(elegant(x), @_{very}(close(x, metro))), \{x_1/x\} \rangle & \overset{SS}{\rightsquigarrow}{}^{R2} \\
& \langle @_{aver}(0.8, @_{very}(close(ritz, metro))), \{x_1/ritz, x/ritz\} \rangle & \overset{FS}{\rightsquigarrow} \\
& \langle @_{aver}(0.8, @_{very}(0)), \{x_1/ritz, x/ritz\} \rangle & \overset{IS}{\rightsquigarrow} \\
& \langle @_{aver}(0.8, 0), \{x_1/ritz, x/ritz\} \rangle & \overset{IS}{\rightsquigarrow} \\
& \langle 0.4, \{x_1/ritz, x/ritz\} \rangle &
\end{array}$$

This derivation corresponds to the leftmost branch in the tree of Figure 3 where we can also observe a second f.c.a. (that is, $\langle 0.38, \{x/hydropolis\} \rangle$*) for the same goal in the rightmost branch of such tree.*

4 Declarative semantics of FASILL

In logic programming, the declarative semantics for a program is traditionally formulated on the basis of the least Herbrand model (conceived as the infimum of a set of interpretations). In this section, we formally introduce the semantic notions of Herbrand interpretation, Herbrand model and least Herbrand model for a FASILL program $\mathscr{P}$, in order to characterize the declarative semantics for this kind of fuzzy programs. The process follows the guidelines of [14] and also generalizes the model-theoretic semantics defined in [10] for multi-adjoint logic programs and in [18] for X-MALP programs.[7] That is, if L is a multi-adjoint lattice and $\mathscr{P}$ is a multi-adjoint program (see [15, 9] for a description of these concepts), or L is a complete lattice and $\mathscr{P}$ is a X-MALP program, our Herbrand model $\mathscr{I}$ coincides with the one corresponding to this framework.

In what follows, we will consider that $\mathscr{B}_\mathscr{P}$ is the Herbrand base of the FASILL program $\mathscr{P}$, that is, the set of all ground atoms which can be formed by using the symbols in Π and in the similarity relation $\mathscr{R}$ of $\mathscr{P}$.

Definition 4.1 (Herbrand Interpretation)**.** *Let* $\mathscr{P} = \langle \Pi, \mathscr{R}, L \rangle$ *be a* FASILL *program. A Herbrand interpretation is a mapping* $\mathscr{I} : \mathscr{B}_\mathscr{P} \to L$*, where* $\mathscr{B}_\mathscr{P}$ *is the Herbrand base of* $\mathscr{P}$*.*

[7] Note that, X-MALP programs do not rely on *adjoint pairs*.

A Herbrand interpretation $\mathscr{I}$ can be extended in a natural way to the set of ground formulae of the language by simply making use of the following definition:

$$\mathscr{I}(\varsigma(F_1,\ldots,F_n)) = \dot{\varsigma}(\mathscr{I}(F_1),\ldots,\mathscr{I}(F_n))$$

where ς is an arbitrary connective. Note that, by abuse of language we use the same symbol for the Herbrand interpretation and its extension. In order to interpret a non ground (closed and universally quantified) formula A, it suffices to take

$$\mathscr{I}(A) = inf\{\mathscr{I}(A\theta) : A\theta \text{ is a ground instance of } A\}$$

Let $\mathscr{H}$ be the set of Herbrand interpretations whose order is induced from the order of L.

$$\mathscr{I}_1 \leq \mathscr{I}_2 \iff \mathscr{I}_1(F) \leq \mathscr{I}_2(F), \forall F \in \mathscr{B}_{\mathscr{P}}$$

It is trivial to check that $(\mathscr{H}, \leq)$ inherits the structure of complete lattice from $(L, \leq)$.

