Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Notes in Theoretical Computer Science
ELSEVIER Electronic Notes in Theoretical Computer Science 137 (2005) 69–103
www.elsevier.com/locate/entcs

Unfolding-based Improvements on Fuzzy Logic Programs[1]

Pascual Julián[2]

Dep. of Computer Science, ESI, Univ. of Castilla–La Mancha
Paseo de la Universidad, 4; 13071 Ciudad Real, Spain

Ginés Moreno[2]

Dep. of Computer Science, EPSA, Univ. of Castilla–La Mancha
Campus Universitario, s/n; 02071 Albacete, Spain

Jaime Penabad[2]

Dep. of Mathematics, EPSA, Univ. of Castilla–La Mancha
Campus Universitario, s/n; 02071 Albacete, Spain

Abstract

Unfolding is a semantics-preserving program transformation technique that consists in the expansion of subexpressions of a program using their own definitions. In this paper we define two unfolding-based transformation rules that extend the classical definition of the unfolding rule (for pure logic programs) to a fuzzy logic setting. We use a fuzzy variant of Prolog where each program clause can be interpreted under a different (fuzzy) logic. We adapt the concept of a *computation rule*, a mapping that selects the subexpression of a goal involved in a computation step, and we prove the independence of the computation rule. We also define a basic transformation system and we demonstrate its strong correctness, that is, original and transformed programs compute the same fuzzy computed answers. Finally, we prove that our transformation rules always produce an improvement in the efficiency of the residual program, by reducing the length of successful Fuzzy SLD-derivations.

Keywords: Fuzzy Logic Programming, Program Transformation.

[1] This work is partially supported by MCYT under grant TIN 2004-07943-C04-03
[2] Emails: {Pascual.Julian, Gines.Moreno, Jaime.Penabad}@uclm.es

doi:10.1016/j.entcs.2005.01.040

1 Introduction

Logic Programming [12] has been widely used for problem solving and knowledge representation in the past. Nevertheless, traditional logic programming languages do not incorporate techniques or constructs in order to treat explicitly uncertainty and approximated reasoning. *Fuzzy Logic* provides a mathematical background for modeling uncertainty and/or vagueness. Fuzzy logic relays on the concept of fuzzy set, the theory of fuzzy connectives (t-norms, t-conorms, etc.) and the extension of two-values classical predicate logic to a logic where formulas can be evaluated in the range of the $[0, 1]$ real interval (see [22] or [13] for a comprehensive introduction of this subject). Fuzzy sets [23] are objects introduced to deal with the fuzziness or vagueness we find in the real world when we try to describe phenomena that have not sharply defined boundaries. Given a set U, an ordinary subset A of U can be defined in terms of its *characteristic function* $\chi_A(x)$ (that returns 1 if $x \in A$ or 0 otherwise) which neatly specifies whether or not an element x is in A. On the other hand, a *fuzzy subset* A of U is a function $A : U \to [0, 1]$. The function A is called the *membership function*, and the value $A(x)$ represents the *degree of membership* (it is not meant to convey the likelihood that x has some particular attribute such as "young" [13]) of x in the fuzzy set A. Different functions A can be considered for a fuzzy concept and, in general, they will present a soft shape instead of the characteristic function's crisp slope of an ordinary set.

Fuzzy Logic Programming is an interesting and still growing research area that agglutinates the efforts to introduce Fuzzy Logic into Logic Programming. During the last decades, several fuzzy logic programming systems have been developed, where the classical inference mechanism of SLD-Resolution is replaced with a fuzzy variant which is able to handle partial truth and to reason with uncertainty. Most of these systems implement the fuzzy resolution principle introduced by Lee in [10], such as the Prolog-Elf system [4], Fril Prolog system [2] and the F-Prolog language [11].

On the other hand, there is also no agreement about which fuzzy logic must be used when fuzzifying Prolog. Most systems use min-max logic (for modeling the conjunction and disjunction operations) but other systems just use Lukasiewicz logic [7]. Other approaches are parametric with respect the interpretation of the fuzzy connectives, letting them unspecified to obtain a more general framework [21]. Recently, it has been appeared in [20] a theoretical model for fuzzy logic programming which deals with many values implications. Finally, in [19] we find an extremely flexible scheme where, apart from introducing negation and dealing with interval-valued fuzzy sets [8], each clause on a given program may be interpreted with a different logic. In this paper, we

follow this last extension in a very natural way.

The fold/unfold transformation approach (also known as "rules+strategies" approach [15]) was first introduced in [3] to optimize functional programs and then used for logic programs [17] and integrated functional-logic programs [1]. This approach is based on the construction, by means of a *strategy*, of a sequence of equivalent programs —called *transformation sequence* and usually denoted by $\mathcal{P}_0, \ldots, \mathcal{P}_n$— where each program $\mathcal{P}_i$ is obtained from the preceding ones $\mathcal{P}_0, \ldots, \mathcal{P}_{i-1}$ by using an *elementary* transformation rule. The essential rules are *folding* and *unfolding*, i.e., contraction and expansion of subexpressions of a program using the definitions of this program (or of a preceding one). Other rules which have been considered are, e.g., definition introduction/elimination, and algebraic replacement.

Focusing in unfolding, the objective of this program transformation operation is to replace a program rule by the set of rules obtained after applying a *symbolic computation* step (in all its possible forms) on the body of the selected rule [15]. Depending on the concrete paradigm taken into account (functional [3], logic [17] or integrated functional–logic [1]) the considered computation step will be performed using —some variant of— its associated operational mechanism (rewriting, resolution or narrowing, respectively).

