Beyond Multi-Adjoint Logic Programming -cmmse-13[1]

Ginés Moreno[a], Jaime Penabad[b], Carlos Vázquez[a]

[a] *Department of Computing Systems, Faculty of Computer Science Engineering*
University of Castilla-La Mancha, 02071 Albacete (Spain)
[b] *Department of Mathematics, Faculty of Computer Science Engineering*
University of Castilla-La Mancha, 02071 Albacete (Spain)

Abstract

In "*Multi-Adjoint Logic Programming*", MALP in brief, each fuzzy logic program is associated with its own "*multi-adjoint lattice*" for modeling truth degrees beyond the simpler case of *true* and *false*, where a large set of fuzzy connectives can be defined. On this wide repertoire, it is crucial to connect each implication symbol with a proper conjunction thus conforming constructs of the form $(\leftarrow_i, \&_i)$ called "*adjoint pairs*", whose use directly affects both declarative and operational semantics of the MALP framework. In this work, we firstly show how the strong dependence of adjoint pairs can be largely weakened for an interesting "sub-class" of MALP programs. Then, we reason in a similar way till conceiving a "super-class" of fuzzy logic programs beyond MALP which definitively drops out the need for using adjoint pairs, since the new semantics behaviour relies on much more relaxed lattices than multi-adjoint ones.

Keywords: Multi-adjoint Logic Programming, Adjoint Pairs, Multi-adjoint Lattice

1. Introduction

MALP represents a modern and powerful approach for fuzzifying pure *Logic Programming* (LP) [12] by dealing with truth degrees and connectives col-

[1] Extended version of a previous work entitled "*Relaxing the Role of Adjoint Pairs in Multi-adjoint Logic Programming*" and presented in the "*13th International Conference on Mathematical Methods in Science and Engineering, CMMSE 2013 [23]*".

Email addresses: `Gines.Moreno@uclm.es` (Ginés Moreno), `Jaime.Penabad@uclm.es` (Jaime Penabad), `Carlos.Vazquez@uclm.es` (Carlos Vázquez)

lected from lattices more complex than the "*boolean*" case. Focusing on this setting, in [8, 19, 21, 22, 18, 1, 2, 6, 20] we describe some fundamental and practical issues regarding the design of declarative languages, programming environments and real-world applications with a fuzzy taste[2] developed in our research group during the last years. In essence, our research on MALP follows the fruitful path initiated by Lee in [11], where the classical inference mechanism of SLD–Resolution used in LP, has been replaced by a fuzzy variant able to handle partial truth and uncertainty in a comfortable way. Some other approaches trying to introduce inside the LP paradigm some techniques/constructs based on fuzzy logic can be found in the fuzzy logic programming systems [10, 4, 31, 5, 27, 24].

In the MALP framework [16], each program has its own associated multi-adjoint lattice L and each program rule is "weighted" with an element of L, whereas the components in its body are *linked* with connectives of the lattice. For instance, in the following propositional MALP program whose underlyin lattice is based on real numbers in the unit interval obeying *product's logic* (and where obviously $@_{\texttt{aver}}$ refers to the classical average operator):

$$\begin{array}{llll} p & \leftarrow & @_{\texttt{aver}}(q,r) & with \quad 0.9 \\ q & \leftarrow & & with \quad 0.8 \\ r & \leftarrow & & with \quad 0.6 \end{array}$$

the last two rules directly assign truth values 0.8 and 0.6 to propositional symbols q and r, respectively, and the execution of p using the first rule, simply consists in evaluating the expression "$0.9 * @_{\texttt{aver}}(0.8, 0.6)$", which returns the final truth degree 0.63.

On the other hand, in [8] we have obtained, for the first time in the literature related with MALP, 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 [21], we used this concept of least fuzzy Herbrand model to characterize correct answers and the notion of logical consequence, as well as their relationships.

After resuming in Sections 2 and 3 the syntax and operational/declarative semantics of MALP, the rest of the paper focuses on both a sub-class as well as a super-class of fuzzy logic programs non depending on adjoint pairs.

[2]Visit too `dectau.uclm.es/floper/` and `dectau.uclm.es/fuzzyXPath/`.

Firstly, in Section 4 we define the sub-class of MALP $\top$-programs which use simpler versions of models and computational steps than MALP. Next, in Section 5 we build a very simple technique able to map every MALP program with other one in this sub-class, thus moving the need for using adjoint pairs from the semantic core of MALP to a purely syntactic pre-process [23].

From here, in Section 6 we reason in a similar way for building the X-MALP super-class of fuzzy logic programs beyond MALP, which definitively drops out the need for using adjoint pairs since the new semantics behaviour relies on much more relaxed lattices (complete lattices) than multi-adjoint ones. Moreover, before concluding in Section 8, we detail the fix-point semantics of the new enlarged framework in Section 7.

2. Multi-adjoint Lattices and MALP Syntax

In essence, the notion of multi-adjoint lattice considers a *carrier* set L (whose elements verify a concrete ordering $\leq$) equipped with a set of connectives like implications, conjunctions, disjunctions and other *hybrid operators* (not always belonging to a standard taxonomy) with the particularity that for each implication symbol there exists its adjoint conjunction used for modeling the *modus ponens* inference rule in a fuzzy logic setting. For instance, some adjoint pairs -i.e. conjunctors and implications- in the lattice $([0,1], \leq)$ are presented in Figure 1, where labels `L`, `G` and `P` mean respectively *Łukasiewicz logic*, *Gödel logic* and *product logic* (with different capabilities for modeling *pessimist*, *optimist* and *realistic scenarios*, respectively).

$\&_{\mathtt{P}}(x,y)$	$\triangleq$	$x * y$	$\leftarrow_{\mathtt{P}}(x,y)$	$\triangleq$	$\min(1, x/y)$	*Product*
$\&_{\mathtt{G}}(x,y)$	$\triangleq$	$\min(x,y)$	$\leftarrow_{\mathtt{G}}(x,y)$	$\triangleq$	$\begin{cases} 1 & \text{if } y \leq x \\ x & \text{otherwise} \end{cases}$	*Gödel*
$\&_{\mathtt{L}}(x,y)$	$\triangleq$	$\max(0, x+y-1)$	$\leftarrow_{\mathtt{L}}(x,y)$	$\triangleq$	$\min\{x-y+1, 1\}$	*Łukasiewicz*

Figure 1: Adjoint pairs in $[0,1]$ for *Łukasiewicz*, *Gödel* and *product* fuzzy logics

Definition 2.1. *Let $(L, \leq)$ be a lattice. A* multi-adjoint lattice *is a tuple $(L, \leq, \leftarrow_1, \&_1, \ldots, \leftarrow_n, \&_n)$ such that:*

i) $(L, \leq)$ *is a complete lattice, namely,* $\forall S \subset L, S \neq \emptyset, \exists\, inf(S), sup(S)$[3].

ii) $(\&_i, \leftarrow_i)$ *is an* adjoint pair *in* $(L, \leq)$*, i.e.:*

1) $\&_i$ *is non-decreasing in both arguments, for all* i*,* $i = 1, \ldots, n$*.*

2) $\leftarrow_i$ *is non-decreasing in the first argument and non-increasing in the second one.*

3) $x \leq (y \leftarrow_i z)$ *if and only if* $(x \&_i z) \leq y$*, for any* $x, y, z \in L$ *(adjoint property)*[4]*.*

iii) $\top \&_i v = v \&_i \top = v$ *for all* $v \in L, i = 1, \ldots, n$*, where* $\top = sup(L)$*.*

In what follows, we present a short summary of the main features of the MALP language (we refer the reader to [16, 15] 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,

[3]Then, it is a bounded lattice, i.e. it has bottom and top elements, denoted by $\bot$ and $\top$, respectively.

[4]This condition is the most important feature of the framework.

Multi-adjoint logic program:

$$\mathcal{P} = \begin{cases} \mathcal{R}_1: & \langle p(X) & \leftarrow_{\mathtt{P}} & q(X,Y)\ \&_{\mathtt{G}}\ (r(Y)\ |_{\mathtt{L}}\ s(Y)) & ; & 0.8\rangle \\ \mathcal{R}_2: & \langle q(a,Y) & \leftarrow & & ; & 0.9\rangle \\ \mathcal{R}_3: & \langle r(b) & \leftarrow & & ; & 1\rangle \end{cases}$$

Admissible derivation:

$$\begin{array}{ll} \langle \underline{p(X)}; id\rangle & \overset{\text{AS1}}{\rightsquigarrow}\ \mathcal{R}_1 \\ \langle 0.8\ \&_{\mathtt{P}}\ (\underline{q(X_1,Y_1)}\ \&_{\mathtt{G}}\ (r(Y1)\ |_{\mathtt{L}}\ s(Y1)));\{X/X_1\}\rangle & \overset{\text{AS2}}{\rightsquigarrow}\ \mathcal{R}_2 \\ \langle 0.8\ \&_{\mathtt{P}}\ (0.9\ \&_{\mathtt{G}}\ (\underline{r(Y2)}\ |_{\mathtt{L}}\ s(Y2)));\{X/a, X_1/a, Y_1/Y_2\}\rangle & \overset{\text{AS2}}{\rightsquigarrow}\ \mathcal{R}_3 \\ \langle 0.8\ \&_{\mathtt{P}}\ (0.9\ \&_{\mathtt{G}}\ (1\ |_{\mathtt{L}}\ \underline{s(Y2)}));\{X/a, X_1/a, Y_1/b, Y_2/b\}\rangle & \overset{\text{AS3}}{\rightsquigarrow} \\ \langle 0.8\ \&_{\mathtt{P}}\ (0.9\ \&_{\mathtt{G}}\ (1\ |_{\mathtt{L}}\ 0));\{X/a, X_1/a, Y_1/b, Y_2/b\}\rangle & \end{array}$$

Interpretive derivation:

$$\begin{array}{ll} \langle 0.8\ \&_{\mathtt{P}}\ (0.9\ \&_{\mathtt{G}}\ \underline{(1\ |_{\mathtt{L}}\ 0)});\{X/a\}\rangle & \overset{\text{IS}}{\rightsquigarrow} \\ \langle 0.8\ \&_{\mathtt{P}}\ \underline{(0.9\ \&_{\mathtt{G}}\ 1)};\{X/a\}\rangle & \overset{\text{IS}}{\rightsquigarrow} \\ \langle \underline{0.8\ \&_{\mathtt{P}}\ 0.9};\{X/a\}\rangle & \overset{\text{IS}}{\rightsquigarrow} \\ \langle 0.72;\{X/a\}\rangle & \end{array}$$

— f.c.a. means "*p(X)* is proved with truth degree 0.72 when $X = a$".

