Proceedings of the 12th International Conference
on Computational and Mathematical Methods
in Science and Engineering, CMMSE 2012
La Manga, Spain, July, 2–5, 2012.

On Fuzzy Correct Answers and Logical Consequences in Multi-Adjoint Logic Programming

G. Moreno, J. Penabad and C. Vázquez[1]

[1] *University of Castilla-La Mancha, Faculty of Computer Science Engineering*
02071, Albacete (Spain),

emails: {Gines.Moreno,Jaime.Penabad}@uclm.es, Carlos.Vazquez@alu.uclm.es

Abstract

In this paper we reuse our previous concept of *least fuzzy Herbrand model* of a multi-adjoint logic program in order to extend the classic notions of logical consequence and correct answer to this framework, thus adapting to this programming style in a very natural way, usual concepts coming from pure logic programming. The least model, traditionally accepted as the declarative semantics of the corresponding logic programs, allows us too to relate both notions, which supposes a generalization w.r.t. the classic context in this sense. Furthermore, by using its set-based formulation, it can be thought as the set of formulae from the Herbrand base that are logical consequences of the rules of a multi-adjoint logic program, in a similar way to pure logic programming.

Key words: Multi-adjoint Logic Programming, Fuzzy Herbrand Model.
MSC 2000: Multi-adjoint Lattice.

1 Introduction

In [3] we have obtained, for the first time, the semantic notion of least fuzzy model conceived as the infimum of a set of interpretations. This concept, which reproduces the classical conception of least model of pure logic programming, is equivalent to the fix-point semantics and, also, to the procedural semantics conceived as the set of fuzzy computed answers. In the following, we will use this concept of least fuzzy Herbrand model to characterize correct answers and the notion of logical consequence, as well as their relationship. Furthermore, we include an original proof of the soundness of the procedural semantics of multi-adjoint logic programming. Moreover and even when this result replicates the corresponding one in pure logic programming, our proof provides significant novelties w.r.t. the classic case.

 ISBN:xxxxxxxxx

In what follows, we present a short summary of the main features of our language (we refer the reader to [7, 5, 6] for a complete formulation). We work with a first order language, $\mathcal{L}$, containing variables, function symbols, predicate symbols, constants, quantifiers ($\forall$ and $\exists$), and several (arbitrary) connectives to increase language expressiveness. In our fuzzy setting, we use implication connectives ($\leftarrow_1, \leftarrow_2, \ldots, \leftarrow_m$) and also other connectives which are grouped under the name of *aggregators* or *aggregation operators*. They are used to combine/propagate truth values through the rules. The general definition of aggregation operators subsumes conjunctive operators (denoted by $\&_1, \&_2, \ldots, \&_k$), disjunctive operators ($\vee_1, \vee_2, \ldots, \vee_l$), and average and hybrid operators (usually denoted by $@_1, @_2, \ldots, @_n$). Although the connectives $\&_i$, $\vee_i$ and $@_i$ are binary operators, we usually generalize them as functions with an arbitrary number of arguments. By definition, the truth function for an n-ary aggregation operator $[\![@]\!] : L^n \to L$ is required to be monotone and fulfills $[\![@]\!](\top, \ldots, \top) = \top$, $[\![@]\!](\bot, \ldots, \bot) = \bot$. Additionally, our language $\mathcal{L}$ contains the elements of a multi-adjoint lattice, $(L, \preceq, \leftarrow_1, \&_1, \ldots, \leftarrow_n, \&_n)$, equipped with a collection of adjoint pairs $(\leftarrow_i, \&_i)$, where each $\&_i$ is a conjunctor intended to the evaluation of *modus ponens*. In general, the set of truth values L may be the carrier of any complete lattice.

A *rule* is a formula $H \leftarrow_i \mathcal{B}$, where H is an atomic formula (usually called the *head*) and $\mathcal{B}$ (which is called the *body*) is a formula built from atomic formulas $B_1, \ldots, B_n$ ($n \geq 0$), truth values of L and conjunctions, disjunctions and aggregations. Rules with an empty body are called *facts*. A *goal* is a body submitted as a query to the system. Variables in a rule are assumed to be governed by universal quantifiers. Roughly speaking, a multi-adjoint logic program is a set of pairs $\langle \mathcal{R}; v \rangle$, where $\mathcal{R}$ is a rule and v is a *truth degree* (a value of L) expressing the confidence which the user of the system has in the truth of the rule $\mathcal{R}$.

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

Definition 1.1 (Admissible Step)**.** *Let $\mathcal{Q}$ be a goal and let σ be a substitution. The pair $\langle \mathcal{Q}; \sigma \rangle$ is a* state*. Given a program $\mathcal{P}$, an* admissible computation *is formalized as a state transition system, whose transition relation $\rightarrow_{AS}$ is the smallest relation satisfying the following* admissible rules*:*

1) $\langle \mathcal{Q}[A]; \sigma \rangle \rightarrow_{AS} \langle (\mathcal{Q}[A/v\&_i\mathcal{B}])\theta; \sigma\theta \rangle$ *if* $\theta = mgu(\{H = A\})$, $\langle H \leftarrow_i \mathcal{B}; v \rangle$ *in* $\mathcal{P}$ *and* $\mathcal{B}$ *is not empty.*

2) $\langle \mathcal{Q}[A]; \sigma \rangle \rightarrow_{AS} \langle (\mathcal{Q}[A/v])\theta; \sigma\theta \rangle$ *if* $\theta = mgu(\{H = A\})$, *and* $\langle H \leftarrow_i; v \rangle$ *in* $\mathcal{P}$.