In this paper we study the extension of two unfolding-based transformations to a fuzzy context. In particular, besides defining a fuzzy variant of the unfolding transformation of [17], we also introduce a new transformation, called *T-Norm replacement*, operating on the fuzzy component of an expression, which has some similarities with the algebraic replacement of [3]. As we will see, the adaptation of the unfolding rule from classical LP to fuzzy LP can not be made in a naive way, if we really want to define it in a natural and correct way. In particular we need to take into account the "functional" dimension associated to the fuzzy component and, what it is more important, it is crucial to introduce an intermediate language (not required neither in LP, nor in FP) for coding the programs involved during the transformation process.

Apart from [5] and as far as we now, this is the first approach described in the literature for introducing fuzzy transformation rules. The present work largely improves [5] in the following points: i) Language: We use a richer extension of the fuzzy logic language described in [21] which directly affects all the subsequent definitions, results and proofs. ii) Transformations: Now we present a clearer transformation scheme by providing two unfolding-based transformation rules, each one focused in a language component: whereas the fuzzy unfolding rule concentrates into the logic part, the T-Norm replacement rule handles the fuzzy component. Moreover, the effects obtained by the

(fourth variant of the) T-Norm replacement operation can not be achieved by the single transformation rule introduced in [5]. iii) Properties: Besides correctness results, we proof the reduction in length of successful Fuzzy SLD-derivations (when they are computed in the transformed program).

The outline of this paper is as follows. In the next section, we summarize an extension of the fuzzy Prolog dialect described in [21]. In the new extension, that we call f-Prolog, each program clause can be interpreted under a different logic. Section 3 presents the operational semantics of our language. Moreover, in Section 4, we also adapt the concept of a *computation rule* and we prove a result which is the fuzzy counterpart of the independence of the computation rule theorem demonstrated in [12]. In Section 5 we define two unfolding-based transformation rules for a labeled mark variant of f-Prolog whereas in Section 6 we prove its main theoretical/practical properties. Finally, we show our conclusions in Section 7.

2 Fuzzy Prolog Programs

Among the variety of fuzzy logic programming languages in the literature, the one described in [21] is specially appropriated to define the concept of unfolding of fuzzy logic programs. In this section we present an extension of this language, that we call f-Prolog (*fuzzy* Prolog) and allow us to give a more flexible interpretation of the logical connectives in the style introduced by [19].

Let $\mathcal{L}$ be a first order language containing variables, function symbols, predicate symbols, constants, quantifiers $\forall$ and $\exists$, and connectives $\neg$, seq, et_1, and et_2 (the intended meaning is that seq is an implication —the left-arrow version is written as qes—, et_1 is a conjunction evaluating *modus ponens* with seq, and et_2 is a conjunction typically occurring in the body of clauses). Although et_1 and et_2 are binary connectives, we usually generalize them as functions with an arbitrary number of arguments. That is we write, for instance, $\mathrm{et}_2(x_1, \ldots, x_n)$ instead of $\mathrm{et}_2(x_1, \mathrm{et}_2(x_2, \ldots, \mathrm{et}_2(x_{n-1}, x_n) \ldots))$.

A (definite) *clause* is a formula $\forall(A \text{ qes } \mathrm{et}_2(B_1, \ldots, B_n))$, and a (definite) *goal* is a formula $\forall(\text{qes } \mathrm{et}_2(B_1, \ldots, B_n))$, where A and each B_i are atomic formulas. In general, we call them $(\mathrm{seq}, \mathrm{et}_2)$- *formulas* or simply *formulas* if the kind of connectives used in their writing is not important or can be inferred by the context. Also, we write $A \leftarrow B_1, \ldots, B_n$ as syntactic sugar of $\forall(A \text{ qes } \mathrm{et}_2(B_1, \ldots, B_n))$, etc. As usually, A is said to be the head of the clause and $(B_1, \ldots, B_n)$ the body. Clauses with an empty body are called *facts*, whereas clauses with a head and a body are called *rules*. A sort of degenerate clause is the *empty clause*, denoted by '$\square$', representing a contradictory formula.

The meaning function for the connective seq, denoted by $[\![\text{seq}]\!]$, is the residual implication defined as: $[\![\text{seq}]\!](x, y) = sup\ \{z \in [0, 1] : [\![\text{et}_1]\!](x, y) \leq y\}$. Therefore $[\![\text{seq}]\!]$ is a R-implication, since it is only defined beginning from a t-norm, but it is not necessarily a S-implication, neither a QM-implication [18]. Nevertheless, the meaning function for the connective seq we use in this paper is the one proposed by [21], where $[\![\text{seq}]\!]$ is linked to $[\![\text{et}_1]\!]$ in such a way that modus ponens is a sound rule, what is essential in order to proof the correctness of the Fuzzy SLD-Resolution Calculus.

Note also that, in the discussion above, the meaning functions for connectives et_1 and et_2 were let unspecified as arbitrary t-norms [3] $[\![\text{et}_i]\!] : [0, 1]^2 \to [0, 1]$, properly extended as many valued functions, and they are intended to be fixed in the range of the whole program. However, as it has been told in [19], it may be useful from a practical point of view to associate a concrete interpretation for each operator et_1 or et_2 in the context of a program clause instead of a fixed interpretation for the whole program environment. In particular, note that, the conjunction is defined by Zadeh via the *min* operator but it is widely accepted that no single operator for conjunction can model the wide variety of expressions that is necessary to formalize.