Least fuzzy Herbrand model:

$\mathcal{I}_{\mathcal{P}} : \mathcal{B}_{\mathcal{P}} \to [0,1]$ s.t. $\mathcal{I}_{\mathcal{P}}(p(a)) = 0.72$, $\mathcal{I}_{\mathcal{P}}(q(a,a)) = \mathcal{I}_{\mathcal{P}}(q(a,b)) = 0.9$, and $\mathcal{I}_{\mathcal{P}}(r(b)) = 1$

Figure 2: Illustrative examples of MALP syntax and semantics

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 MALP 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 $\mathcal{R}$, as illustrated in Figure 2.

3. Operational and Declarative Semantics of MALP

In order to describe the procedural semantics of the MALP 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 a set of expressions 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 3.1 (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 $\overset{AS}{\rightsquigarrow}$ is the smallest relation satisfying the following* admissible rules*:*

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

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

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

An *admissible derivation* is a sequence $\langle \mathcal{Q}; id \rangle \overset{\text{AS}}{\rightsquigarrow}{}^{*} \langle \mathcal{Q}'; \theta \rangle$. As usual, rules are taken renamed apart. We shall use the symbols $\overset{\text{AS1}}{\rightsquigarrow}$, $\overset{\text{AS2}}{\rightsquigarrow}$ and $\overset{\text{AS3}}{\rightsquigarrow}$ to distinguish between computation steps performed by applying one of the specific admissible rules. The application of a rule on a step will be annotated as a superscript of the $\overset{\text{AS}}{\rightsquigarrow}$ symbol.

If we exploit all atoms of a given goal, by applying admissible steps as much as needed during the operational phase, then it becomes a formula with no atoms (a L-expression) which can be then interpreted w.r.t. lattice L as follows.

Definition 3.2 (Interpretive Step and Fuzzy Computed Answer)**.** *Let $\mathcal{P}$ be a program, $\mathcal{Q}$ a goal and σ a substitution. Assume that $[\![@]\!]$ is the truth function of connective @ in the lattice $(L, \leq)$ associated to $\mathcal{P}$, such that, for values $r_1, \ldots, r_n, r_{n+1} \in L$, we have that $[\![@]\!](r_1, \ldots, r_n) = r_{n+1}$. Then, we formalize the notion of* interpretive computation *as a state transition system, whose transition relation $\overset{IS}{\rightsquigarrow}$ is defined as the least one satisfying:*

$$\langle \mathcal{Q}[@(r_1, \ldots, r_n)]; \sigma \rangle \quad \overset{IS}{\rightsquigarrow} \quad \langle \mathcal{Q}[@(r_1, \ldots, r_n)/r_{n+1}]; \sigma \rangle$$

An interpretive derivation *is a sequence* $\langle \mathcal{Q}; \sigma \rangle \overset{IS}{\rightsquigarrow} \cdots \overset{IS}{\rightsquigarrow} \langle \mathcal{Q}'; \sigma \rangle$. *When* $\mathcal{Q}' = r \in L$, *the state* $\langle r; \sigma \rangle$ *is called a* fuzzy computed answer *(f.c.a.) for that derivation.*

Moreover, in the MALP framework [16, 15], 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 (i.e., atoms and elements of L) are *linked* with connectives of the lattice.

We formally introduce now the semantic notions of Herbrand interpretation and Herbrand model, or directly, interpretation and model for short, for a MALP program $\mathcal{P}$, in a similar way to [16] and [8].

Definition 3.3 (Herbrand Interpretation)**.** *A Herbrand interpretation is a map* $\mathcal{I} : B_\mathcal{P} \to L$, *where* $B_\mathcal{P}$ *is the Herbrand base of the* MALP *program* $\mathcal{P}$ *and* $(L, \leq)$ *is the multi-adjoint lattice associated to program* $\mathcal{P}$.

Definition 3.4 (Herbrand Model)**.** *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 a Herbrand model of* $\mathcal{P}$ *if, and only if, all rules in* $\mathcal{P}$ *are satisfied by* $\mathcal{I}$.

In [8] 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 [12, 3], 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 ([32, 25, 28]), in alternative fuzzy logic programming frameworks different from MALP.

Furthermore, in [8] 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 3.5 (Least Fuzzy Herbrand Model)**.** *Let* $\mathcal{P}$ *be a* MALP *program with associated lattice* $(L, \leq)$. *The least fuzzy Herbrand*[5] *model of* $\mathcal{P}$ *is the interpretation* $\mathcal{I}_\mathcal{P} = inf\{\mathcal{I}_j : \mathcal{I}_j \text{ is model of } \mathcal{P}\}$.

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

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

Theorem 3.6 ([8]). *Let $\mathcal{P}$ be a* MALP *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 [8], 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 3.6.

In Figure 2 we illustrate most definitions presented in this section, where we wish to remark that:

- In the first rule of $\mathcal{P}$, we mix connectives belonging to three different fuzzy logics, whose truth functions appear in Figure 1, having too that $|_{\text{L}}(x, y) \triangleq min(x + y, 1)$.
- Note that since there are no rules defining predicate s, the last step in the admissible derivation reduces $s(Y_2)$ to 0 by applying an $\overset{\text{AS3}}{\rightsquigarrow}$ step, which contrasts with crisp logic languages such as PROLOG which would abort the whole derivation. Hence, in our fuzzy setting we can reach computed answers (at the end of the interpretive phase) even in the presence of non defined predicates.
- When describing the least fuzzy Herbrand model of $\mathcal{P}$, we omit those elements of the Herbrand Base interpreted as 0.

4. A MALP Sub-class non Depending on Adjoint Pairs

From now on, we call $\mathcal{M}_{\top}$ to the set of MALP programs whose rules are always labeled with the top element $\top$ of their associated lattices. We now speak about $\top$-programs, $\top$-rules and so on. For executing these programs, we can conceive the following operational semantics which is simpler than the one seen in the previous section (see Definition 3.1).

Definition 4.1 ($\top$-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 $\top$-program $\mathcal{P} \in \mathcal{M}_{\top}$, a* $\top$-admissible computation *is formalized as a state transition system, whose transition relation $\overset{AS^{\top}}{\rightsquigarrow}$ is the smallest relation satisfying the following* $\top$-admissible rules*:*

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

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

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

A $\top$-*admissible derivation* is a sequence $\langle \mathcal{Q}; id \rangle \overset{\mathrm{AS}^\top}{\rightsquigarrow} {}^*\langle \mathcal{Q}'; \theta \rangle$. We shall use the symbols $\overset{\mathrm{AS1}^\top}{\rightsquigarrow}$, $\overset{\mathrm{AS2}^\top}{\rightsquigarrow}$ and $\overset{\mathrm{AS3}^\top}{\hookrightarrow}$ to distinguish between computation steps performed by applying one of the specific admissible rules. Note that this definition (which is very close to the classical one of SLD-resolution used in PROLOG) differs for Definition 3.1 just in the first case, since it does not make use on states of the $\&_i$ conjunction adjoint to the $\leftarrow_i$ implication symbol of $\top$-rules.
The following result establishes that, when dealing with $\top$-programs, the derivations built with admissible (together with subsequent interpretive) steps as well as those ones based on $\top$-admissible (and interpretive) steps, lead to the same set of fuzzy computed answers.