Definition 4.2 (Herbrand Model)**.** *Let $\mathscr{P} = \langle \Pi, \mathscr{R}, L\rangle$ be a* FASILL *program. A Herbrand interpretation $\mathscr{I}$ satisfies or is a* Herbrand model[8] *of a rule $A \leftarrow \mathscr{B} \in \mathscr{P}$ if, and only if, it verifies*

i) $\mathscr{I}(A) \geq \mathscr{I}(\mathscr{B})$,

ii) *if A' is an atom such that $\mathscr{R}(A,A') = r$, then $\mathscr{I}(A'\theta) \geq r \wedge \mathscr{I}(\mathscr{B}\theta)$, for all ground instances $A\theta, A'\theta$ of A and A'.*

Moreover, a Herbrand interpretation $\mathscr{I}$ is a Herbrand model *of $\mathscr{P}$ iff $\mathscr{I}$ is a Herbrand model of all rules in $\mathscr{P}$ (that is, all rules in $\mathscr{P}$ are satisfied by $\mathscr{I}$).*

Note that, in this way, given two atoms A and A' (where A is the head of a rule in $\mathscr{P}$) such that $\mathscr{R}(A,A') = r$ is very close to the supremum of L, they will be interpreted by any model $\mathscr{I}$ of $\mathscr{P}$ with very close values. In particular, if $\mathscr{R}(A,A') = \top$, the interpretation of A and A' (for any model $\mathscr{I}$) must satisfy: $\mathscr{I}(A) \geq \mathscr{I}(\mathscr{B})$, by i) and $\mathscr{I}(A'\theta) \geq \top \wedge \mathscr{I}(\mathscr{B}\theta) = \mathscr{I}(\mathscr{B}\theta)$, by ii). This is the expected result when A and A' are syntactically equal, because then $\mathscr{R}(A,A') = \top$.

On the other hand, if $\mathscr{R}(A,A') = \bot$ (i.e., there is no similarity between A and A'), the models of $\mathscr{P}$ are not constrained to provide similar values for A and A', since $\mathscr{I}(A) \geq \mathscr{I}(\mathscr{B})$ and $\mathscr{I}(A'\theta) \geq \bot \wedge \mathscr{I}(\mathscr{B}\theta) = \bot$. The definition of model, in this case, only requires that $\mathscr{I}(A')$ be greater or equal to $\bot$, that is, it can be any value. This is, once again, the expected result, since we defined A and A' as non-similar at all.

Definition 4.3 (Least Herbrand Model)**.** *Let $\mathscr{P} = \langle \Pi, \mathscr{R}, L\rangle$ be a* FASILL *program. The interpretation $\mathscr{I}_{\mathscr{P}} = inf\{\mathscr{I}_j : \mathscr{I}_j$ is Herbrand model of $\mathscr{P}\}$ is called the least fuzzy Herbrand[9] model of $\mathscr{P}$.*

The following result justifies that the previous interpretation $\mathscr{I}_{\mathscr{P}}$ can be really understood as the least fuzzy Herbrand model.

Theorem 4.4. *Let $\mathscr{P} = \langle \Pi, \mathscr{R}, L\rangle$ be a* FASILL *program. The Herbrand interpretation $\mathscr{I}_{\mathscr{P}} = inf\{\mathscr{I}_j :$ $\mathscr{I}_j$ is a Herbrand model of $\mathscr{P}\}$ is the least Herbrand model of $\mathscr{P}$.*

[8] Sometimes we will abbreviate writing "fuzzy model" or simply "model".

[9] Sometimes we will abbreviate writing "least fuzzy model" or simply "least model".

Proof. Let $\mathscr{M}$ be the set of Herbrand models of $\mathscr{P}$, that is, $\mathscr{M} = \{\mathscr{I}_j : \mathscr{I}_j \text{ is a Herbrand model of } \mathscr{P}\}$. $\mathscr{M}$ is not empty, being as the Herbrand interpretation $\mathscr{I} = sup(\mathscr{H})$, defined on each $A \in \mathscr{B}_\mathscr{P}$ by $\mathscr{I}(A) = sup(L)$, is a Herbrand model of $\mathscr{P}$.