Example 2.1 Given a clause $p(x) \longleftarrow q(x), r(x)$ if we interpret $p(x)$ as "x is sportsman", $q(x)$ as "x is young" and $r(x)$ as "x is healthy", the predicates of the body are of positive influence (that is, they mutually reinforce themselves). In this case the conjunction et_2 of body is usually understood as *min*. Similarly, if $q(x)$, and $r(x)$ are interpreted as independent or noninteractive predicates. However, if $q(x)$ is interpreted as "x is low" and $r(x)$ as "x is tall", that is, the predicates $q(x)$ and $r(x)$ has a contradictory meaning, the conjunction et_2 of body is advisable to be interpreted as *max*.

We can redefine the concept of fuzzy theory that appears in [21] in order to cope with this problem.

Definition 2.2 A *fuzzy theory* is a partial mapping T applying a triple $\langle \alpha, le_1, le_2 \rangle$ in $(0, 1] \times Sem \times Sem$, to each formula, where $(0, 1]$ is the domain of *truth degrees* and Sem is a set of semantics labels indicating the associated meaning for et_1 and et_2 respectively.

The real α is a *truth degree* expressing the confidence which the user of the system has in the truth of the clause $\mathcal{C}$. A truth degree $\alpha = 1$ means that the user believes the clause $\mathcal{C}$ is true; on the other hand, a truth degree less

[3] They are commutative, associative, and monotone in both arguments and $[\![\text{et}_i]\!](x, 1) = x$ (hence, they subsume classical conjunction $\{0, 1\}^2 \to \{0, 1\}$) [16]. Note also that, in general, meaning functions for et_i connectives are not distributive.

than 1 represents the degree of uncertainty or lost of confidence on the truth of a belief; a truth degree near 0 expresses the lack of confidence on the truth of a belief. We can use labels indicating the meaning assigned to the et_1 or et_2 operator in the clause $\mathcal{C}$. For instance, a label $le_i =$ `lukasiewicz` interprets an operator et_i as a Lukasiewicz t-norm, that is, $[\![\mathrm{et}_i]\!](x, y) = max(0, x + y - 1)$. Other possible labels would be: `min`, if $[\![\mathrm{et}_i]\!](x, y) = min(x, y)$; `product`, if $[\![\mathrm{et}_i]\!](x, y) = x \cdot y$; etc. A `void` value in Sem is employed to express that no meaning for et_1 or et_2 is selected. Note that, operationally, since `void` must be linked to a t-norm, $void(1, x) = void(x, 1) = x$.

Definition 2.3 A *definite* f-Prolog *program*, $\mathcal{P}$, is a fuzzy theory such that:

(i) $\mathcal{D}om(\mathcal{P})$ is a set of $(\mathrm{seq}, \mathrm{et}_2)$-definite program clauses or facts,

(ii) for $\mathcal{C}_1, \mathcal{C}_2 \in \mathcal{D}om(\mathcal{P})$, we say $\mathcal{C}_1 \approx \mathcal{C}_2$ if and only if $\mathcal{C}_1$ is a variant of $\mathcal{C}_2$ and $\mathcal{P}(\mathcal{C}_1) = \mathcal{P}(\mathcal{C}_2)$, and

(iii) $\mathcal{D}om(\mathcal{P})/_{\approx}$ is finite.

Roughly speaking, a program can be seen as a set of pairs $(\mathcal{C}; \langle r, l1, l2\rangle)$, where $\mathcal{C}$ is a clause, r is the truth degree of the clause $\mathcal{C}$, and $l1$, $l2$ are the semantics labels associated with the operators et_1 and et_1 for this clause. However we prefer to write $\mathcal{C}$ `with` $\langle r, l1, l2\rangle$, or more descriptively: $\mathcal{C}$ `with` $\alpha = r$ `and` $le_1 = l1$ `and` $le_2 = l2$. If clause $\mathcal{C}$ is a fact, le_1 and le_2 are `void` and we simply write: $\mathcal{C}$ `with` $\alpha = r$; omitting the values for le_1 and le_2. Similarly, a goal $\mathcal{G}$ has only associated a semantic label for et_2, but no initial truth degree or semantic label for et_1, and we write: $\mathcal{G}$ `with` $le_2 = l2$.

3 Operational Semantics and Labeled Fuzzy Prolog

Given a goal $\mathcal{G}$ its truth degree, α, is obtained by evaluating a sequence of Fuzzy SLD-Resolution steps leading to an empty clause. In the sequel we formalize the concepts of Fuzzy SLD-Resolution, Fuzzy SLD-Derivation and Fuzzy computer answer, with some variations with regard to the definitions that appear in [21].

Let $\mathcal{P}$ be a program and $\mathcal{G}$ a goal. Since $\mathcal{D}om(\mathcal{P})/_{\approx}$ is a classical definite program, the classical SLD-resolution should still work. Therefore, the main operational problem is to define the machinery for evaluating truth degrees. The truth degree of a expression is a semantic notion that must be evaluated using meaning functions. Considering a program rule $\mathcal{C} \equiv A \leftarrow B_1, \ldots, B_m$, with $\alpha = q$, and a goal $\mathcal{G} \equiv \leftarrow A'$, where A' unifies with the head A of $\mathcal{C}$, it is possible a SLD-resolution step leading to the resolvent $\mathcal{G}' \equiv \leftarrow (B_1, \ldots, B_m)$. If we want to evaluate the truth degree of $\mathcal{G}$, we have to compute the truth degrees $r_1, \ldots, r_n$ of all subgoals $B_1, \ldots, B_m$ before the truth degree q of the