Theorem 4.2. *Let* $\mathcal{P} \in \mathcal{M}_\top$ *be a* MALP $\top$*-program with associated lattice* L, $\mathcal{Q}$ *a goal,* σ *a substitution and* $v \in L$. *Then,*

$$\langle \mathcal{Q}; id \rangle \overset{AS}{\rightsquigarrow} {}^* \cdots \overset{IS}{\rightsquigarrow} {}^*\langle v; \sigma \rangle \text{ w.r.t. } \mathcal{P} \quad \textit{iff} \quad \langle \mathcal{Q}; id \rangle \overset{AS^\top}{\rightsquigarrow} {}^* \cdots \overset{IS}{\rightsquigarrow} {}^*\langle v; \sigma \rangle \text{ w.r.t. } \mathcal{P}.$$

Proof. In our our proof we simply need to show that the effects produced by $\overset{\mathrm{AS}}{\rightsquigarrow}$ steps on a generic state of the form $\langle \mathcal{Q}[A]; \sigma \rangle$, are replicated by $\overset{\mathrm{AS}^\top}{\rightsquigarrow}$ steps and viceversa. We consider three different cases:

1) If $\langle H \leftarrow_i \mathcal{B}; \top \rangle \in \mathcal{P}$, where $\mathcal{B}$ is not empty and $\theta = mgu(\{H = A\})$, then $\langle \mathcal{Q}[A]; \sigma \rangle \overset{\mathrm{AS1}}{\rightsquigarrow} \langle (\mathcal{Q}[A/\top \&_i \mathcal{B}])\theta; \sigma\theta \rangle$ if and only if $\langle \mathcal{Q}[A]; \sigma \rangle \overset{\mathrm{AS1}^\top}{\rightsquigarrow} \langle (\mathcal{Q}[A/\mathcal{B}])\theta; \sigma\theta \rangle$, since $\top \&_i \mathcal{B} \equiv \mathcal{B}$ and hence $(\mathcal{Q}[A/\top \&_i \mathcal{B}])\theta \equiv (\mathcal{Q}[A/\mathcal{B}])\theta$.

2) If $\langle H \leftarrow_i; \top \rangle \in \mathcal{P}$, where $\theta = mgu(\{H = A\})$, then $\langle \mathcal{Q}[A]; \sigma \rangle \overset{\mathrm{AS2}}{\rightsquigarrow}$ $\langle (\mathcal{Q}[A/\top])\theta; \sigma\theta \rangle$ if and only if $\langle \mathcal{Q}[A]; \sigma \rangle \overset{\mathrm{AS2}^\top}{\rightsquigarrow} \langle (\mathcal{Q}[A/\top])\theta; \sigma\theta \rangle$.

3) If there is no $\top$-rule in $\mathcal{P}$ whose head unifies with A then, $\langle \mathcal{Q}[A]; \sigma \rangle \overset{\mathrm{AS3}}{\rightsquigarrow}$ $\langle (\mathcal{Q}[A/\bot]); \sigma \rangle$ if and only if $\langle \mathcal{Q}[A]; \sigma \rangle \overset{\mathrm{AS3}^\top}{\hookrightarrow} \langle (\mathcal{Q}[A/\bot]); \sigma \rangle$.

□

Let us continue now with declarative semantics aspects related with $\top$-programs.

Definition 4.3 ($\top$-model)**.** *An interpretation $\mathcal{I}$ $\top$-satisfies (or is* $\top$-model *of) a $\top$-rule $\langle H \leftarrow_i \mathcal{B}; \top\rangle$ if, and only if, $\mathcal{I}(\mathcal{B}) \leq \mathcal{I}(H)$. An interpretation $\mathcal{I}$ is $\top$-model of a $\top$-program $\mathcal{P}$ if, and only if, all $\top$-rules in $\mathcal{P}$ are $\top$-satisfied by $\mathcal{I}$.*

Definition 4.4 (Least Fuzzy Herbrand $\top$-Model)**.** *Let $\mathcal{P} \in \mathcal{M}_\top$ be a* MALP $\top$*-program with associated lattice $(L, \leq)$. The interpretation $\mathcal{I}_\mathcal{P}^\top = inf\{\mathcal{I}_j : \mathcal{I}_j$ is $\top$-model of $\mathcal{P}\}$ is the least fuzzy Herbrand $\top$-model of $\mathcal{P}$.*

In the following result we prove that the notion of least fuzzy Herbrand model of a given MALP $\top$-program is just the same construct than its least fuzzy Herbrand $\top$-model.

Theorem 4.5. *The least fuzzy Herbrand model of a* MALP $\top$*-program $\mathcal{P} \in \mathcal{M}_\top$ coincides with its least fuzzy Herbrand $\top$-model, that is, $\mathcal{I}_\mathcal{P} = \mathcal{I}_\mathcal{P}^\top$.*

Proof. Consider a generic $\top$-rule $\mathcal{R} : \langle H \leftarrow_i \mathcal{B}; \top\rangle \in \mathcal{P}$ and an interpretation $\mathcal{I}$. In order to prove that $\mathcal{I}$ satisfies $\mathcal{R}$ if and only if it $\top$-satisfies $\mathcal{R}$, it suffices by showing that $\top \leq \mathcal{I}(H \leftarrow_i \mathcal{B})$ becomes into $\mathcal{I}(\top \&_i \mathcal{B}) \leq \mathcal{I}(H)$ thanks to the adjoint property, and then this expression can be simplified to $\mathcal{I}(\mathcal{B}) \leq \mathcal{I}(H)$ (since obviousy $\top \&_i v = v$, for any $v \in L$), as desired. So, since the set of models for a given $\top$-program $\mathcal{P}$ is the same as its set of $\top$-models, the infimmun of such set is just $\mathcal{I}_\mathcal{P} = \mathcal{I}_\mathcal{P}^\top$, which concludes our proof. □

It is noteworthy that, in the previous definitions related with $\top$-models and $\mathcal{I}_\mathcal{P}^\top$, we don't require that the lattice associated to MALP $\top$-programs be a multi-adjoint lattice (in fact, we never use adjoint pairs), as occurred too when defining the new operational semantics for $\top$-programs. For this reason, from now on we can simplify the syntax of $\top$-rules, by removing the label of their implication symbols as well as their weights (or associated truth degrees), i.e., instead of $\langle H \leftarrow_i \mathcal{B}; \top\rangle$ we will simply write $H \leftarrow \mathcal{B}$.

5. A Mapping from MALP Programs to $\mathcal{M}_\top$

The following definition represents a very simple, purely syntactic preprocess which, by making use of adjoint pairs, is able to link MALP programs with $\top$-programs.

Multi-adjoint logic $\top$-program:

$$\mathcal{P}' = \mathcal{P}^{\mathcal{M}_\top} = \begin{cases} \mathcal{R}'_1: & p(X) & \leftarrow & 0.8 \;\&_{\mathtt{P}}\; (q(X,Y) \;\&_{\mathtt{G}}\; (r(Y) \;|_{\mathtt{L}}\; s(Y)) \\ \mathcal{R}'_2: & q(a,Y) & \leftarrow & 0.9 \\ \mathcal{R}'_3: & r(b) & \leftarrow & \end{cases}$$

$\top$-Admissible derivation:

$$\begin{array}{ll} \langle \underline{p(X)}; id\rangle & \overset{\text{AS1}^\top}{\rightsquigarrow}\ \mathcal{R}'_1 \\ \langle 0.8 \;\&_{\mathtt{P}}\; (\underline{q(X_1,Y_1)} \;\&_{\mathtt{G}}\; (r(Y1) \;|_{\mathtt{L}}\; s(Y1)));\{X/X_1\}\rangle & \overset{\text{AS1}^\top}{\rightsquigarrow}\ \mathcal{R}'_2 \\ \langle 0.8 \;\&_{\mathtt{P}}\; (0.9 \;\&_{\mathtt{G}}\; (\underline{r(Y2)} \;|_{\mathtt{L}}\; s(Y2)));\{X/a, X_1/a, Y_1/Y_2\}\rangle & \overset{\text{AS2}^\top}{\rightsquigarrow}\ \mathcal{R}'_3 \\ \langle 0.8 \;\&_{\mathtt{P}}\; (0.9 \;\&_{\mathtt{G}}\; (1 \;|_{\mathtt{L}}\; \underline{s(Y2)}));\{X/a, X_1/a, Y_1/b, Y_2/b\}\rangle & \overset{\text{AS3}^\top}{\hookrightarrow} \\ \langle 0.8 \;\&_{\mathtt{P}}\; (0.9 \;\&_{\mathtt{G}}\; (1 \;|_{\mathtt{L}}\; \overline{0}));\{X/a, X_1/a, Y_1/b, Y_2/b\}\rangle & \end{array}$$

Figure 3: Illustrative examples of concepts defined in Sections 4 and 5

Definition 5.1. *We define a mapping that associates to each* MALP *program $\mathcal{P}$ a $\top$-program in $\mathcal{M}_\top$ with the following shape:*

$$\mathcal{P}^{\mathcal{M}_\top} = \{\mathcal{R}^{\mathcal{M}_\top} \;:\; \mathcal{R} \in \mathcal{P}\}$$

where for each $\top$-rule $\mathcal{R}: \langle H \leftarrow_i \mathcal{B}; v\rangle \in \mathcal{P}$, the mapping is defined too as:

$$\mathcal{R}^{\mathcal{M}_\top} \;=\; \begin{cases} H \leftarrow v \&_i \mathcal{B} & \text{if} \quad v \neq \top \\ H \leftarrow \mathcal{B} & \text{otherwise} \end{cases}$$

In Figure 3 we illustrate this definition as well as other concepts introduced in the previous section. Note that:

- The $\top$-program $\mathcal{P}'$ coincides with the transformation of the MALP program $\mathcal{P}$ seen in Figure 2, that is $\mathcal{P}' = \mathcal{P}^{\mathcal{M}_\top}$, since $\mathcal{R}'_1 = \mathcal{R}_1^{\mathcal{M}_\top}$, $\mathcal{R}'_2 = \mathcal{R}_2^{\mathcal{M}_\top}$ and $\mathcal{R}'_3 = \mathcal{R}_3^{\mathcal{M}_\top}$. In this last case, we have simply removed the weight of the rule (since it is just 1, i.e., the top element of lattice $[0,1]$) and both $\mathcal{R}_1$ and $\mathcal{R}'_1$ are facts in $\mathcal{P}$ and $\mathcal{P}'$, respectively.
- On the other hand, note that even when $\mathcal{R}_2 \in \mathcal{P}$ is a fact, $\mathcal{R}'_2 \in \mathcal{P}'$ is not a fact, since its body is not empty (it is composed by just a truth degree). For this reason, while the second admissible step of the

admissible derivation in Figure 2 is of kind $\overset{\text{AS2}}{\rightsquigarrow}$, the corresponding second $\top$-admissible step of the $\top$-admissible derivation in Figure 3 is not a $\overset{\text{AS2}^{\top}}{\rightsquigarrow}$ but a $\overset{\text{AS1}^{\top}}{\rightsquigarrow}$ step.