3) $\langle \mathcal{Q}[A]; \sigma \rangle \rightarrow_{AS} \langle (\mathcal{Q}[A/\bot]); \sigma \rangle$ *if there is no rule in* $\mathcal{P}$ *whose head unifies with* A *(this case copes with possible unsuccessful branches).*

Definition 1.2 (Admissible Derivation)**.** *Let* $\mathcal{P}$ *be a program with an associated multi-adjoint lattice* $(L, \leq)$ *and let* $\mathcal{Q}$ *be a goal. An* admissible derivation *is a sequence* $\langle \mathcal{Q}; id \rangle \rightarrow^*_{AS} \langle \mathcal{Q}'; \theta \rangle$.

When $\mathcal{Q}'$ *is a formula not containing atoms and* $r \in L$ *is the result of interpreting* $\mathcal{Q}'$ *in* $(L, \leq)$*, the pairs* $\langle \mathcal{Q}'; \sigma \rangle$ *and* $\langle r; \sigma \rangle$*, where* $\sigma = \theta[\mathcal{V}ar(\mathcal{Q})]$*, are called* admissible computed answer *(a.c.a.) and* fuzzy computed answer *(f.c.a.), respectively (see [2] for details).*

Moreover, in the MALP framework [7, 5, 6], each program has its own associated multi-adjoint lattice and each program rule is "weighted" with an element of L, whereas the components in its body are *linked* with connectives of the lattice.

Furthermore, and continuing with semantic aspects, we formally introduce the semantic notions of interpretation and model for a multi-adjoint logic program $\mathcal{P}$, in a similar way to [7].

Definition 1.3. *An interpretation*[1] *is a map* $\mathcal{I} : B_{\mathcal{P}} \longrightarrow L$*, where* $B_{\mathcal{P}}$ *is the Herbrand base of the multi-adjoint logic program* $\mathcal{P}$ *and* $(L, \leq)$ *is the multi-adjoint lattice associated to program* $\mathcal{P}$*.*

$\mathcal{I}$ is extended in a natural way to the set of ground formulae of the language. In order to interpret a non ground formula A (closed, and universally quantified in the case of the multi-adjoint language), it suffices to take $\mathcal{I}(A) = inf\{\mathcal{I}(A\xi) \mid A\xi \text{ is a ground instance of A}\}$. Let $\mathcal{H}$ be the set of interpretations whose order is induced from the order of L:

$$\mathcal{I}_j \leq \mathcal{I}_k \Longleftrightarrow \mathcal{I}_j(F) \leq \mathcal{I}_k(F), \forall F \in B_{\mathcal{P}}$$

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

Definition 1.4. *An interpretation* $\mathcal{I}$ *satisfies (or is* model *of) a rule* $\langle H \leftarrow_i \mathcal{B}; \alpha_i \rangle$ *if, and only if,* $v \leq \mathcal{I}(H \leftarrow_i \mathcal{B})$*. An interpretation* $\mathcal{I}$ *is model*[2] *of* $\mathcal{P}$ *if, and only if, all rules in* $\mathcal{P}$ *are satisfied by* $\mathcal{I}$*.*

We will also admit that $\mathcal{I}$ satisfies (or is *model* of) a rule $\mathcal{R}_i = \langle R_i; \alpha_i \rangle$ if, and only if, $\alpha_i \leq \mathcal{I}(R_i)$.

[1] We will also say fuzzy Herbrand interpretation.
[2] We will also say fuzzy Herbrand model.

2 Fuzzy Herbrand Model

In [3] we have defined, for the first time in the literature using model theory, a declarative semantics for multi-adjoint logic programming in terms of the least fuzzy Herbrand model. This construction reproduces, in our fuzzy context, the classic construction of least Herbrand model of pure logic programming [4, 1], which has been traditionally accepted as the declarative semantics of logic programs.

In the last years, other adaptations of this concept have been provided, using model theory too ([12, 10, 11]), in alternative fuzzy logic programming frameworks different from the multi-adjoint logic programming approach.

Furthermore, in [3] we have related our notion of least fuzzy model with the already existing procedural semantics and fix-point semantics, and we have given revealing examples in which our declarative semantics has still sense beyond the multi-adjoint case, while the previously mentioned ones remain undefined.