rule can be applied to obtain $[\![\mathrm{et}_1]\!](q, [\![\mathrm{et}_2]\!](r_1, \ldots, r_n))$, the truth degree of the goal $\mathcal{G}$. We need a mechanism in order to remenber that a program rule was applied in former steps, since it is necessary to distinguish when to apply $[\![\mathrm{et}_1]\!]$ or $[\![\mathrm{et}_2]\!]$. In [21] a context grammar was introduced to solve this problem. This grammar contains left and right marks ($\boxed{\mathrm{L}_q}$ and $\boxed{\mathrm{R}_q}$) labeled by a real value, to remember the exact point where a rule with $\alpha = q$ was applied. Hence the previous resolution step can be annotated as: $\boxed{\mathrm{L}_q}\, B_1, \ldots, B_m \,\boxed{\mathrm{R}_q}$. For our case, in order to manage the resolution process properly while extending the expressive power of our language, it is also necessary to expand the label mechanism to distinguish what is the meaning operator $[\![\mathrm{et}_1]\!]$ or $[\![\mathrm{et}_2]\!]$ that must be applied. Hence, we introduce the marks $\boxed{\mathrm{L}_{\langle q, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle}}$ and $\boxed{\mathrm{R}_{\langle q, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle}}$. We call lf-Prolog the extended language obtained by adding labeled marks and real numbers to the f-Prolog alphabet. A lf-expression is an atom, a sequence of real numbers, or a real number enclosed between labeled marks. The following definition makes use of lf-Prolog in the formalization of Fuzzy SLD-Resolution (we write $\overline{o}$ for the -possibly empty- sequence of syntactic objects $o_1, \ldots, o_n$).

Definition 3.1 Let $\mathcal{G} \equiv \leftarrow \mathcal{Q}$ with $le_2 = [\![\mathrm{et}_2]\!]$ be a lf-Prolog goal and let ϑ be a substitution, a lf-Prolog *state* is a pair $\langle \mathcal{Q}; \vartheta \rangle$. Let $\mathcal{E}$ be the set of lf-Prolog states. Given a f-Prolog program $\mathcal{P}$, we define *Fuzzy SLD-Resolution* as a state transition system, whose transition relation $\rightarrow_{FR} \subseteq (\mathcal{E} \times \mathcal{E})$ is the smallest relation satisfying the following rules:

Rule 1. **(Clause Resolution Rule)**

$\langle \overline{X}, A_m, \overline{Y}; \vartheta \rangle \rightarrow_{FR} \langle (\overline{X}, \boxed{\mathrm{L}_{\langle q, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle}} B_1, \ldots, B_l \boxed{\mathrm{R}_{\langle q, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle}}, \overline{Y})\theta; \vartheta\theta \rangle$ if

(i) A_m is the selected atom,
(ii) θ is an mgu of A_m and A,
(iii) $\mathcal{C} \equiv (A \leftarrow B_1, \ldots, B_l \; with \; \langle q, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle) \in \mathcal{P}$ and $l \geq 1$.

Rule 2. **(Fact Resolution Rule)**

$\langle \overline{X}, A_m, \overline{Y}; \vartheta \rangle \rightarrow_{FR} \langle (\overline{X}, r, \overline{Y})\theta; \vartheta\theta \rangle$ if

(i) A_m is the selected atom,
(ii) θ is an mgu of A_m and A, and
(iii) $\mathcal{C} \equiv (A \leftarrow \; with \; r) \in \mathcal{P}$.

Rule 3. **($[\![\mathrm{et}_1]\!]$ Resolution Rule)** [4]

[4] In [21], the $[\![\mathrm{et}_1]\!]$ resolution rule is expressed as a combination of our third and fourth rules but our reformulation is completely equivalent to the original when the labels le_1 and le_2 are fixed in the whole program.

$\langle \overline{X}, \boxed{\mathrm{L}_{\langle q,[\![\mathrm{et}_1]\!],[\![\mathrm{et}_2]\!]\rangle}}\ r\ \boxed{\mathrm{R}_{\langle q,[\![\mathrm{et}_1]\!],[\![\mathrm{et}_2]\!]\rangle}}, \overline{Y}; \vartheta\rangle \rightarrow_{FR} \langle \overline{X}, [\![\mathrm{et}_1]\!](q,r), \overline{Y}; \vartheta\rangle$ if

(i) r is a real number.

Rule 4. ($[\![\mathrm{et}_2]\!]$ **Resolution Rule)**

$\langle \overline{X}, \boxed{\mathrm{L}_{\langle q,[\![\mathrm{et}_1]\!],[\![\mathrm{et}_2]\!]\rangle}}\ \overline{r}\ \boxed{\mathrm{R}_{\langle q,[\![\mathrm{et}_1]\!],[\![\mathrm{et}_2]\!]\rangle}}, \overline{Y}; \vartheta\rangle \rightarrow_{FR}$

$\langle \overline{X}, \boxed{\mathrm{L}_{\langle q,[\![\mathrm{et}_1]\!],[\![\mathrm{et}_2]\!]\rangle}}\ [\![\mathrm{et}_2]\!](\overline{r})\ \boxed{\mathrm{R}_{\langle q,[\![\mathrm{et}_1]\!],[\![\mathrm{et}_2]\!]\rangle}}, \overline{Y}; \vartheta\rangle$ if

(i) $\overline{r} \equiv r_1, \ldots, r_n$, where $n > 1$, are real numbers.

All familiar logic programming concepts can be extended for the fuzzy case, assuming also that clauses involved in fuzzy SLD-computation steps are renamed before being used. In the following, symbols $\rightarrow_{FR1}$, $\rightarrow_{FR2}$, $\rightarrow_{FR3}$ and $\rightarrow_{FR4}$ may be used for explicitly referring to the application of each one of the four fuzzy resolution rules. When needed, the exact lf-expression and/or clause used in the corresponding step, will be also annotated as a super-index of the $\rightarrow_{FR}$ symbol. In order to extend the notion of computer answer in our fuzzy setting, in the following definition we use *id* to refer to the empty substitution, $\mathcal{V}ar(s)$ denotes the set of distinct variables occurring in the syntactic object s, and $\theta[\mathcal{V}ar(s)]$ corresponds to the substitution obtained from θ by restricting its domain, $Dom(\theta)$, to $\mathcal{V}ar(s)$.