- The sequence of states in the admissible derivation of Figure 2 coincides with the sequence of states in the $\top$-admissible of Figure 3, and after applying exactly the same sequence of interpretive steps drawn in Figure 2 (for this reason we have omitted it in Figure 3), the same fuzzy computed answer is reached.

- Note that even when the notion of $\top$-model[6] is less involved than the one of model, the least fuzzy Herbrand $\top$-model of $\mathcal{P}'$ coincides with $\mathcal{I}_{\mathcal{P}}$ in Figure 2, as wanted.

The following result establishes that derivations built with admissible (together with subsequent interpretive) steps lead to the same set of fuzzy computed answers than those ones based on $\top$-admissible (and interpretive) steps when dealing with $\top$-programs obtained from previous MALP programs after being transformed according Definition 5.1.

Theorem 5.2. *Let $\mathcal{P}$ be a* MALP *program with associated lattice L, $\mathcal{Q}$ a goal, σ a substitution and $v \in L$. Then,*

$$\langle \mathcal{Q}; id\rangle \overset{AS}{\rightsquigarrow}{}^{*} \dots \overset{IS}{\rightsquigarrow}{}^{*} \langle v; \sigma\rangle \text{ w.r.t. } \mathcal{P} \text{ iff } \langle \mathcal{Q}; id\rangle \overset{AS^{\top}}{\rightsquigarrow}{}^{*} \dots \overset{IS}{\rightsquigarrow}{}^{*} \langle v; \sigma\rangle \text{ w.r.t. } \mathcal{P}^{\mathcal{M}_{\top}}.$$

Proof. We distinguish four cases for showing that the effects produced by $\overset{\text{AS}}{\rightsquigarrow}$ steps on a generic state of the form $\langle \mathcal{Q}[A]; \sigma\rangle$, are replicated by $\overset{\text{AS}^{\top}}{\rightsquigarrow}$ steps and viceversa.

1) Note that $\langle H \leftarrow_i \mathcal{B}; v\rangle \in \mathcal{P}$, where $\mathcal{B}$ is not empty and $\theta = mgu(\{H = A\})$, if and only if $\langle H \leftarrow v \&_i \mathcal{B}\rangle \in \mathcal{P}^{\mathcal{M}_{\top}}$, and hence $\langle \mathcal{Q}[A]; \sigma\rangle \overset{\text{AS1}}{\rightsquigarrow} \langle (\mathcal{Q}[A/v \&_i \mathcal{B}])\theta; \sigma\theta\rangle$ if and only if $\langle \mathcal{Q}[A]; \sigma\rangle \overset{\text{AS1}^{\top}}{\rightsquigarrow} \langle (\mathcal{Q}[A/v \&_i \mathcal{B}])\theta; \sigma\theta\rangle$.

2) Now, $\langle H \leftarrow; v\rangle \in \mathcal{P}$, where $v \neq \top$, $\theta = mgu(\{H = A\})$ if and only if $\langle H \leftarrow v\rangle \in \mathcal{P}^{\mathcal{M}_{\top}}$, and hence $\langle \mathcal{Q}[A]; \sigma\rangle \overset{\text{AS2}}{\rightsquigarrow} \langle (\mathcal{Q}[A/v])\theta; \sigma\theta\rangle$ if and only if $\langle \mathcal{Q}[A]; \sigma\rangle \overset{\text{AS1}^{\top}}{\rightsquigarrow} \langle (\mathcal{Q}[A/v])\theta; \sigma\theta\rangle$ (note this particular correspondence between $\overset{\text{AS2}}{\rightsquigarrow}$ and $\overset{\text{AS1}^{\top}}{\rightsquigarrow}$ steps).

[6] This concept does not make use of adjoint pairs and weights of program rules.

3) Observe that $\langle H\leftarrow;\top\rangle \in \mathcal{P}$, where $\theta = mgu(\{H = A\})$, if and only if $\langle H\leftarrow\rangle \in \mathcal{P}^{\mathcal{M}_\top}$, and hence $\langle \mathcal{Q}[A];\sigma\rangle \overset{\text{AS2}}{\rightsquigarrow} \langle(\mathcal{Q}[A/\top])\theta;\sigma\theta\rangle$ if and only if $\langle \mathcal{Q}[A];\sigma\rangle \overset{\text{AS2}^\top}{\rightsquigarrow} \langle(\mathcal{Q}[A/\top])\theta;\sigma\theta\rangle$ (now we have shown the equivalence between $\overset{\text{AS2}}{\rightsquigarrow}$ and $\overset{\text{AS2}^\top}{\rightsquigarrow}$ steps).

4) Finally, there is no rule in $\mathcal{P}$ whose head unifies with A if and only if there is no rule in $\mathcal{P}^{\mathcal{M}_\top}$ whose head unifies with A and so, $\langle \mathcal{Q}[A];\sigma\rangle \overset{\text{AS3}}{\rightsquigarrow} \langle(\mathcal{Q}[A/\bot]);\sigma\rangle$ if and only if $\langle \mathcal{Q}[A];\sigma\rangle \overset{\text{AS3}^\top}{\hookrightarrow} \langle(\mathcal{Q}[A/\bot]);\sigma\rangle$.

□

In the following result we prove that the notion of least fuzzy Herbrand model of MALP programs is just the same construct than the least fuzzy Herbrand $\top$-model of those $\top$-programs obtained by applying the transformation process described in Definition 5.1.

Theorem 5.3. *The least fuzzy Herbrand model of a* MALP *program $\mathcal{P}$ coincides with the least fuzzy Herbrand $\top$-model of its associated $\top$-program $\mathcal{P}^{\mathcal{M}_\top}$, that is, $\mathcal{I}_\mathcal{P} = \mathcal{I}^\top_{\mathcal{P}^{\mathcal{M}_\top}}$.*

Proof. Consider a generic rule $\mathcal{R} : \langle H\leftarrow_i \mathcal{B}; v\rangle \in \mathcal{P}$ and correspondingly $\mathcal{R}^{\mathcal{M}_\top} : H\leftarrow v\&_i\mathcal{B} \in \mathcal{P}^{\mathcal{M}_\top}$. Assume an interpretation $\mathcal{I}$ such that $\mathcal{I}$ satisfies $\mathcal{R}$ if and only if $\mathcal{I}$ $\top$-satisfies $\mathcal{R}^{\mathcal{M}_\top}$ since by the adjoint property $v \leq \mathcal{I}(H\leftarrow_i \mathcal{B})$ if and only if $\mathcal{I}(v\&_i \mathcal{B}) \leq \mathcal{I}(H)$ and hence, the set of models of $\mathcal{P}$ coincides with the set of $\top$-models of $\mathcal{P}^{\mathcal{M}_\top}$ and, in particular, the infimmum of such set is $\mathcal{I}_\mathcal{P}$ as well as $\mathcal{I}^\top_{\mathcal{P}^{\mathcal{M}_\top}}$, as wanted. □

6. The Wider Class of X-MALP Programs

From now on, X-MALP stands for *relaXed Multi-Adjoint Logic Programming*. In contrast with multi-adjoint lattices associated to MALP programs (and also MALP $\top$-programs), in the new X-MALP class, truth degrees are modeled in a more relaxed kind of lattices (exactly, complete lattices) non requiring the presence of adjoint pairs, which justifies why MALP is really a subset of X-MALP (see Figure 4). In general, the syntactic shape of X-MALP program rules coincides with the one of MALP $\top$-programs, that is, they are expressions of the form $H \leftarrow_i \mathcal{B}$, where H is an atomic formula (the *head*) and $\mathcal{B}$ (the *body*) is a formula built from atomic formulas $B_1, \ldots, B_n$ ($n \geq 0$), truth values of the associated lattice and conjunctions,

disjunctions and aggregations. Moreover, we also allow weighted rules to embed MALP rules, but it is important to note that in the new framework, such weights are purely "syntactic sugar" in the sense that any expression of the form $\mathcal{R} : \langle H \leftarrow_i \mathcal{B}; v \rangle$ refers to the X-MALP rule $\mathcal{R} : H \leftarrow v \&_i \mathcal{B}$ where, in the associated lattice, it is only required that the conjunction symbol $\&_i$ be defined, but not the whole adjoint pair $\langle \leftarrow_i, \&_i \rangle$ (in fact, the implication symbol $\leftarrow_i$ will be never used when defining the semantics of X-MALP, in contrast with MALP). The following definition shows the general syntax of X-MALP programs, highlighting the expressive power of this style of fuzzy logic programming which easily covers other well-established frameworks.