Definition 2.1. *Let $\mathcal{P}$ be a multi-adjoint logic program with associated lattice $(L, \leq)$. The interpretation $\mathcal{I}_{\mathcal{P}} = inf\{\mathcal{I}_j : \mathcal{I}_j$ is model of $\mathcal{P}\}$ is called least fuzzy Herbrand[3] model of $\mathcal{P}$.*

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

Theorem 2.2 ([3])**.** *Let $\mathcal{P}$ be a multi-adjoint logic program with associated lattice L. The map $\mathcal{I}_{\mathcal{P}} = inf\{\mathcal{I}_j : \mathcal{I}_j$ is a model of $\mathcal{P}\}$ is the least model of $\mathcal{P}$.*

In the proof of this result provided in [3], it is essential that the lattice associated to the program be a multi-adjoint lattice. In this reference it is possible to contrast the necessity of this hypothesis for Theorem 2.2.

Moreover, the concepts of interpretation and model can be expressed in a set-theoretic approach, by understanding an interpretation of $\mathcal{P}$, instead of a map, as the corresponding binary relation. Let us give the following result to justify this fact.

Proposición 2.3. *Let $\mathcal{P}$ be a multi-adjoint logic program with associated lattice $(L, \leq)$. Let $\mathcal{I}$ be an interpretation of $\mathcal{P}$, namely, a map $\mathcal{I} : B_{\mathcal{P}} \longrightarrow L$. Then $\mathcal{I}$ determines a unique binary relation $R_{\mathcal{I}} \subset B_{\mathcal{P}} \times L$.*

Proof. It suffices by considering that map $\mathcal{I}$ is determined by its set of images, and the relation $R_{\mathcal{I}}$ is a set of pairs of type $\langle A, \mathcal{I}(A)\rangle, A \in B_{\mathcal{P}}$. Consequently, the map $\mathcal{I}$ can be seen as a binary relation, namely, a set of ordered pairs whose first component is a ground formula from the Herbrand base, and whose second component is an element of lattice L. □

[3] Sometimes we will say only least fuzzy model or least model.

Thanks to the previous result, a Herbrand interpretation $\mathcal{I}$ can be given by its binary relation $R_{\mathcal{I}}$ that, when not necessary to emphasize in this set-based aspect, will be simply denoted by $\mathcal{I}$.

Under this point of view, since each model of $\mathcal{P}$ can be seen as a set of pairs (subset of $B_{\mathcal{P}} \times L$), that is, $\mathcal{I}_j = \{\langle A, \alpha\rangle : A \in B_{\mathcal{P}}, \alpha \in L\}$, the least fuzzy model can be characterized also by the following result (formulated as the definition of least Herbrand model in [4]).

Theorem 2.4. *The least fuzzy Herbrand model of $\mathcal{P}$ is the intersection of all models of $\mathcal{P}$, namely, $\mathcal{I}_{\mathcal{P}} = \cap \mathcal{I}_j$, where $\mathcal{I}_j$ is a model of $\mathcal{P}$, for all j.*

Furthermore, the next theorem guarantees the existence of fuzzy Herbrand models by expressing interpretations as sets, in the terms we have just specified.

Theorem 2.5. *Let $\mathcal{P}$ be a multi-adjoint logic program and suppose that $\mathcal{P}$ has a model. Then, $\mathcal{P}$ has a fuzzy Herbrand model.*

Proof. Given an interpretation $\mathcal{I}$, $\mathcal{I}' = \{\mathcal{A} = \langle A; \alpha\rangle \in \mathcal{B}_{\mathcal{P}} \times L : \alpha \leq \mathcal{I}(\mathcal{A})\}$ is a Herbrand interpretation. Moreover, it is easy to justify that if $\mathcal{I}$ is a model of $\mathcal{P}$, $\mathcal{I}'$ is also a Herbrand model of $\mathcal{P}$. □

The following theorem states that the declarative semantics of the least fuzzy model is equivalent to the fix-point semantics built in [7] for multi-adjoint logic programming.

Theorem 2.6 ([3])**.** *Given the multi-adjoint logic program $\mathcal{P}$, the least fuzzy Herbrand model $\mathcal{I}_{\mathcal{P}}$ of $\mathcal{P}$ is the least fix-point of the immediate consequences operator $T_{\mathcal{P}}$.*

This last result states that the declarative semantics of a multi-adjoint logic program $\mathcal{P}$ can be obtained by the iteration of $T_{\mathcal{P}}$ from the least interpretation. It should be noted that operator $T_{\mathcal{P}}$ may not be continuous, and when this is the case –unlike in pure logic programming– it may be necessary to apply an uncountable number of iterations to reach the pursued fix-point [5].

Although the multi-adjoint property is very relevant for lattices associated to programs in order to define its operational and fix-point semantics, surprisingly, it is not always required for characterizing its least fuzzy model. As stated in their formal definitions, if the set $(L, \leq)$ is not a multi-adjoint lattice, then operator $T_{\mathcal{P}}$ is not defined and the fix-point semantics for the considered program $\mathcal{P}$ does not exist. The main reason is that both semantics exploit the relationship between the applications of the rules and their adjoint conjunctions, since the first ones have to be explicitly "substituted" by the second ones in the new states that are obtained by the application of admissible steps $AS1$ or when iterating the $T_{\mathcal{P}}$ operator. There is also no procedural semantics, and thus, fuzzy computed answers do not exist. But we wish to remark again that, even in this case, it might have sense to define the least fuzzy model as we have proposed before, since the declarative semantics could still exists (see again [3]).