Then, if we denote $\mathscr{I}_\mathscr{P} = \inf(\mathscr{M})$, $\mathscr{I}_\mathscr{P}$ is a Herbrand interpretation: since $(\mathscr{H}, \leq)$ is a complete lattice, there exists the infimum of the subset $\mathscr{M} \subset \mathscr{H}$ and it is a member of $\mathscr{H}$. We will prove that $\mathscr{I}_\mathscr{P}$ is a Herbrand model of $\mathscr{P}$, that is, it satisfies all rules of $\mathscr{P}$. Consider a rule $R = H \leftarrow \mathscr{B} \in \Pi$ and $\mathscr{I}_j \in \mathscr{M}$ ($\mathscr{I}_j$ is a model of $\mathscr{P}$). Since $\mathscr{I}_\mathscr{P}$ is the infimum of $\mathscr{M}$, $\mathscr{I}_\mathscr{P} \leq \mathscr{I}_j$, for all model $\mathscr{I}_j$ of $\mathscr{P}$. Therefore, $\mathscr{I}_\mathscr{P}(A) \leq \mathscr{I}_j(A)$ for each atom $A \in \mathscr{B}_\mathscr{P}$. Moreover, given that $\mathscr{I}_j$ is a Herbrand model of $\mathscr{P}$, $\mathscr{I}_j$ satisfies rule R, that is,

i) $\mathscr{I}_j(H) \geq \mathscr{I}_j(\mathscr{B})$,

ii) if H' is an atom such that $\mathscr{R}(H, H') = r$, then $\mathscr{I}_j(H'\theta) \geq r \wedge \mathscr{I}_j(\mathscr{B}\theta)$, for all ground instances $H\theta, H'\theta$ of H and H'.

By making use, again, of the definition of infimum, we have that:

$$\begin{array}{rcll} \mathscr{I}_\mathscr{P}(\mathscr{B}) & = & \inf\{\mathscr{I}_j(\mathscr{B}) : \mathscr{I}_j \text{ is Herbrand model of } \mathscr{P}\} & \\ & \leq & \inf\{\mathscr{I}_j(H) : \mathscr{I}_j \text{ is Herbrand model of } \mathscr{P}\} & = \mathscr{I}_\mathscr{P}(H) \end{array}$$

Consider now an atom H' verifying *ii*). Then, using the monotonicity of $\wedge$,

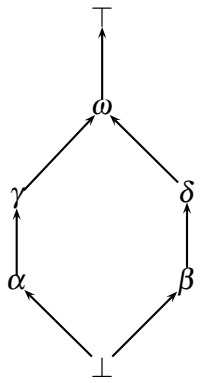
$$\begin{array}{rcll} \mathscr{I}_\mathscr{P}(H'\theta) & = & inf\{\mathscr{I}_j(H'\theta) : \mathscr{I}_j \text{ is Herbrand model of } \mathscr{P}\} & \\ & \geq & inf\{r \wedge \mathscr{I}_j(\mathscr{B}\theta) : \mathscr{I}_j \text{ is Herbrand model of } \mathscr{P}\} & \\ & \geq & r \wedge inf\{\mathscr{I}_j(\mathscr{B}\theta) : \mathscr{I}_j \text{ is Herbrand model of } \mathscr{P}\} & = r \wedge \mathscr{I}_\mathscr{P}(\mathscr{B}\theta) \end{array}$$

So, the conditions $\mathscr{I}_\mathscr{P}(H) \geq \mathscr{I}_\mathscr{P}(\mathscr{B})$ and $\mathscr{I}_\mathscr{P}(H'\theta) \geq r \wedge \mathscr{I}_\mathscr{P}(\mathscr{B}\theta)$ are fulfilled, and $\mathscr{I}_\mathscr{P}$ satisfies rule R and (since it analogously satisfies all rules in $\mathscr{P}$) it is a Herbrand model of $\mathscr{P}$, as we expected. Finally, since $\mathscr{I}_\mathscr{P} = \inf(\mathscr{M})$, using again the definition of infimum, $\mathscr{I}_\mathscr{P} \leq \mathscr{I}_j, \forall j$, so $\mathscr{I}_\mathscr{P}$ is the least Herbrand model of $\mathscr{P}$, which concludes the proof. □

In the next example we illustrate how to calculate the least Herbrand model $\mathscr{I}_\mathscr{P}$ of a FASILL program and, in general, how to calculate a Herbrand model $\mathscr{I}$ of $\mathscr{P}$.