Definition 6.1. *A* X-MALP program $\mathcal{P}$*, with associated complete lattice* $(L, \leq)$*, is a set of rules* $A \leftarrow \mathcal{B}$ *verifying:*

i) A is an atomic formula (usually called "head").

ii) $\mathcal{B}$ *is an arbitrary formula ("body") built with atoms* $B_1, \ldots, B_n$*,* $n \geq 0$ *and any conjunctions, disjunctions, aggregations and truth degrees (i.e., elements collected from the underlying lattice* L*).*

Moreover, the rules of $\mathcal{P}$ will be expressed too by using the syntactic expression $A \leftarrow f(B_1, \ldots, B_n)$, where f is a computable function that is the result of combining all the connective present in the body (this same syntax can be explicitly found in [31, 13, 14] and it is also accepted in many other fuzzy logic programming frameworks). Program rules are interpreted in a complete lattice $(L, \leq)$ which is previously associated to $\mathcal{P}$.

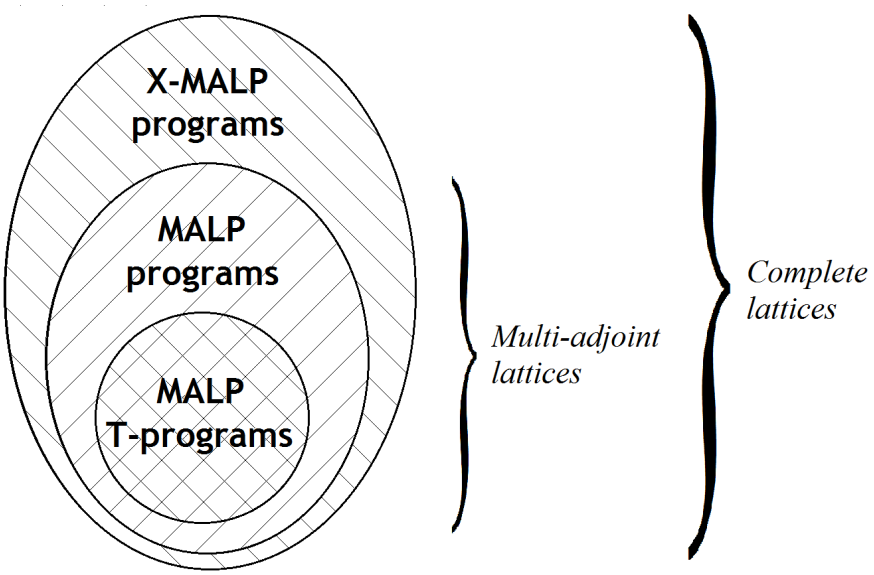
Now, it is mandatory to re-adapt in an obvious way all the definitions presented in Section 4 for referring now to X-MALP programs instead of MALP $\top$-programs.

Definition 6.2 (Admissible Step for X-MALP programs)**.** *Let* $\mathcal{Q}$ *be a goal and let* σ *be a substitution. The pair* $\langle \mathcal{Q}; \sigma \rangle$ *is a* state*. Given a* X-MALP*program* $\mathcal{P}$*, an* admissible computation *is formalized as a state transition system, whose transition relation* $\overset{AS}{\hookrightarrow}$ *is the smallest relation satisfying the following* admissible rules*:*

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

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

Figure 4: Comparing MALP programs, MALP $\top$-programs and X-MALP programs



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

Similarly to the MALP case, if we exploit all atoms of a given goal, by applying admissible steps as much as needed during the operational phase, then it becomes a formula with no atoms which can be then interpreted w.r.t. complete lattice L for obtaining the set of fuzzy computed answers according Definition 3.2.

In the following, we formally introduce the semantic notion of Herbrand model for X-MALP programs, similarly to how we have proceeded in the MALP case. We will base our declarative semantics only on fuzzy Herbrand interpretations similarly to what it is considered, among others, in [32].

Definition 6.3 (Herbrand Interpretation)**.** *A Herbrand interpretation is a map* $\mathcal{I} : B_\mathcal{P} \to L$*, where* $B_\mathcal{P}$ *is the Herbrand base of the* X-MALP *program* $\mathcal{P}$ *and* $(L, \leq)$ *is the complete lattice associated to* $\mathcal{P}$*.*

$\mathcal{I}$ 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 X-MALP language), it suffices to take

$\mathcal{I}(A) = inf\{\mathcal{I}(A\xi) : A\xi$ is a ground instance of A$\}$. Let $\mathcal{H}$ be the set of Herbrand 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 $(L, \leq)$.

Definition 6.4 (Herbrand Model of a X-MALP program)**.** *An Herbrand interpretation $\mathcal{I}$ satisfies (or is* Herbrand model *of) a* X-MALP *rule $H \leftarrow \mathcal{B}$ if, and only if, $\mathcal{I}(\mathcal{B}) \leq \mathcal{I}(H)$. An Herbrand interpretation $\mathcal{I}$ is Herbrand model of a* X-MALP *program $\mathcal{P}$ if, and only if, all rules in $\mathcal{P}$ are satisfied by $\mathcal{I}$.*

Definition 6.5 (Least Fuzzy Herbrand Model of a X-MALP program)**.** *Let $\mathcal{P}$ be a* X-MALP *program with associated (complete) lattice $(L, \leq)$. The least fuzzy Herbrand model of $\mathcal{P}$ is the interpretation $\mathcal{I}_{\mathcal{P}} = inf\{\mathcal{I}_j :$ $\mathcal{I}_j$ is Herbrand model of $\mathcal{P}\}$.*

The following result justifies why the previous interpretation $\mathcal{I}_{\mathcal{P}}$ can be really thought as the least fuzzy Herbrand model. It is very important to contrast it with its homologous one for MALP programs (Theorem 3.6), since in the proof of the new result adapted to X-MALP obviously does not rely on adjoint pairs.

Theorem 6.6. *Let $\mathcal{P}$ be a* X-MALP *program with associated (complete) lattice L. The mapping $\mathcal{I}_{\mathcal{P}} = inf\{\mathcal{I}_j : \mathcal{I}_j$ is a model of $\mathcal{P}\}$ is the least Herbrand model of $\mathcal{P}$.*

Proof. In the following, we denote by $\mathcal{M}$ the set of Herbrand model interpretations of $\mathcal{P}$, that is, $\mathcal{M} = \{\mathcal{I}_j : \mathcal{I}_j$ is a Herbrand model of $\mathcal{P}\}$. Note that $\mathcal{M}$ is not empty, being as the Herbrand interpretation $\mathcal{I} = sup(\mathcal{H})$, defined on each $A \in \mathcal{B}_{\mathcal{P}}$ by $\mathcal{I}(A) = sup(L)$, is a Herbrand model (of all rules) of $\mathcal{P}$.

Then, given that $\mathcal{I}_{\mathcal{P}} = inf(\mathcal{M})$, $\mathcal{I}_{\mathcal{P}}$ is a Herbrand interpretation: since $(\mathcal{H}, \leq)$ is a complete lattice, there exists the infimum of the subset $\mathcal{M}$ and it is a member of $\mathcal{H}$.

We prove now that $\mathcal{I}_{\mathcal{P}}$ is a Herbrand model of $\mathcal{P}$, that is, it satisfies all rules of $\mathcal{P}$.

Indeed, let $\mathcal{R} : H \leftarrow \mathcal{B}$ be a rule of $\mathcal{P}$. Since $\mathcal{I}_{\mathcal{P}}$ is the infimum of $\mathcal{M}$, $\mathcal{I}_{\mathcal{P}} \leq \mathcal{I}_j$, for each model $\mathcal{I}_j$ of $\mathcal{P}$. Therefore, $\mathcal{I}_{\mathcal{P}}(A) \leq \mathcal{I}_j(A)$ for each atom

A. Moreover, given that $\mathcal{I}_j$ is a Herbrand model of $\mathcal{P}$, $\mathcal{I}_j$ satisfies each rule $\mathcal{R}$, that is, $\mathcal{I}_j(\mathcal{B}) \leq \mathcal{I}_j(H)$.

Making use of the definition of infimum, we have:

$$\mathcal{I}_{\mathcal{P}}(\mathcal{B}) = inf\{\mathcal{I}_j(\mathcal{B}) : \mathcal{I}_j \text{ is Herbrand model of } \mathcal{P}\} \leq$$
$$inf\{\mathcal{I}_j(H) : \mathcal{I}_j \text{ is Herbrand model of } \mathcal{P}\} = \mathcal{I}_{\mathcal{P}}(H)$$

Consequently, $\mathcal{I}_{\mathcal{P}}$ satisfies rule $\mathcal{R}$ and (since it analogously satisfies all rules in $\mathcal{P}$) it is a Herbrand model of $\mathcal{P}$, as we expected. Finally, since $\mathcal{I}_{\mathcal{P}} = inf(\mathcal{M})$, using again the definition of infimum, $\mathcal{I}_{\mathcal{P}} \leq \mathcal{I}_j, \forall j$, so $\mathcal{I}_{\mathcal{P}}$ is the least Herbrand model of $\mathcal{P}$, which concludes the proof. □

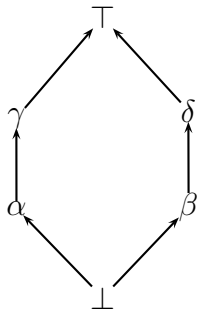
The declarative semantics representing the least Herbrand model $\mathcal{I}_{\mathcal{P}}$ extends the corresponding one of MALP defined in [8] in the kindest way: if L is a multi-adjoint lattice, the previous model $\mathcal{I}_{\mathcal{P}}$ coincides with the one corresponding to this framework. Analogously, $\mathcal{I}_{\mathcal{P}}$ extends the concept of least model of logic programming expressed in [12]; when looking for a complete similarity between both concepts it is possible to use the set-theoretic formulation of $\mathcal{I}_{\mathcal{P}}$ in the Proposition 6.8 below.