3 Correct Answers via Least Fuzzy Herbrand Model

After justifying in the previous section the equivalence between the declarative and the fixpoint semantics, and using the equivalence considered in [7] between the last one and the procedural semantics, we have as a consequence that the declarative semantics of the least fuzzy model is equivalent to the procedural one in the terms specified in [7]. More exactly, soundness is always guaranteed, but only an approximated completeness result is achieved after incorporating to multi-adjoint programs the notion of reductant [2, 8]. Anyway, in Theorem 3.5 we provide an original proof (which relies on our notion of least fuzzy model) for the soundness of the procedural semantics.

Moreover, we will see too that, in the multi-adjoint framework, it is possible to characterize correct answers through the least fuzzy model. It must be taken into account that, in pure logic programming, the least Herbrand model does not suffice to characterize with maximum generality the set of correct answers (see Theorem 6.6 of [4]); so, in this aspect, our results in the multi-adjoint logic programming approach improve the classic ones of logic programming[4]. This characterization will be very useful in practice for obtaining the set of correct answers of a program from the least fuzzy model, as illustrated in Example 3.3 in this section.

Definition 3.1 ([7])**.** *Let $\mathcal{P}$ be a multi-adjoint program, and G a goal. The pair $\langle \lambda; \theta \rangle$ is a correct answer for $\mathcal{P}$ and G if, and only if, $\lambda \leq \mathcal{I}_j(G\theta)$, for each model $\mathcal{I}_j$ of $\mathcal{P}$.*

The following theorem states the referred characterization for the correct answer $\langle \lambda; \theta \rangle$, and moreover it shows that $\mathcal{I}_{\mathcal{P}}(G\theta)$ is the best correct answer obtainable for goal G in the multi-adjoint program $\mathcal{P}$.

Theorem 3.2. *Let $\mathcal{P}$ be a multi-adjoint program and G a goal. The pair $\langle \lambda; \theta \rangle$ is a correct answer for $\mathcal{P}$ and G if, and only if, $\lambda \leq \mathcal{I}_{\mathcal{P}}(G\theta)$, where $\mathcal{I}_{\mathcal{P}}$ is the least fuzzy model of $\mathcal{P}$.*

Proof. It is enough to use the definitions of the least fuzzy Herbrand model and correct answer.

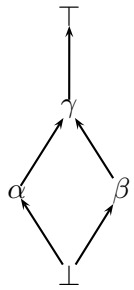
□

Note that if $G\theta$ is ground, $\mathcal{I}_{\mathcal{P}}(G\theta) = \alpha$ and the least fuzzy model is conceived as a set, so we could write $(G\theta, \alpha) \in \mathcal{I}_{\mathcal{P}}$ to adapt to the fuzzy context the habitual convention of logic programming. Also, notice that the previous definition of correct answer differs from the one given in classic logic programming (see [4] for this purpose) since, while SLD-resolution has a refutational behaviour, the procedural semantics of multi-adjoint programs is not. The following example adequately suggests how correct answers can be obtained from the least fuzzy Herbrand model.

[4]The non-refutational behaviour of the multi-adjoint language has a decisive influence in this sense.

Example 3.3. *Consider the following program $\mathcal{P}$, whose associated lattice $(L, \leq)$ is given by its depicted Hasse diagram:*

$$\begin{array}{lll} \mathcal{R}_1: & \langle p(a) & \leftarrow_{\mathtt{G}} ; \quad \alpha\rangle \\ \mathcal{R}_2: & \langle p(b) & \leftarrow_{\mathtt{G}} ; \quad \beta\rangle \\ \mathcal{R}_3: & \langle q(a) & \leftarrow_{\mathtt{G}} p(a); \quad \gamma\rangle \end{array}$$



Here, $(\&_{\mathtt{G}}, \leftarrow_{\mathtt{G}})$ is the pair of connectives following the Gödel*'s intuitionistic logic, whose truth functions are defined as:*

$$\dot{\&}_{\mathtt{G}}(x,y) = \inf\{x,y\} \qquad \textit{and} \qquad \dot{\leftarrow}_{\mathtt{G}}(y,x) = \begin{cases} \top, & \textit{if } x \leq y \\ y, & \textit{otherwise} \end{cases}$$

It is important to note that with these definitions, the pair $(\&_{\mathtt{G}}, \leftarrow_{\mathtt{G}})$ verify the condition for conforming an adjoint-pair regarding lattice $(L, \leq)$ of the figure above.