Definition 3.2 Let $\mathcal{P}$ be a f-Prolog program and $\mathcal{G} \equiv \leftarrow \mathcal{Q}$ with $le_2 = [\![\mathrm{et}_2]\!]$ be a lf-Prolog goal. A pair $\langle r; \theta\rangle$ consisting of a real number r and a substitution θ is a *fuzzy computed answer* (f.c.a.) if there is a sequence $\mathcal{E}_0, \ldots, \mathcal{E}_n$ (called f-derivation) such that:

1. $\mathcal{E}_0 = \langle \boxed{\mathrm{L}_{\langle 1,void,[\![\mathrm{et}_2]\!]\rangle}}, \mathcal{Q}, \boxed{\mathrm{R}_{\langle 1,void,[\![\mathrm{et}_2]\!]\rangle}}; id\rangle$,
2. for each $0 \leq i < n$, $\mathcal{G}_i \rightarrow_{FR} \mathcal{G}_{i+1}$ is a fuzzy SLD-resolution step,
3. $\mathcal{E}_n = \langle r; \theta'\rangle$ and $\theta = \theta'[\mathcal{V}ar(\mathcal{Q})]$.

We illustrate the last definition by means of the following example.

Example 3.3 Let $\mathcal{P}$ be the f-Prolog program,

$\mathcal{C}_1$:	$p(X) \leftarrow q(X,Y), r(Y)$	with $\langle 0.8, \mathtt{prod}, \mathtt{min}\rangle$
$\mathcal{C}_2$:	$q(a,Y) \leftarrow s(Y)$	with $\langle 0.7, \mathtt{prod}, \mathtt{void}\rangle$
$\mathcal{C}_3$:	$q(Y,a) \leftarrow r(Y)$	with $\langle 0.8, \mathtt{luka}, \mathtt{void}\rangle$
$\mathcal{C}_4$:	$r(Y) \leftarrow$	with 0.7
$\mathcal{C}_5$:	$s(b) \leftarrow$	with 0.9

The following is a successful f-derivation for the program $\mathcal{P}$ and the goal

"$\leftarrow p(X), r(a)$ with `min`" with f.c.a. $\langle 0.504; \{X/a\}\rangle$:

$$\langle \boxed{\mathrm{L}_{\Psi_0}}\ \underline{p(X)}, r(a)\ \boxed{\mathrm{R}_{\Psi_0}}\ ; \sigma_0\rangle \qquad \rightarrow_{FR1}{}^{\mathcal{C}_1}$$

$$\langle \boxed{\mathrm{L}_{\Psi_0}}\ \boxed{\mathrm{L}_{\Psi_1}}\ \underline{q(X_1, Y_1)}, r(Y_1)\ \boxed{\mathrm{R}_{\Psi_1}}\ r(a)\ \boxed{\mathrm{R}_{\Psi_0}}\ ; \sigma_1\rangle \qquad \rightarrow_{FR1}{}^{\mathcal{C}_2}$$

$$\langle \boxed{\mathrm{L}_{\Psi_0}}\ \boxed{\mathrm{L}_{\Psi_1}}\ \boxed{\mathrm{L}_{\Psi_2}}\ \underline{s(Y_2)}\ \boxed{\mathrm{R}_{\Psi_2}}\ r(Y_2)\ \boxed{\mathrm{R}_{\Psi_1}}\ r(a)\ \boxed{\mathrm{R}_{\Psi_0}}\ ; \sigma_2\rangle \qquad \rightarrow_{FR2}{}^{\mathcal{C}_5}$$

$$\langle \boxed{\mathrm{L}_{\Psi_0}}\ \boxed{\mathrm{L}_{\Psi_1}}\ \boxed{\mathrm{L}_{\Psi_2}}\ 0.9\ \boxed{\mathrm{R}_{\Psi_2}}\ \underline{r(b)}\ \boxed{\mathrm{R}_{\Psi_1}}\ r(a)\ \boxed{\mathrm{R}_{\Psi_0}}\ ; \sigma_3\rangle \qquad \rightarrow_{FR2}{}^{\mathcal{C}_4}$$

$$\langle \boxed{\mathrm{L}_{\Psi_0}}\ \boxed{\mathrm{L}_{\Psi_1}}\ \boxed{\mathrm{L}_{\Psi_2}}\ 0.9\ \boxed{\mathrm{R}_{\Psi_2}}\ 0.7\ \boxed{\mathrm{R}_{\Psi_1}}\ \underline{r(a)}\ \boxed{\mathrm{R}_{\Psi_0}}\ ; \sigma_4\rangle \qquad \rightarrow_{FR2}{}^{\mathcal{C}_4}$$

$$\langle \boxed{\mathrm{L}_{\Psi_0}}\ \boxed{\mathrm{L}_{\Psi_1}}\ \underline{\boxed{\mathrm{L}_{\Psi_2}}\ 0.9\ \boxed{\mathrm{R}_{\Psi_2}}}\ 0.7\ \boxed{\mathrm{R}_{\Psi_1}}\ 0.7\ \boxed{\mathrm{R}_{\Psi_0}}\ ; \sigma_5\rangle \qquad \rightarrow_{FR3}$$