In the following example, inspired by the MALP program in [8], we obtain the least Herbrand model of a X-MALP program.

Example 6.7. *Let* $(\&_{\mathtt{G}}, \leftarrow_{\mathtt{G}})$ *be a pair of connectives following the* Gödel*'s intuitionistic logic, that is, the truth functions for* $\&_{\mathtt{G}}$ *and* $\leftarrow_{\mathtt{G}}$ *are:*

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

For the lattice $(L, \leq)$ *with Hasse diagram below, these two connectives do not conform an adjoint pair since, in particular, note that although* $\alpha \dot{\&} \delta \leq \beta$*, it is not fulfilled that* $\alpha \leq \beta \dot{\leftarrow} \delta$*.*



	$\mathcal{I}_1$	$\mathcal{I}_2$	$\mathcal{I}_3$	$\mathcal{I}_4$	$\mathcal{I}_5$	$\mathcal{I}_6$	$\mathcal{I}_7$	$\mathcal{I}_8$	$\mathcal{I}_9$
p	$\bot$	α	β	γ	δ	$\top$	α	γ	$\top$
q	δ	δ	δ	δ	δ	δ	$\top$	$\top$	$\top$

Thus, the program $\mathcal{P} = \{\mathcal{R}_1, \mathcal{R}_2\}$ *with* $\mathcal{R}_1 : \langle p \leftarrow_{\mathtt{G}} q; \alpha \rangle$ *and* $\mathcal{R}_2 : \langle q \leftarrow_{\mathtt{G}} ; \delta \rangle$ *is not valid in multi-adjoint framework since it is supported on a non multi-adjoint lattice. Consider now the* X-MALP *program* $\mathcal{P}' = \{\mathcal{R}'_1, \mathcal{R}'_2\}$*, where* $\mathcal{R}'_1 : p \leftarrow \alpha \&_{\mathtt{G}} q$ *and* $\mathcal{R}'_2 : q \leftarrow \delta$*, which in essence is equivalent to the* MALP *program* $\mathcal{P}$*. The table of the figure summarizes the set* $\mathcal{M} = \{\mathcal{I}_1, \ldots, \mathcal{I}_9\}$ *of all Herbrand models of* $\mathcal{P}'$*. Let us explain in detail how we have obtained such table. Note that, by definition of Herbrand model, each* $\mathcal{I}_j$ *is a Herbrand model of rule* $\mathcal{R}'_2$ *if, and only if,* $\delta \leq \mathcal{I}_j(q)$*, which implies that* $\mathcal{I}_j(q)$ *admits the set of values* $\delta, \top \in L$*. Moreover,* $\mathcal{I}_j$ *is a Herbrand model of* $\mathcal{R}'_1$ *if, and only if,* $\mathcal{I}_j(\alpha \&_{\mathtt{G}} q) \leq \mathcal{I}_j(p)$*, that is,* $\alpha \dot{\&}_{\mathtt{G}} \mathcal{I}_j(q) \leq \mathcal{I}_j(p)$*. Then, once fixed* $\mathcal{I}_j(q)$ *to each one of the previous truth degrees, we add the condition* $\alpha \dot{\&}_{\mathtt{G}} \mathcal{I}_j(q) \leq \mathcal{I}_j(p)$*, which determines the set of valid values for* $\mathcal{I}_j(p)$ *and, consequently, establishes the final set of Herbrand models* $\mathcal{I}_j \in \mathcal{M}$*. Finally, it is easy to see that* $\mathcal{M}$ *has an infimum element (the interpretation* $\mathcal{I}_{\mathcal{P}'}$ *such that,* $\mathcal{I}_{\mathcal{P}'}(p) = \bot$ *and* $\mathcal{I}_{\mathcal{P}'}(q) = \delta$*), so that* $\mathcal{I}_{\mathcal{P}'} \in \mathcal{M}$*, namely,* $\mathcal{I}_{\mathcal{P}'}$ *it is the least Herbrand model of* $\mathcal{P}'$*.*

The previous concepts of Herbrand interpretation and Herbrand model can be expressed in a set-theoretic way, where an interpretation of $\mathcal{P}$, instead of a mapping $\mathcal{I} : B_{\mathcal{P}} \to L$, is conceived as its unique corresponding binary relation $R_{\mathcal{I}} \subset B_{\mathcal{P}} \times L$. In what follows, we try to give an elemental result justifying this fact. Under this point of view, since each Herbrand model of $\mathcal{P}$ can be seen as a set of pairs (subset of $B_{\mathcal{P}} \times L$), the least fuzzy Herbrand model admits a second characterization which enjoys a nice property also reported in [12] for the weaker case of the least Herbrand model of pure logic programs. This new characterization is established in terms of the following theorem, which is immediate to prove and where we consider that $\cap \mathcal{I}_j = \{(A_i; \alpha) : (A_i, \alpha_i) \in \mathcal{I}_j, \forall j, \alpha = \inf\{\alpha_i\} \in L\}$.

Proposición 6.8. *Let* $\mathcal{P}$ *be a* X-MALP *program with associated lattice* $(L, \leq)$*. Let* $\mathcal{I}$ *be a Herbrand interpretation of* $\mathcal{P}$*, that is, a mapping* $\mathcal{I} : \mathcal{B}_{\mathcal{P}} \to L$*. Then,* $\mathcal{I}$ *determines a unique binary relation* $R_{\mathcal{I}} \subset \mathcal{B}_{\mathcal{P}} \times L$*.*

Proof. Note that the mapping $\mathcal{I}$ is determined by its set of images, and the relation $R_{\mathcal{I}}$ is a set of pairs of type $(A, \mathcal{I}(A)), A \in \mathcal{B}_{\mathcal{P}}$. Therefore, mapping $\mathcal{I}$ can be seen as a binary relation, that is, as a certain set of ordered pairs whose first component is a ground formula of the Herbrand base and whose second component is an element of the lattice L. □

By the previous proposition, it is possible to give a Herbrand interpretation $\mathcal{I}$ by means of its formulation based on set theory $R_{\mathcal{I}}$, that we will denote

also by $\mathcal{I}$ whenever it is not necessary to emphasize this one. Then, taking each Herbrand model of $\mathcal{P}$ as a set (a subset of $\mathcal{B}_\mathcal{P} \times L$), that is, $\mathcal{I}_j = \{(A, \alpha) : A \in \mathcal{B}_\mathcal{P}, \alpha \in L\}$, the least fuzzy Herbrand model can be characterized also by the following result.

Theorem 6.9. *The least fuzzy Herbrand model of a program $\mathcal{P}$ is the intersection of all Herbrand models of $\mathcal{P}$, i.e., $\mathcal{J}_\mathcal{P} = \cap \mathcal{I}_j$, where $\mathcal{I}_j$ is a Herbrand model of $\mathcal{P}$ for all j.*

7. X-MALP Fix-point Semantics

In this section, we provide a fix-point characterization of the least fuzzy Herbrand model of a X-MALP logic program, extending the corresponding fix-point operator of [16]. Also, we concrete the fix-points of this operator over some particular examples.

By definition, in a complete lattice $(L, \leq)$, a function $f : L \to L$ is monotone if, and only if, $\forall x, y \in L, x \leq y \Rightarrow f(x) \leq f(y)$. A fix-point in f is an element $x \in L$ such that $f(x) = x$. The basic result for the study of fix-points of functions over lattices is the following theorem of Knaster-Tarski (see [29]). Other theorems of fix-point can be seen in [9, 26].

Theorem 7.1. *Let f be a monotonic function over a complete lattice $(L, \leq)$. Then, f has a fix-point.*

Also, the set of fix-points of f is a complete lattice, so it is possible to consider the least fix-point of f. It is obtained iterating f over $\bot \in L$, that is, it is the supremum of the non decreasing succession $x_0, \ldots, x_i, x_{i+1}, \ldots, x_\lambda, \ldots$ verifying that for each $i \geq 0, x_0 = \bot$, $x_{i+1} = f(x_i)$, while for a certain index λ, $y_\lambda = sup\{y_i : f(y_i) = y_i, i > \lambda\}$.

In [15, 16] the operator $T_\mathcal{P}$, defined by [30] is extended for the multi-adjoint language. The following definition provides a fix-point operator for the X-MALP language that extends the one of the MALP framework and represents the (fix-point) semantics of a X-MALP program $\mathcal{P}$, taken as the least fix-point of $T_\mathcal{P}$. We also justify in Theorem 7.11 the equivalence between this construction and the notion of least fuzzy Herbrand model in Definition 6.5.