Now, it is easy to check that the least fuzzy Herbrand model $\mathcal{I}_{\mathcal{P}}$ is defined by: $\mathcal{I}_{\mathcal{P}}(p(a)) = \alpha$, $\mathcal{I}_{\mathcal{P}}(p(b)) = \beta$, $\mathcal{I}_{\mathcal{P}}(q(a)) = \alpha$, and $\mathcal{I}_{\mathcal{P}}(q(b)) = \bot$. Then:

i) For goal $p(a)$ we obtain the set of correct answers $\{\langle \lambda; id\rangle : \lambda \in L, \lambda \leq \alpha\}$.

ii) For goal $p(b)$ we obtain the set of correct answers $\{\langle \lambda; id\rangle : \lambda \in L, \lambda \leq \beta\}$.

iii) For goal $q(a)$ we obtain the set of correct answers $\{\langle \lambda; id\rangle : \lambda \in L, \lambda \leq \alpha\}$.

iv) For goal $p(x)$, the set of correct answers is $\{\langle \lambda; \theta\rangle : \lambda \in L, \lambda \leq \mathcal{I}_{\mathcal{P}}(p(x)\theta)\} = \{\langle \bot; \{X/a\}\rangle, \langle \alpha; \{X/a\}\rangle, \langle \bot; \{X/b\}\rangle, \langle \beta, \{X/b\}\rangle\}$. Remember that the result of applying the least model $\mathcal{I}_{\mathcal{P}}$ over a formula $p(x)$ gives $\mathcal{I}_{\mathcal{P}}(p(x)) = inf\{\mathcal{I}_{\mathcal{P}}(p(x)\sigma) : p(x)\sigma is\ ground\} =^5 inf\{\mathcal{I}_{\mathcal{P}}(p(a)), \mathcal{I}_{\mathcal{P}}(p(b))\} = inf\{\alpha, \beta\} = \bot$.

v) For goal $q(x)$, the set of correct answers is $\{\langle \lambda; \theta\rangle : \lambda \in L, \lambda \leq \mathcal{I}_{\mathcal{P}}(q(x)\theta)\} = \{\langle \alpha; \{X/a\}\rangle, \langle \bot; \{X/b\}\rangle\}$. Analogously to the previous case, we have now that the interpretation $\mathcal{I}_{\mathcal{P}}(q(x)) = inf\{\mathcal{I}_{\mathcal{P}}(q(a)), \mathcal{I}_{\mathcal{P}}(q(b))\} = inf\{\alpha, \bot\} = \bot$ (it is easy to check that the least model has to be defined this way from $q(a), q(b)$ formulae).

[5] Substitutions will only consider terms from the Herbrand universe of the program instead of variables.

In the following theorem we provide an original demonstration for the soundness of the procedural semantics of multi-adjoint programming. Therein we observe a certain analogy with the one included in [4] for the pure logic programming case, despite that the non refutational character of our language and its fuzzy feature determine very significative differences between both ones. Before tackling the mentioned result, we state the following lemma, that has an instrumental character.

Lema 3.4. *Let $(L, \leq)$ be a complete lattice. For all A, B of L, $A \subset B$, implies $inf(B) \leq inf(A)$.*

Proof. It suffices to consider the definition of the infimum and the complete character of lattice $(L, \leq)$. □

Observe that, thanks to the previous lemma, we have $\mathcal{I}(A\theta) \geq \mathcal{I}(A)$ [6], for all substitution θ and for all interpretation $\mathcal{I}$, whenever the set of ground instances of formula $A\theta$ is a subset of the set of ground instances of A.

Theorem 3.5 (Soundness)**.** *Let $\mathcal{P}$ be a multi-adjoint logic program, A an atomic goal and $\langle \lambda; \theta \rangle$ a fuzzy computed answer for A in $\mathcal{P}$. Then, $\langle \lambda; \theta \rangle$ is a correct answer for $\mathcal{P}$ and A.*

Proof. Let $D : [G_1, \ldots, G_n]$ be a derivation where $G_1 = \langle A; id \rangle \rightarrow^n_{AS/IS} \langle \lambda; \theta \rangle = G_n$. We prove the claim by induction on n, being n the length of D.

We see that, in first place, the result holds for $n = 1$. Indeed, if for goal A exists the derivation $\langle A; id \rangle \rightarrow_{AS} \langle \lambda; \theta \rangle$, then rule $\mathcal{R} : \langle H \leftarrow_i; \lambda \rangle \in \mathcal{P}$ and $A\theta = H\theta$. In that case, every model $\mathcal{I}$ of $\mathcal{P}$ satisfies rule $\mathcal{R}$ and, then, $\mathcal{I}(H \leftarrow_i) \geq \lambda$, namely, $\mathcal{I}(H) \geq \lambda$. Furthermore, from the equality $A\theta = H\theta$ it follows that $\mathcal{I}(A\theta) = \mathcal{I}(H\theta)$ and by Lemma 3.4, we obtain $\mathcal{I}(H\theta) \geq \mathcal{I}(H)$.

Consequently, we have $\mathcal{I}(A\theta) \geq \lambda$ and $\langle \lambda; \theta \rangle$ is a correct answer for $\mathcal{P}$ and A, as wanted.