`//since product(0.9,0.7)=0.63`

$$\langle \boxed{\mathrm{L}_{\Psi_0}}\ \underline{\boxed{\mathrm{L}_{\Psi_1}}\ 0.63, 0.7\ \boxed{\mathrm{R}_{\Psi_1}}}\ 0.7\ \boxed{\mathrm{R}_{\Psi_0}}\ ; \sigma_5\rangle \qquad \rightarrow_{FR4}$$

`//since min(0.63,0.7)=0.63`

$$\langle \boxed{\mathrm{L}_{\Psi_0}}\ \underline{\boxed{\mathrm{L}_{\Psi_1}}\ 0.63\ \boxed{\mathrm{R}_{\Psi_1}}}\ 0.7\ \boxed{\mathrm{R}_{\Psi_0}}\ ; \sigma_5\rangle \qquad \rightarrow_{FR3}$$

`//since product(0.63,0.8)=0.504`

$$\langle \underline{\boxed{\mathrm{L}_{\Psi_0}}\ 0.504, 0.7\ \boxed{\mathrm{R}_{\Psi_0}}}\ ; \sigma_5\rangle \qquad \rightarrow_{FR4}$$

`//since min(0.504,0.7)=0.504`

$$\langle \underline{\boxed{\mathrm{L}_{\Psi_0}}\ 0.504\ \boxed{\mathrm{R}_{\Psi_0}}}\ ; \sigma_5\rangle \qquad \rightarrow_{FR3}$$

`// since void(1,0.504)=0.504`

$$\langle\ 0.504\ ; \sigma_5\rangle$$

where $\Psi_0 \equiv \langle 1, \mathtt{void}, \mathtt{min}\rangle$, $\Psi_1 \equiv \langle 0.8, \mathtt{prod}, \mathtt{min}\rangle$ and $\Psi_2 \equiv \langle 0.7, \mathtt{prod}, \mathtt{max}\rangle$, are the triples associated with the original goal and clauses $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively. Also the substitutions $\sigma_0 = id$, $\sigma_1 = \{X/X_1\}$, $\sigma_2 = \{X/a, X_1/a, Y_1/Y_2\}$, $\sigma_3 = \{X/a, X_1/a, Y_1/b, Y_2/b\}$, $\sigma_4 = \{X/a, X_1/a, Y_1/b, Y_2/b, Y_3/b\}$ and $\sigma_5 = \{X/a, X_1/a, Y_1/b, Y_2/b, Y_3/b, Y_4/a\}$.

In [21], the authors established the correctness results for the Fuzzy SLD-Resolution operational mechanism (following a technique similar as the one proposed by Lloyd, in [12], for classical logic programming), but extending all results with the treatment of truth degrees. These correctness results can be

easily adapted to our case.

As for the classical SLD-Resolution calculus, we assume the existence of a fixed selection function, also called *fuzzy computation rule*, deciding, for a given goal, which is the selected lf-expression to be exploited in the next fuzzy SLD-Resolution step. For instance, when building the f-derivation shown in Example 3.3, we have used a computation rule similar to the left to right selection rule of Prolog but delaying the application of the $[\![\mathrm{et}_1]\!]$ and $[\![\mathrm{et}_2]\!]$ resolution rules until all atoms have been resolved. Given a fuzzy computation rule $\mathcal{R}$, we say that a fuzzy SLD-derivation is *via* $\mathcal{R}$ if the selected lf-expression in every step is obtained by the application of the mapping $\mathcal{R}$ to the corresponding goal in that step. In the following section, we stablish in our fuzzy setting the independence of the computation rule proved in [12] for the pure logic programming case.

4 Independence of the Fuzzy Computation Rule

Before starting with the core of the paper, we introduce some technical notations and concepts that will help us to develop our proofs. In the following we use $\mathcal{C} \ll \mathcal{P}$ to denote an *standardised apart* new variant of a clause in a lf-Prolog program $\mathcal{P}$ such that $\mathcal{C}$ contains no variable which was previously met during a computation. The equational representation of a substitution $\theta = \{x_1/t_1, \ldots, x_n/t_n\}$ is the set of equations $\widehat{\theta} = \{x_1 = t_1, \ldots, x_n = t_n\}$. Let $mgu(E)$ denote the *most general unifier* of an equation set E (see [9] for a formal definition of this concept). *Parallel composition* of substitutions [14] corresponds to the notion of unification generalized to substitutions. Given two idempotent substitutions θ_1 and θ_2, the parallel composition $\theta_1 \Uparrow \theta_2 = mgu(\widehat{\theta_1} \cup \widehat{\theta_2})$. The following propperty will be usefull later.

Lemma 4.1 *[14] Let θ_1 and θ_2 be idempotent substitutions,*

$$\theta_1 \Uparrow \theta_2 = \theta_1 mgu(\widehat{\theta_2}\theta_1) = \theta_2 mgu(\widehat{\theta_1}\theta_2).$$

In order to prove the independence of the fuzzy computation rule, we need the following auxiliary Lemmas. The first one focuses on the preservation of substitutions in f.c.a.'s obtained on two-steps fuzzy SLD-derivations exploiting two different atoms of a given goal, with independence of the order in which they are exploited.