Example 6. *Let $\mathscr{P} = \langle \Pi, \mathscr{R}, L \rangle$ be a* FASILL *program, where Π and the lattice $(L, \leq)$ are given in the following diagram ($(L, \leq)$ is stated by its Hasse diagram) and the relation $\mathscr{R}$ is the one establishing that $\mathscr{R}(a, a') = \alpha$ and $\mathscr{R}(b, b') = \beta$.*

If we assume that the truth function $\dot{\vee}_\mathtt{G}$ for connective $\vee_\mathtt{G}$ is defined by $\dot{\vee}_\mathtt{G}(x, y) = \sup\{x, y\}$, the least Herbrand model $\mathscr{I}_\mathscr{P}$ is determined by $\mathscr{I}_\mathscr{P}(p(a)) = \gamma$, $\mathscr{I}_\mathscr{P}(p(b)) = \delta$, $\mathscr{I}_\mathscr{P}(p(a')) = \alpha$, $\mathscr{I}_\mathscr{P}(p(b')) = \beta$, $\mathscr{I}_\mathscr{P}(q(a)) = \gamma$, $\mathscr{I}_\mathscr{P}(q(a')) = \alpha$, $\mathscr{I}_\mathscr{P}(r(b)) = \delta$, $\mathscr{I}_\mathscr{P}(r(b')) = \beta$, as shown in the next table (where atoms interpreted as $\bot$ have been omitted).



$$\Pi = \begin{cases} R_1: & r(b) & \leftarrow & \delta \\ R_2: & q(a) & \leftarrow & \gamma \\ R_3: & p(x) & \leftarrow & q(x) \vee_\mathtt{G} r(x) \end{cases}$$

	$p(a)$	$p(a')$	$p(b)$	$p(b')$	$q(a)$	$q(a')$	$r(b)$	$r(b')$
$\mathscr{I}_\mathscr{P}$	γ	α	δ	β	γ	α	δ	β

Indeed, by Definition 4.2,

$$\cdot\ \mathscr{I} \textit{ is Herbrand model of } R_1 \textit{ iff } \begin{cases} i)\mathscr{I}(r(b)) \geq \delta \\ ii)\mathscr{I}(r(b')) \geq \mathscr{R}(b,b') \wedge \mathscr{I}(r(b)) \geq \beta \wedge \delta = \beta \end{cases}$$

$$\cdot\ \mathscr{I} \textit{ is Herbrand model of } R_2 \textit{ iff } \begin{cases} i)\mathscr{I}(q(a)) \geq \gamma \\ ii)\mathscr{I}(q(a')) \geq \mathscr{R}(a,a') \wedge \mathscr{I}(q(a)) \geq \alpha \wedge \gamma = \alpha \end{cases}$$

$$\cdot\ \mathscr{I} \textit{ is Herbrand model of } R_3 \textit{ iff } \begin{cases} \mathscr{I}(p(a)) \geq \mathscr{I}(q(a) \vee r(a)) = \mathscr{I}(q(a)) \dot{\vee} \mathscr{I}(r(a)) \\ \mathscr{I}(p(b)) \geq \mathscr{I}(q(b) \vee r(b)) = \mathscr{I}(q(b)) \dot{\vee} \mathscr{I}(r(b)) \\ \mathscr{I}(p(a')) \geq \mathscr{R}(a,a') \wedge \mathscr{I}(q(a)) \dot{\vee} \mathscr{I}(r(a)) \\ \mathscr{I}(p(b')) \geq \mathscr{R}(b,b') \wedge \mathscr{I}(q(b)) \dot{\vee} \mathscr{I}(r(b)) \end{cases} \textit{, that is, } \mathscr{I}$$

$$\textit{fulfills} \begin{cases} \mathscr{I}(p(a)) \geq \gamma \vee \bot = \gamma \\ \mathscr{I}(p(b)) \geq \bot \vee \delta = \delta \\ \mathscr{I}(p(a')) \geq \alpha \wedge \gamma = \alpha \\ \mathscr{I}(p(b')) \geq \beta \wedge \delta = \beta \end{cases}$$

Note that this process allows us to calculate the least Herbrand model $\mathscr{I}_{\mathscr{P}}$ and also suggests how to obtain all Herbrand models $\mathscr{I}$ of $\mathscr{P}$.