Next suppose that the result is true for all derivation with length k and let us see that it is verified for an arbitrary derivation of length $k+1$, $D : [G_1, \ldots, G_{k+1}]$. Noting the first step of derivation D, we have $G_1 = \langle A; id \rangle \rightarrow_{AS} \langle v \&_i \mathcal{B}\sigma; \sigma \rangle = G_2$. That is, the admissible step has been executed using the program rule $\mathcal{R} : \langle H \leftarrow_i \mathcal{B}; v \rangle$, where atom A unifies with the head of rule $\mathcal{R}$ through the mgu σ. For each atom $B_i\sigma$ [7] , $i = 1, \ldots, n$, of $\mathcal{B}\sigma$ exists a derivation whose length is less or equal to k, which gives the computed answer $\langle b_i; \tau_i \rangle$.

More precisely, D include the following derivation steps:

[6] See, for instance, paragraphs $iv), v)$ of Example 3.3.

[7] Without lost of generality, we can suppose that in the considered derivation all admissible steps are executed before applying interpretive steps.

$$
\begin{array}{ll}
D : [\,\langle A; id\rangle & \rightarrow_{AS} \\
\quad \langle v \&_i \mathcal{B}\sigma; \sigma\rangle & = \\
\quad \langle v \&_i @(B_1\sigma, \ldots, B_n\sigma); \sigma\rangle & \rightarrow^{l_1}_{AS/IS} \\
\quad \langle v \&_i @(b_1, \ldots, B_n\sigma); \sigma \circ \tau_1\rangle & \rightarrow^{l_n}_{AS/IS} \\
\quad \langle v \&_i @(b_1, \ldots, b_n); \sigma \circ \tau_1 \circ \cdots \circ \tau_n\rangle^{8} & \rightarrow_{IS} \\
\quad \langle v \&_i b; \sigma \circ \tau\rangle & \rightarrow_{IS} \\
\quad \langle \lambda; \theta\rangle\,] &
\end{array}
$$

where $\tau = \tau_1 \circ \tau_2 \circ \cdots \circ \tau_n$, $\theta = \sigma \circ \tau$, $\lambda = v\dot{\&}_i b$, $l_1 + l_2 + \cdots + l_n = k - 2$ y $b = \dot{@}(b_1, \ldots, b_n)$, being @ the combination of all conjunctions, disjunctions and aggregators that links the elements $b_i \in L$ in order to obtain the correct answer $\langle b; \tau\rangle$ for program $\mathcal{P}$ and goal $\mathcal{B}\sigma$.

By the induction hypothesis, for each $B_i\sigma$, $\langle b_i; \tau_i\rangle$ is a correct answer and, then, $\mathcal{I}(B_i\sigma\tau_i) \geq b_i$, for all interpretation $\mathcal{I}$ that is model of $\mathcal{P}$. In that case, from $\mathcal{I}(B_i\sigma\tau_i) \geq b_i$ it follows that $\mathcal{I}(\mathcal{B}\sigma) \geq b$ since $\mathcal{I}(\mathcal{B}\sigma)$ is obtained from $\mathcal{I}(B_i\tau_i)$ as a result of applying the truth functions of conjunctions, disjunctions or aggregators, being all them monotone in each component.

Then, the equality $A\sigma = H\sigma$ entails $A\theta = H\theta$ and, therefore, $\mathcal{I}(A\theta) = \mathcal{I}(H\theta)$. Furthermore, by firstly using Lemma 3.4 having into account later that $(\leftarrow_i, \&_i)$ is an adjoint pair, it results $\mathcal{I}(H\theta) \geq \mathcal{I}(H) \geq v\dot{\&}_i\mathcal{I}(\mathcal{B}\sigma) \geq v\dot{\&}_i b = \lambda$.

Consequently, $\mathcal{I}(A\theta) \geq \lambda$ and $\langle \lambda; \theta\rangle$ is a correct answer for program $\mathcal{P}$ and atom A, as claimed. □

4 Logical Consequences via Least Fuzzy Herbrand Model

In what follows we formalize the concept of logical consequence in terms of the least fuzzy Herbrand model and we relate it with the notion of correct answer. Finally, we express the least model as the set of formulae from the Herbrand base that are a logical consequence of the set of rules of a multi-adjoint program, which allows us also to extend the classical formulation of least model to this programming style.

The following definition formalizes, in a natural way, the concept of logical consequence.

Definition 4.1. *Let $\mathcal{P}$ be a multi-adjoint logic program and $\mathcal{A} = \langle A; \alpha\rangle$ a multi-adjoint formula*[9]*. $\mathcal{A}$ is a logical consequence of $\mathcal{P}$ if, and only if, all model of $\mathcal{P}$ is a model of $\mathcal{A}$.*

[9] In fact, a ground or a closed and universally quantified formula in order to its interpretation has sense.