Lemma 4.2 *Let $\mathcal{P}$ be a* lf-Prolog *program and let $\mathcal{G} \equiv \leftarrow \mathcal{Q}$ `with` $le_2 = [\![\mathrm{et}_2]\!]^0$ be a* lf-Prolog *goal, such that A and A' are different atoms in $\mathcal{Q}$ and $\Psi_0 \equiv \langle 1, void, [\![et_2]\!]^0 \rangle$. Then, $\langle \boxed{\mathrm{L}_{\Psi_0}} \mathcal{Q} \boxed{\mathrm{R}_{\Psi_0}} ; \theta_0 \rangle \rightarrow_{FR}{}^{A} \langle \boxed{\mathrm{L}_{\Psi_0}} \mathcal{Q}_1 \boxed{\mathrm{R}_{\Psi_0}} ; \theta_0\theta_1 \rangle$*

$\rightarrow_{FR}{}^{A'\theta_1}\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}_2\boxed{\mathrm{R}_{\Psi_0}}\ ;\theta_0\theta_1\theta_2\rangle$ *iff* $\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}\boxed{\mathrm{R}_{\Psi_0}}\ ;\theta_0\rangle \rightarrow_{FR}{}^{A'}$
$\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}'_1\boxed{\mathrm{R}_{\Psi_0}}\ ;\theta_0\theta'_1\rangle \rightarrow_{FR}{}^{A\theta'_1}\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}'_2\boxed{\mathrm{R}_{\Psi_0}}\ ;\theta_0\theta'_1\theta'_2\rangle$, *where* $\theta_0\theta_1\theta_2 = \theta_0\theta'_1\theta'_2$.

Proof.
($\Rightarrow$) Let atoms H_1 and H_2 be the heads of the two clauses $\mathcal{C}_1, \mathcal{C}_2 \ll \mathcal{P}$ used to exploit (instances of) atoms A and A', respectively, in the considered f-derivation: $\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}\boxed{\mathrm{R}_{\Psi_0}}\ ;\theta_0\rangle \rightarrow_{FR}{}^{A}\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}_1\boxed{\mathrm{R}_{\Psi_0}}\ ;\theta_0\theta_1\rangle \rightarrow_{FR}{}^{A'\theta_1}$ $\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}_2\boxed{\mathrm{R}_{\Psi_0}}\ ;\theta_0\theta_1\theta_2\rangle$, where $\theta_1 = mgu(\{A = H_1\})$ and $\theta_2 = mgu(\{A'\theta_1 = H_2\})$. Moreover, since $\theta_0\theta_1\theta_2 \neq fail$, in particular $\theta_2 \neq fail$, and hence $\theta'_1 = mgu(\{A' = H_2\}) \neq fail$. Now, the following equalities hold:

$$
\begin{array}{lll}
\theta_1\theta_2 & = & \\
\theta_1 mgu(\{A'\theta_1 = H_2\}) & = & (\text{since } Dom(\theta_1) \cap \mathcal{V}ar(\mathcal{C}_2) = \emptyset) \\
\theta_1 mgu(\widehat{mgu}(\{A' = H_2\})\theta_1) & = & \\
\theta_1 mgu(\widehat{\theta'_1}\theta_1) & = & (\text{by Lemma 4.1}) \\
\theta_1 \Uparrow \theta'_1 & = & (\text{by Lemma 4.1}) \\
\theta'_1 mgu(\widehat{\theta_1}\theta'_1) & = & \\
\theta'_1 mgu(\widehat{mgu}(\{A = H_1\})\theta'_1) & = & (\text{since } Dom(\theta'_1) \cap \mathcal{V}ar(\mathcal{C}_1) = \emptyset) \\
\theta'_1 mgu(\{A\theta'_1 = H_1\}) & &
\end{array}
$$

Moreover, since $\theta_1\theta_2 \neq fail$ then $\theta'_1 mgu(\{A\theta'_1 = H_1\}) \neq fail$ and, in particular, $\theta'_2 = mgu(\{A\theta'_1 = H_1\}) \neq fail$. Hence, $\theta_1\theta_2 = \theta'_1\theta'_2$ which implies that $\theta_0\theta_1\theta_2 = \theta_0\theta'_1\theta'_2$, as we wanted to prove.
($\Leftarrow$) This case can be easily proved in a similar way as the previous one, by also exploiting the equivalence between $\theta_1\theta_2$ and $\theta'_1\theta'_2$. □

The following Lemma generalizes the previous one at two different levels:

- by implying the preservation of both elements in the derivation states i.e., not only substitution, but also partially evaluated truth degrees, and
- by considering the whole set of resolution rules of Definition 3.1 (instead of the first pair uniquely) when applying the two resolution steps on the considered derivation.

Lemma 4.3 (Switching Lemma) *Let $\mathcal{P}$ be a* If-Prolog *program and let $\mathcal{G} \equiv \leftarrow\mathcal{Q}$ `with` $le_2 = [\![\mathrm{et}_2]\!]^0$ be a* If-Prolog *goal, such that E and E' are different lf-expressions in $\mathcal{Q}$ and $\Psi_0 \equiv \langle 1, void, [\![et_2]\!]^0\rangle$. Then,*

$\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}\boxed{\mathrm{R}_{\Psi_0}};\theta_0\rangle \rightarrow_{FR}{}^{E} \langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}_1\boxed{\mathrm{R}_{\Psi_0}};\theta_1\rangle \rightarrow_{FR}{}^{E'\theta_1} \langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}_2\boxed{\mathrm{R}_{\Psi_0}};\theta_2\rangle$
iff
$\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}\boxed{\mathrm{R}_{\Psi_0}};\theta_0\rangle \rightarrow_{FR}{}^{E'} \langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}'_1\boxed{\mathrm{R}_{\Psi_0}};\theta'_1\rangle \rightarrow_{FR}{}^{E\theta'_1} \langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}'_2\boxed{\mathrm{R}_{\Psi_0}};\theta'_2\rangle,$
where $\mathcal{Q}_2 = \mathcal{Q}'_2$ *and* $\theta_2 = \theta'_2$.