Definition 7.2. *Let $\mathcal{P}$ be a* X-MALP *program with associated complete lattice $(L, \leq)$ and $\mathcal{I}$ a Herbrand interpretation. Then, the $T_\mathcal{P}$ operator is a*

mapping in the set of Herbrand interpretations such that, for any ground atom A

$$T_{\mathcal{P}}(\mathcal{I})(A) = sup\{\mathcal{I}(\mathcal{B}\theta) : H \leftarrow_i \mathcal{B} \in \mathcal{P}, A = H\theta\}$$

Because $(L, \leq)$ is complete, $T_{\mathcal{P}}$ is well defined and $T_{\mathcal{P}}(\mathcal{I})(A) \in L$. The following theorems guaranty, for X-MALP programming, some classic results from logic programming for the fix-point operator and that also extend the corresponding ones of the MALP framework.

Particularly, it is easy to prove that the previous operator $T_{\mathcal{P}}$ extends the fix-point operator of [16] in the most satisfactory way: if program $\mathcal{P}$ is MALP our fix-point operator coincides with the one given by Medina et all.

Theorem 7.3. *The operator* $T_{\mathcal{P}}$ *is monotonic.*

Proof. $T_{\mathcal{P}}$ is the morphism $T_{\mathcal{P}} : (\mathcal{H}, \leq) \rightarrow (\mathcal{H}, \leq)$ such that over each $\mathcal{I} \in \mathcal{H}$ determines the mapping in the Herbrand base $T_{\mathcal{P}}(\mathcal{I}) : \mathcal{B}_{\mathcal{P}} \rightarrow \mathcal{B}_{\mathcal{P}}$ defined by $T_{\mathcal{P}}(\mathcal{I})(A) = sup\{\mathcal{I}(\mathcal{B}\theta) : H \leftarrow_i \mathcal{B} \in \mathcal{P}, A = H\theta\}$. Then, given the Herbrand interpretations $\mathcal{I}_1, \mathcal{I}_2 \in \mathcal{H}$, if we suppose that $\mathcal{I}_1 \leq \mathcal{I}_2$, since the supremum preserves the order of the lattice $(L, \leq)$ associated to $\mathcal{P}$, then $T_{\mathcal{P}}(\mathcal{I}_1) \leq T_{\mathcal{P}}(\mathcal{I}_2)$, that is, the required monotonicity of $T_{\mathcal{P}}$.

□

Definition 7.4. *Let* $(L, \leq)$ *be a complete lattice and* $f : L \rightarrow L$ *a mapping.* f *is* continuous *if, and only if, it preserves the supremum of directed sets*[7]*, that is, for each directed set* $X \subset L$, $f(sup(X)) = sup\{f(x) : x \in X\}$ *holds. Also, the map* $f : L^n \rightarrow L$ *is said to be continuous if it is continuous in each variable.*

The next lemma is instrumental; its proof is straightforward: it is enough to proceed by induction in the number of atomic formulas in the body of the rules.

Lema 7.5. *Let* $\mathcal{P}$ *be a* X-MALP *program and* $\mathcal{B}$ *the body of an arbitrary rule of* $\mathcal{P}$*. If all the truth functions of the connectives of* $\mathcal{B}$ *are continuous,* $sup(X)(\mathcal{B}) = sup\{\mathcal{I}(\mathcal{B}) : \mathcal{I} \in X\}$ *holds for all directed set* $X \subset \mathcal{H}$*.*

[7]By definition, $X \subset (L, \leq)$ is a directed set if for all subset $\{x_1, \ldots, x_n\} \subset X$, $sup\{x_1, \ldots, x_n\} \in X$.

The theorem below guaranties the continuity of the fix-point operator under the expected hypothesis. The given sufficient condition is also a necessary condition for achieving such result.

Theorem 7.6. *Let $\mathcal{P}$ be a* X-MALP *program. If all the truth functions of the connectives in the bodies of the rules of $\mathcal{P}$ are continuous, then the $T_{\mathcal{P}}$ operator is continuous.*

Proof. Let X be a directed set of the complete lattice, $(\mathcal{H}, \leq)$, of Herbrand interpretations of $\mathcal{P}$. Let us prove that $T_{\mathcal{P}}(sup(X)) = sup\{T_{\mathcal{P}}(\mathcal{I})$; indeed, for each $A \in \mathcal{B}_{\mathcal{P}}$,

$$\begin{array}{ll} T_{\mathcal{P}}(sup(X))(A) & = sup\{sup(X)(\mathcal{B}) : H \leftarrow_i \mathcal{B} \in \mathcal{P}, A = H\theta\} \\ & = sup\{sup\{\mathcal{I}(\mathcal{B}) : H \leftarrow_i \mathcal{B} \in \mathcal{P}, A = H\theta, \mathcal{I} \in X\}\} \\ & = sup\{T_{\mathcal{P}}(\mathcal{I})(A) : \mathcal{I} \in X\} \end{array}$$

simply by making use of Lemma 7.5. □

Below, we prove the next result for which $T_{\mathcal{P}}$ allows to characterize the interpretations that are Herbrand model of $\mathcal{P}$. Previously, we enunciate the following basic lemma.

Lema 7.7. *Let $(L, \leq)$ be a complete lattice. For any subsets A, B of L it is verified that if $A \subset B$, then $inf(B) \leq inf(A)$.*

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

Note that, thanks to the previous lemma, $\mathcal{I}(A\theta) \geq \mathcal{I}(A)$, for all substitution θ, for all Herbrand interpretation $\mathcal{I}$, whenever the set of ground instances of the formula $A\theta$ is included in the set of ground instances of A.

Theorem 7.8. *A Herbrand interpretation $\mathcal{I}$ is a Herbrand model of a* X-MALP *program $\mathcal{P}$ if, and only if, $T_{\mathcal{P}}(\mathcal{I}) \leq \mathcal{I}$.*

Proof. Let $\mathcal{I}$ be a Herbrand model of $\mathcal{P}$ and let us see that $T_{\mathcal{P}}(\mathcal{I}) \leq \mathcal{I}$. If $H \leftarrow \mathcal{B}$ is a rule of $\mathcal{P}$, by definition of model and by the Lemma 7.7 it follows that $\mathcal{I}(\mathcal{B}\theta) \leq \mathcal{I}(H) \leq \mathcal{I}(H\theta) = \mathcal{I}(A)$, and, then, $T_{\mathcal{P}}(\mathcal{I})(A) \leq \mathcal{I}(A)$ by the definition of supremum, as we expected.

Reciprocally, suppose now that $T_{\mathcal{P}}(\mathcal{I}) \leq \mathcal{I}$ and let us prove that $\mathcal{I}$ is a Herbrand model of $\mathcal{P}$, that is, a Herbrand model of an arbitrary rule

$\mathcal{R} : H \leftarrow \mathcal{B} \in \mathcal{P}$. Fixed this rule, let $A \in \mathcal{B}_\mathcal{P}$ such that $H\theta = A$[8]. Since, by hypothesis, $T_\mathcal{P}(\mathcal{I})(A) = sup\{\mathcal{I}(\mathcal{B}\theta) : H \leftarrow \mathcal{B} \in \mathcal{P}, A = H\theta\} \leq \mathcal{I}(A)$, it follows that $\mathcal{I}(\mathcal{B}) \leq \mathcal{I}(\mathcal{B}\theta) \leq sup\{\mathcal{I}(\mathcal{B}\theta) : H \leftarrow_i \mathcal{B} \in \mathcal{P}, A = H\theta\} \leq \mathcal{I}(A)$, so $\mathcal{I}$ is a Herbrand model of $\mathcal{R}$. □

So, for each Herbrand model $\mathcal{I}$ of $\mathcal{P}$ we have $T_\mathcal{P}(\mathcal{I})(A) = sup\{\mathcal{I}(\mathcal{B}\theta) : H \leftarrow \mathcal{B} \in \mathcal{P}, A = H\theta\} \leq \mathcal{I}(A)$ and, also, the equality $T_\mathcal{P}(\mathcal{I})(A) = \mathcal{I}(A)$ is not reached in general, as it happens in the case of pure logic programming; we can suggest that the more we iterate the operator $T_\mathcal{P}$ over an atom, best is the upper bound obtained for the corresponding correct answer.

On the other hand, in both the pure logic programming and the X-MALP logic programming there can exist more than one fix-point. The next example illustrates this fact for the classical case.

Example 7.9. *Let $\mathcal{P}$ be the logic program with the single clause $p(a) \leftarrow p(a)$. The Herbrand base of $\mathcal{P}$ is the set $\mathcal{B}_\mathcal{P} = \{p(a)\}$ and the subsets of the Herbrand base $\mathcal{I}_1 = \varnothing, \mathcal{I}_2 = \{p(a)\}$ are the only Herbrand models of $\mathcal{P}$. Both are fix-points of the (fix-point) operator defined in [12].*

In the X-MALP context we could take the same example, whenever any definite logic program can be expressed as a X-MALP program. In order to consider a more specific case we provide also the next example.