Given the multi-adjoint logic program $\mathcal{P} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$ where $\mathcal{R}_i : \langle R_i; \alpha_i \rangle, i = 1, \ldots, n$, the previous definition means that $\mathcal{A}$ is a logical consequence of $\mathcal{P}$ if, and only if, $\alpha_i \leq \mathcal{I}_j(R_i) \Rightarrow \alpha \leq \mathcal{I}_j(A), \forall i, j$. Now, the following theorem gives a characterization of this concept in terms of the least model.

Theorem 4.2. *Let $\mathcal{P}$ be a multi-adjoint program and $\mathcal{A}$ a multi-adjoint formula. $\mathcal{A} = \langle A; \alpha \rangle$ is a logical consequence of $\mathcal{P}$ if, and only if, $\mathcal{I}_\mathcal{P}$ is a model of $\mathcal{A}$.*

Proof. It is enough to consider the definition of the least fuzzy Herbrand model in order to obtain the equivalence: $\mathcal{A}$ is a logical consequence of $\mathcal{P}$ if, and only if, $\alpha_i \leq \mathcal{I}_\mathcal{P}(R_i) \Rightarrow \alpha \leq \mathcal{I}_\mathcal{P}(A)$. □

In the following results we relate the concepts of logical consequence and correct answer.

Theorem 4.3. *Let $\mathcal{P}$ be a multi-adjoint program and G a goal. If $\langle \lambda; \theta \rangle$ is a correct answer for $\mathcal{P}$ and G then $\langle G\theta; \lambda \rangle$ is a logical consequence of $\mathcal{P}$.*

Proof. Let $\mathcal{I}_\mathcal{P}$ be the least model of $\mathcal{P}$ and see that $\mathcal{I}_\mathcal{P}$ is model of $\langle G\theta; \lambda \rangle$. However, by definition of correct answer is verified that $\lambda \leq \mathcal{I}_\mathcal{P}(G\theta)$, as required. □

Theorem 4.4. *Let $\mathcal{P}$ be a multi-adjoint program and $\mathcal{A} = \langle A; \alpha \rangle$ a multi-adjoint formula such that A is a goal. If $\mathcal{A}$ is a logical consequence of $\mathcal{P}$, then the pair $\langle \alpha; id \rangle$ is a correct answer for $\mathcal{P}$ and A.*

Proof. By the Theorem 4.2, $\mathcal{I}_\mathcal{P}$ is model of $\mathcal{A}$, so that $\alpha \leq \mathcal{I}_\mathcal{P}(A)$ and therefore $\langle \alpha; id \rangle$ is a correct answer for $\mathcal{P}$ and A as wanted. □

The following theorem is the natural extension of the corresponding theorem of pure logic programming, included in [4], which characterizes the least Herbrand model as the set of formulae from the Herbrand base that are logical consequences of the multi-adjoint program.

Theorem 4.5. *Let $\mathcal{I}_\mathcal{P}$ be the least fuzzy Herbrand model of a multi-adjoint program $\mathcal{P}$ with associated lattice L. Then, $\mathcal{I}_\mathcal{P} = \{\mathcal{A} = \langle A; \alpha \rangle \in \mathcal{B}_\mathcal{P} \times L : \mathcal{A}$ is a logical consequence of $\mathcal{P}\}$.*

Proof. If $\mathcal{A} \in \mathcal{I}_\mathcal{P} \subset \mathcal{B}_\mathcal{P} \times L$, then $\alpha \leq \mathcal{I}_\mathcal{P}(A)$ so $\mathcal{A}$ is logical consequence of $\mathcal{P}$ and this shows that $\mathcal{I}_\mathcal{P} \subset \{\mathcal{A} \in \mathcal{B}_\mathcal{P} \times L : \mathcal{A}$ is a logical consequence of de $\mathcal{P}\}$. The reverse inclusion is analogous. □