	$\mathscr{I}_{\mathscr{P}}$
vanguardist(hydropolis)	0.9
vanguardist(ritz)	0.6
elegant(hydropolis)	0.6
elegant(ritz)	0.8
close(hydropolis, taxi)	0.7
close(hydropolis, metro)	0.4
close(hydropolis, bus)	0.5
good_hotel(hydropolis)	0.38
good_hotel(ritz)	0.4

Following the same methodology explained so far, the interested reader can easily check that the least Herbrand model for the program illustrated in Sections 3 and 4 is the one given in the adjoint table (where the interpretations for all atoms not included on it are assumed to be 0).

5 Conclusions and Future Work

FASILL (acronym of "Fuzzy Aggregators and Similarity Into a Logic Language") is a fuzzy logic programming language with implicit/explicit truth degree annotations, a great variety of connectives and unification by similarity. In [8] we have recently provided the syntax, operational semantics, and implementation issues[10] of this language which in essence integrates and extends features coming from MALP (*Multi-Adjoint Logic Programming*, a fuzzy logic language with explicitly annotated rules) and

[10]The last version of the FLOPER system which copes with similarity relations can be freely downloaded from `http://dectau.uclm.es/floper/?q=sim` and it can be tested on-line through `http://dectau.uclm.es/floper/?q=sim/test`.

Bousi∼Prolog (which uses a weak unification algorithm and is well suited for flexible query answering). Hence, it properly manages similarity and truth degrees in a single framework combining the expressive benefits of both languages. In this work we have focused on the formulation of a least model declarative semantics for FASILL, being this action a mandatory task in the development of the design of this framework. Obviously, a pending task for the immediate future consists in establishing the connections between our new fuzzy version of the least Herbrand model and the operational semantics of FASILL programs, in order to prove the correctness of the whole framework.

References

[1] F. Arcelli (2002): *Likelog for flexible query answering*. *Soft Computing* 7(2), pp. 107–114.

[2] F. Arcelli & F. Formato (1999): *Likelog: A Logic Programming Language for Flexible Data Retrieval*. In: *Proc. of the ACM Symposium on Applied Computing, SAC'99, San Antonio, Texas*, ACM, Artificial Intelligence and Computational Logic, pp. 260–267. Electronic Edition in http://doi.acm.org/10.1145/298151.298348.

[3] R. Caballero, M. Rodríguez-Artalejo & C. A. Romero-Díaz (2008): *Similarity-based reasoning in qualified logic programming*. In: *Proc. of the 10th Int. ACM SIGPLAN Conference on Principles and Practice of Declarative Programming*, PPDP'08, ACM, New York, USA, pp. 185–194.

[4] R. Caballero, M. Rodríguez-Artalejo & C. A. Romero-Díaz (2014): *A Transformation-based implementation for CLP with qualification and proximity*. *Theory and Practice of Logic Programming* 14(1), pp. 1–63.

[5] D. Dubois & H. Prade (1980): *Fuzzy Sets and Systems: Theory and Applications*. Academic Press, New York.

[6] M. H. van Emden (1986): *Quantitative Deduction and its Fixpoint Theory*. *Journal of Logic Programming* 3(1), pp. 37–53.

[7] F. Formato, G. Gerla & M. I. Sessa (2000): *Similarity-based Unification. Fundamenta Informaticae* 41(4), pp. 393–414.

[8] P. Julián Iranzo, G. Moreno, J. Penabad & C. Vázquez (2015): *A Fuzzy Logic Programming Environment for Managing Similarity and Truth Degrees*. In S. Escobar, editor: *Proc. of XIV Jornadas sobre Programación y Lenguajes, PROLE'14, Cádiz, Spain,, EPTCS* 173, pp. 71–86, doi:10.4204/EPTCS.173.6. Available at http://dx.doi.org/10.4204/EPTCS.173.6.

[9] P. Julián, G. Moreno & J. Penabad (2005): *On Fuzzy Unfolding. A Multi-adjoint Approach*. *Fuzzy Sets and Systems* 154, pp. 16–33.

[10] P. Julián, G. Moreno & J. Penabad (2009): *On the Declarative Semantics of Multi-Adjoint Logic Programs*. In: *Proc. of 10th Int. Work-Conference on Artificial Neural Networks (Part I), IWANN'09*, Lectures Notes in Computer Science, 5517, Springer Verlag, pp. 253–260.