Example 7.10. *Let $\mathcal{P}$ be a* X-MALP *program consisting in just one rule $p(a) \leftarrow 0.5 \&_\mathtt{G} p(a)$ using the interval $([0,1], \leq)$ as associated lattice, and where the truth function of the connective $\&_\mathtt{G}$ is given by $\dot{\&}_\mathtt{G}(x,y) = inf\{x,y\}$. Then, the interpretations $\mathcal{I}_1, \mathcal{I}_2$ defined by $\mathcal{I}_1(p(a)) = 0, \mathcal{I}_2(p(a)) = 0.5$ verify*

$$T_\mathcal{P}(\mathcal{I}_1)(p(a)) = sup\{0.5\dot{\&}\mathcal{I}_1(p(a))\} = sup\{0.5\&0\} = 0 = \mathcal{I}_1(p(a))$$

$$T_\mathcal{P}(\mathcal{I}_2)(p(a)) = sup\{0.5\dot{\&}\mathcal{I}_2(p(a))\} = sup\{0.5\&0.5\} = 0.5 = \mathcal{I}_2(p(a))$$

and, then, both are fix-point of the operator $T_\mathcal{P}$, resulting $\mathcal{I}_1$ the least fix-point. Also, it is easy to concrete that operator $T_\mathcal{P}$ has infinite fix-points: all the Herbrand interpretations $\mathcal{I}$ defined by $\mathcal{I}(p(a)) = z$, where $z \in [0, 0.5]$.

As we present now, the declarative semantics by least fuzzy Herbrand model is equivalent to the fix-point semantics for X-MALP programming.

[8]The existence of this atom is guaranteed by the definition of the Herbrand universe and Herbrand base of $\mathcal{P}$.

Theorem 7.11. *Given a* X-MALP *program* $\mathcal{P}$*,* $\mathcal{I}_\mathcal{P}$ *is the least fuzzy Herbrand model* $\mathcal{J}_\mathcal{P}$ *of* $\mathcal{P}$ *if, and only if, is the least fix-point of* $T_\mathcal{P}$*.*

Proof. By Theorem 6.6, $\mathcal{I}_\mathcal{P}$ is the least fuzzy Herbrand model of $\mathcal{P}$ if, and only if, $\mathcal{I}_\mathcal{P} = \inf\{\mathcal{I}_j : \mathcal{I}_j \text{ is a Herbrand model of } \mathcal{P}\}$ and by Theorem 7.8, $\mathcal{I}_\mathcal{P} = \inf\{\mathcal{I}_j : T_\mathcal{P}(\mathcal{I}_j) \leq \mathcal{I}_j\}$. Moreover, since $T_\mathcal{P}$ is monotone in the complete lattice L (see Theorem 7.3), there exists the least fix-point of $T_\mathcal{P}$ which coincides with $\inf\{\mathcal{I}_j : T_\mathcal{P}(\mathcal{I}_j) \leq \mathcal{I}_j\}$ (see [12]). Hence, $\mathcal{I}_\mathcal{P}$ is the least fix-point of the $T_\mathcal{P}$ operator. □

This last result states that the declarative semantics of a X-MALPprogram $\mathcal{P}$ can be obtained by transfinitely iterating $T_\mathcal{P}$ from the least interpretation. Note that $T_\mathcal{P}$ may not be continuous, in which case –and differently from the pure logic programming– an uncountable infinite number of iterations may be required to reach the fix-point.

8. Conclusions and Future Work

The high expressive power (and even the sense of its name) of the MALP language, very often relies on the possibility of using multiple adjoint pairs when coding programs. Although we have shown that the adjoint property plays an important role when defining and proving the properties of MALP, it somehow restricts (at least under a theoretical point of view) the class of lattices for being safely used in fuzzy programs.

In this paper we have collected from our previous work in [23], a semantics-preserving transformation which makes use of adjoint pairs in order to produce MALP programs with a very simple shape, which will no longer depend on *adjoint constraints*, thus opening the door for future developments intended to increase the range of fuzzy logic programs beyond MALP.

From here, we have built on top of MALP the so-called "*relaXed Multi-Adjoint Logic Programming*" framework, X-MALP in brief, where the dependence of adjoint pairs is definitively dropped out. This wider class of fuzzy logic programs relying under simple complete (not necessarily multi-adjoint) lattices, has been formally described in a detailed and illustrated way, by showing its syntax as well as its operational and declarative semantics. In this last sense, we have provided a model-theoretic and fix-point characterizations (proving too their equivalences) of fuzzy Herbrand model for X-MALP programs.

For the immediate future, its is mandatory to establish the soundness and completeness properties of the enriched framework for which we plan

to take profit of our previous experiences in the MALP setting regarding some improved formulations of the notion of *reductant* described in [7, 17].

References

[1] J. M. Almendros-Jiménez, A. Luna, and G. Moreno. A Flexible XPath-based Query Language Implemented with Fuzzy Logic Programming. In N. Bassiliades et al., editor, *Proc. of 5th International Symposium on Rules: Research Based, Industry Focused, RuleML'11. Barcelona, July 19-21*, pages 186–193. Lecture Notes in Computer Science 6826. Springer Verlag, 2011.

[2] J.M. Almendros-Jiménez, A. Luna, and G. Moreno. Annotating Fuzzy Chance Degrees when Debugging Xpath Queries. In *Advances in Computational Intelligence. Proc. of the 12th International Work-Conference on Artificial Neural Networks, IWANN'13, Tenerife, Spain, June 12-14*, pages 300–311. Lecture Notes in Computer Science 7903, Part II. Springer Verlag, 2013.

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

[4] J. F. Baldwin, T. P. Martin, and B. W. Pilsworth. *Fril-Fuzzy and Evidential Reasoning in Artificial Intelligence.* John Wiley & Sons, Inc., 1995.

[5] S. Guadarrama, S. Muñoz, and C. Vaucheret. Fuzzy Prolog: A new approach using soft constraints propagation. *Fuzzy Sets and Systems, Elsevier*, 144(1):127–150, 2004.

[6] P. Julián, J. Medina, P. J. Morcillo, G. Moreno, and M. Ojeda-Aciego. An unfolding-based preprocess for reinforcing thresholds in fuzzy tabulation. In *Advances in Computational Intelligence. Proc. of the 12th International Work-Conference on Artificial Neural Networks, IWANN'13, Tenerife, Spain, June 12-14*, pages 647–655. Lecture Notes in Computer Science 7903, Part I. Springer Verlag, 2013.

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

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

[9] M. A. Khamsi and D. Misane. Fixed point theorems in logic programming. *Annals of Mathematics and Artificial Intelligence*, 21:231–243, 1997.

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

[11] R. C. T. Lee. Fuzzy Logic and the Resolution Principle. *Journal of the ACM*, 19(1):119–129, 1972.

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

[13] Y. Loyer and U. Straccia. The well-founded semantics in normal logic programs with uncertainty. In *Proc. of the 6th International Symposium on Functional and Logic Programming*, FLOPS'02, pages 152–166, London, UK, 2002. Springer-Verlag.

[14] Y. Loyer and U. Straccia. Approximate well-founded semantics, query answering and generalized normal logic programs over lattices. *Annals of Mathematics and Artificial Intelligence*, 55(3-4):389–417, 2009.

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

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

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

[18] 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, *Proc. of 4th International Symposium on Rule Interchange and Applications, RuleML'10. Washington, USA, October 21-23*, pages 119–126. Lecture Notes in Computer Science 6403. Springer Verlag, 2010.

[19] P. J. Morcillo, G. Moreno, J. Penabad, and C. Vázquez. Dedekind-MacNeille completion and cartesian product of multi-adjoint lattices. *International Journal of Computer Mathematics*, 89(13-14):1742–1752, 2012.

[20] G. Moreno, P. J. Morcillo, J. Penabad, and C. Vázquez. String-based multi-adjoint lattices for tracing fuzzy logic computations. *Electronic Communications of the European Association of Software Science and Technology (EASST)*, 55:1–17, 2012.

[21] G. Moreno, J. Penabad, and C. Vázquez. On fuzzy correct answers and logical consequences in multi-adjoint logic programming. In J. Vigo-Aguiar, editor, *Proc. of 12th International Conference on Mathematical Methods in Science and Engineering, CMMSE'12. La Manga, Spain, July 2-5*, volume III, pages 864–875, 2012.

[22] G. Moreno, J. Penabad, and C. Vázquez. SSE: Similarity-based strict equality for multi-adjoint logic programs. In J. Vigo-Aguiar, editor, *Proc. of 12th International Conference on Mathematical Methods in Science and Engineering, CMMSE'12. La Manga, Spain, July 2-5*, volume III, pages 876–887, 2012.

[23] G. Moreno, J. Penabad, and C. Vázquez. Relaxing the role of adjoint pairs in multi-adjoint logic programming. In I. Hamilton and J. Vigo-Aguiar, editors, *Proc. of 13th International Conference on Mathematical Methods in Science and Engineering, CMMSE'13. Cabo de Gata, Almería, Spain, June 24-27*, volume III, pages 1156–1167, 2013.

[24] S. Muñoz, V. P. Ceruelo, and H. Strass. Rfuzzy: Syntax, semantics and implementation details of a simple and expressive fuzzy tool over prolog. *Information Sciences*, 181(10):1951–1970, 2011.

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

[26] A. Stouti. A fuzzy version of tarski's fixpoint theorem. *Archivum Mathematicum*, 40(3):273–279, 2004.

[27] U. Straccia. Managing uncertainty and vagueness in description logics, logic programs and description logic programs. In *Reasoning Web, 4th International Summer School, Tutorial Lectures*, Lecture Notes in Computer Science 5224, pages 54–103. Springer Verlag, 2008.

[28] 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.

[29] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[30] M. H. van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM*, 23(4):733–742, 1976.

[31] P. Vojtáš. Fuzzy Logic Programming. *Fuzzy Sets and Systems, Elsevier*, 124(1):361–370, 2001.

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