 ISBN:xxxxxxxxx

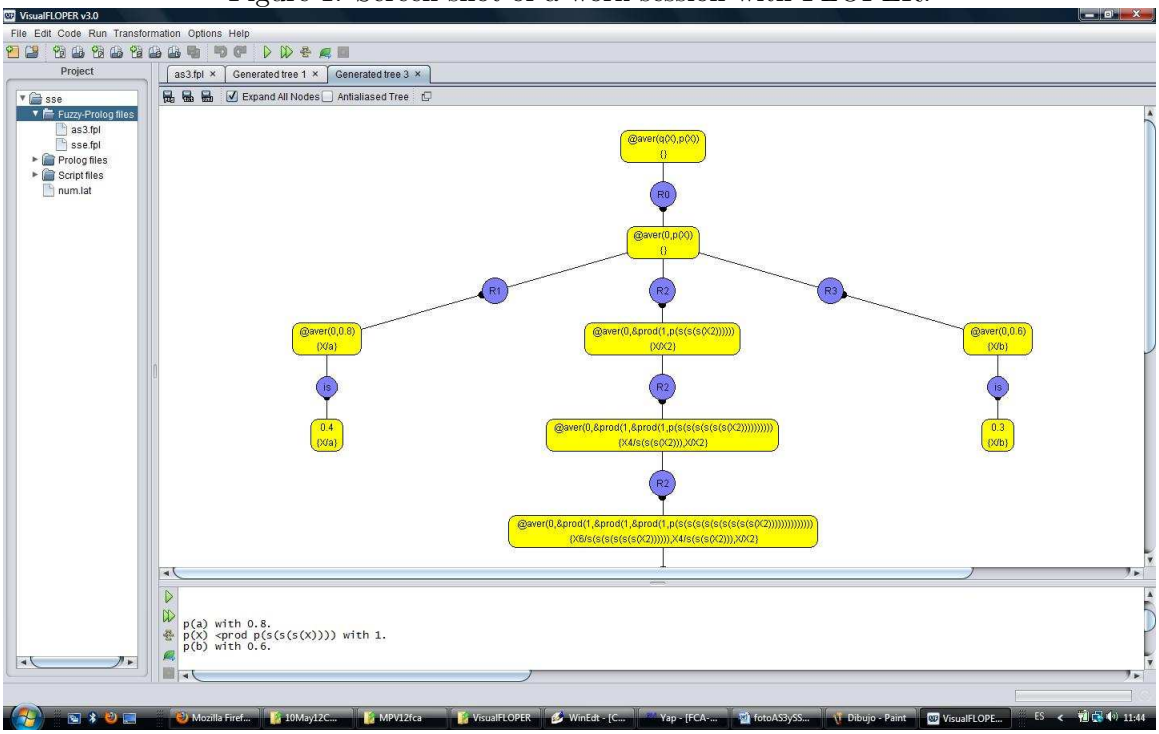
5 Conclusions and Future Work

In this paper we have recasted from [3] our concept of least fuzzy Herbrand model for multi-adjoint logic programs in order to characterize, in this setting, the notions of logical consequence and correct answer, thus extending the corresponding classic concepts of pure logic programming to this kind of fuzzy logic programs.

Furthermore, we have related in a very natural way both notions and we have proved that, as in the classic context, if we use a set-theoretic approach of the least model, it can be seen as the set of formulae of the Herbrand base that are logical consequence of the ones of the multi-adjoint program. Finally, we have provided an original proof of the soundness for the procedural semantics of MALP.

We are nowadays implementing most notions defined in this paper inside our "*Fuzzy LOgic Programming Environment for Research*" FLOPER (see Figure 1 and references [2, 9]), which is freely accessible from `http://dectau.uclm.es/floper/`.

Figure 1: Screen-shot of a work session with FLOPER.



 ISBN:xxxxxxxxx

References

[1] K. R. Apt. *From Logic Programming to Prolog.* International Series in Computer Science, Prentice Hall, 1997.

[2] P. Julián, G. Moreno, and J. Penabad. An Improved Reductant Calculus using Fuzzy Partial Evaluation Techniques. *Fuzzy Sets and Systems*, 160:162–181, 2009.

[3] P. Julián, G. Moreno, and J. Penabad. On the declarative semantics of multi-adjoint logic programs. In *Proceedings of the 10th International Work-Conference on Artificial Neural Networks, IWANN'09*, pages 253–260. Springer, Lecture Notes In Computer Science 5517, 2009.

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

[5] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. Multi-adjoint logic programming with continuous semantics. *Proceedings of Logic Programming and Non-Monotonic Reasoning, LPNMR'01, Springer, Lecture Notes in Artificial Intelligence*, 2173:351–364, 2001.

[6] J. Medina, M. Ojeda-Aciego, and P. Vojtáš. A procedural semantics for multi-adjoint logic programing. *Progress in Artificial Intelligence, EPIA'01, Springer-Verlag, Lecture Notes in Artificial Intelligence*, 2258(1):290–297, 2001.

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

[8] P.J. Morcillo and G. Moreno. Improving completeness in multi-adjoint logic computations via general reductants. In *Proc. of 2011 IEEE Symp. on Foundations of Computational Intelligence, April 11-15, Paris, France*, pages 138–145. IEEE, 2011.

[9] P.J. Morcillo, G. Moreno, J. Penabad, and C. Vázquez. A Practical Management of Fuzzy Truth Degrees using FLOPER. In M. Dean et al., editor, *Proceedings of 4th Intl Symposium on Rule Interchange and Applications, RuleML'10. Washington, USA, October 21–23*, pages 119–126. Springer Verlag, LNCS 6403, 2010.

[10] M. I. Sessa. Approximate reasoning by similarity-based SLD resolution. *Fuzzy Sets and Systems*, 275:389–426, 2002.

[11] U. Straccia, M. Ojeda-Aciego, and C. V. Damásio. On fixed-points of multivalued functions on complete lattices and their application to generalized logic programs. *SIAM Journal on Computing*, 38(5):1881–1911, 2009.

[12] P. Vojtáš and L. Paulík. Soundness and completeness of non-classical extended SLD-resolution. In R. Dyckhoff et al, editor, *Proceedings of ELP'96 Leipzig*, pages 289–301. Lecture Notes in Computer Science 1050, Springer Verlag, 1996.

 ISBN:xxxxxxxxx