[11] P. Julián-Iranzo & C. Rubio-Manzano (2009): *A declarative semantics for Bousi∼Prolog*. In: *Proc. of 11th Int. ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP'09, Coimbra, Portugal*, ACM, pp. 149–160.

[12] P. Julián-Iranzo & C. Rubio-Manzano (2010): *An efficient fuzzy unification method and its implementation into the Bousi∼Prolog system*. In: *Proc. of the 2010 IEEE Int. Conference on Fuzzy Systems*, pp. 1–8.

[13] M. Kifer & V.S. Subrahmanian (1992): *Theory of generalized annotated logic programming and its applications. Journal of Logic Programming* 12, pp. 335–367.

[14] J.W. Lloyd (1987): *Foundations of Logic Programming*. Springer-Verlag, Berlin.

[15] J. Medina, M. Ojeda-Aciego & P. Vojtáš (2004): *Similarity-based Unification: a multi-adjoint approach*. *Fuzzy Sets and Systems* 146, pp. 43–62.

[16] P. J. Morcillo, G. Moreno, J. Penabad & C. Vázquez (2010): *A Practical Management of Fuzzy Truth Degrees using FLOPER*. In: *Proc. of 4nd Int. Symposium on Rule Interchange and Applications, RuleML'10*, Lectures Notes in Computer Science, 6403, Springer Verlag, pp. 20–34.

[17] G. Moreno, J. Penabad & C. Vázquez (2013): *Relaxing the Role of Adjoint Pairs in Multi-adjoint Logic Programming*. In I. Hamilton & J. Vigo-Aguiar, editors: *Proc. of 13th Int. Conference on Mathematical Methods in Science and Engineering, CMMSE'13 (Volume III), Cabo de Gata, Almería*, pp. 1156–1167.

[18] G. Moreno, J. Penabad & C. Vázquez (2014): *Beyond Multi-adjoint Logic Programming*. *International Journal of Computer Mathematics* 92(9), pp. 1956–1975, doi:10.1080/00207160.2014.975218.

[19] G. Moreno, J. Penabad & C. Vázquez (2014): *Fuzzy Sets for a Declarative Description of Multi-adjoint Logic Programming*. In: *Proc. of the 2014 Joint Rough Set Symposium, JRS'14*, Lecture Notes in Computer Science, 8536, Springer Verlag, pp. 71–82.

[20] G. Moreno & C. Vázquez (2014): *Fuzzy Logic Programming in Action with FLOPER*. *Journal of Software Engineering and Applications* 7, pp. 237–298. Available at http://dx.doi.org/10.4236/jsea.2014.74028.

[21] S. Muñoz-Hernández, V. P. Ceruelo & H. Strass (2011): *RFuzzy: Syntax, semantics and implementation details of a simple and expressive fuzzy tool over Prolog*. *Information Sciences* 181(10), pp. 1951–1970.

[22] H. T. Nguyen & E. A. Walker (2006): *A First Course in Fuzzy Logic*. Chatman & Hall, Boca Ratón, Florida.

[23] M. Rodríguez-Artalejo & C. Romero-Díaz (2008): *Quantitative logic programming revisited*. In J. Garrigue & M. Hermenegildo, editors: *Proc. of 9th Functional and Logic Programming Symposium, FLOPS'08*, Lecture Notes in Computer Science 4989, Springer, pp. 272–288.

[24] C. Rubio-Manzano & P. Julián-Iranzo (2014): *A Fuzzy linguistic prolog and its applications*. *Journal of Intelligent and Fuzzy Systems* 26(3), pp. 1503–1516. Available at http://dx.doi.org/10.3233/IFS-130834.

[25] M. I. Sessa (2001): *Translations and similarity-based logic programming*. *Soft Computing* 5(2), pp. 160–170. Http://dx.doi.org/10.1007/PL00009891.

[26] M. I. Sessa (2002): *Approximate reasoning by similarity-based SLD resolution*. *Theoretical Computer Science* 275(1-2), pp. 389–426. Available at http://dx.doi.org/10.1016/S0304-3975(01)00188-8.