Proof. In our proof, we assume that $\mathcal{Q} \equiv \overline{X}, E_1, \overline{Y}, E_2, \overline{Z}$ where $\overline{X}, \overline{Y}$ and $\overline{Z}$ are arbitrary sequences of valid lf-expressions (such that $\overline{X}$ is headed with $\boxed{\mathrm{L}_{\Psi_0}}$ and $\overline{Z}$ ends with $\boxed{\mathrm{R}_{\Psi_0}}$)and E_1 and E_2, i.e., the lf-expressions selected in the first and the second f-derivation steps, will be atoms (denoted by A or A'), sequences of real numbers (like $\overline{r}$ or $\overline{n}$) or real numbers enclosed between labeled marks (for instance, $\boxed{\mathrm{L}_{\Psi}}\,r\,\boxed{\mathrm{R}_{\Psi}}$ or $\boxed{\mathrm{L}_{\Psi}}\,n\,\boxed{\mathrm{R}_{\Psi}}$, with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$, $\Psi' \equiv \langle q, [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle$), depending on the concrete resolution rule of Definition 3.1 used on any step. For readability reasons, we underline the selected lf-expression exploited in each derivation step. Now, we exhaustively proceed with each one of all the possible cases. Fortunately, note that it is not relevant if E_1 is to the left or to the right of E_2 and, moreover, the case where the first step is done with rule i and the second one with rule j is perfectly analogous to the case where first step is done with rule j and the second one with rule i, which drastically reduces the number of alternatives.

(i) First step with Rule 1 and second step with Rule 1.
Assume that: $\mathcal{C}_1 \equiv (H_1 \leftarrow \overline{B_1}; \Psi) \ll \mathcal{P}$ and $\mathcal{C}_2 \equiv (H_2 \leftarrow \overline{B_2}; \Psi') \ll \mathcal{P}$, with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$ and $\Psi' \equiv \langle q, [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle$. Then,

$$\langle \overline{X}, \underline{A}, \overline{Y}, A', \overline{Z}; \theta_0\rangle \rightarrow_{FR1}{}^{\mathcal{C}_1}$$
$$\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\,\overline{B_1}\,\boxed{\mathrm{R}_{\Psi}}, \overline{Y}, \underline{A'}, \overline{Z})\theta_1; \theta_1\rangle \rightarrow_{FR1}{}^{\mathcal{C}_2}$$
$$\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\,\overline{B_1}\,\boxed{\mathrm{R}_{\Psi}}, \overline{Y}, \boxed{\mathrm{L}_{\Psi'}}\,\overline{B_2}\,\boxed{\mathrm{R}_{\Psi'}}, \overline{Z})\theta_2; \theta_2\rangle$$

iff

$$\langle \overline{X}, A, \overline{Y}, \underline{A'}, \overline{Z}; \theta_0\rangle \rightarrow_{FR1}{}^{\mathcal{C}_2}$$
$$\langle (\overline{X}, \underline{A}, \overline{Y}, \boxed{\mathrm{L}_{\Psi'}}\,\overline{B_2}\,\boxed{\mathrm{R}_{\Psi'}}, \overline{Z})\theta'_1; \theta'_1\rangle \rightarrow_{FR1}{}^{\mathcal{C}_1}$$
$$\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\,\overline{B_1}\,\boxed{\mathrm{R}_{\Psi}}, \overline{Y}, \boxed{\mathrm{L}_{\Psi'}}\,\overline{B_2}\,\boxed{\mathrm{R}_{\Psi'}}, \overline{Z})\theta'_2; \theta'_2\rangle$$

where, by Lemma 4.2 we have that $\theta_2 = \theta'_2$, which also implies that the first element of the final states in both derivations are syntactically identical, as we wanted to prove.

(ii) First step with Rule 1 and second step with Rule 2.
Assume that: $\mathcal{C}_1 \equiv (H_1 \leftarrow \overline{B_1}; \Psi) \ll \mathcal{P}$ and $\mathcal{C}_2 \equiv (H_2 \leftarrow; q) \ll \mathcal{P}$, with

$\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$. Then,

$$\langle \overline{X}, \underline{A}, \overline{Y}, A', \overline{Z}; \theta_0\rangle \rightarrow_{FR1}{}^{\mathcal{C}_1}$$
$$\langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{B_1}\boxed{\mathrm{R}_\Psi}, \overline{Y}, \underline{A'}, \overline{Z})\theta_1; \theta_1\rangle \rightarrow_{FR2}{}^{\mathcal{C}_2}$$
$$\langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{B_1}\boxed{\mathrm{R}_\Psi}, \overline{Y}, q, \overline{Z})\theta_2; \theta_2\rangle$$

iff

$$\langle \overline{X}, A, \overline{Y}, \underline{A'}, \overline{Z}; \theta_0\rangle \rightarrow_{FR2}{}^{\mathcal{C}_2}$$
$$\langle (\overline{X}, \underline{A}, \overline{Y}, q, \overline{Z})\theta_1'; \theta_1'\rangle \rightarrow_{FR1}{}^{\mathcal{C}_1}$$
$$\langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{B_1}\boxed{\mathrm{R}_\Psi}, \overline{Y}, q, \overline{Z})\theta_2'; \theta_2'\rangle$$

where, by Lemma 4.2 we have that $\theta_2 = \theta_2'$, which also implies that the first element of the final states in both derivations are syntactically identical, as we wanted to prove.

(iii) First step with Rule 1 and second step with Rule 3.
Assume that: $\mathcal{C}_1 \equiv (H_1 \leftarrow \overline{B_1}; \Psi) \ll \mathcal{P}$ with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$, and $\Psi' \equiv \langle q, [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle$. Then,

$$\langle \overline{X}, \underline{A}, \overline{Y}, \boxed{\mathrm{L}_{\Psi'}}\ n\boxed{\mathrm{R}_{\Psi'}}, \overline{Z}; \theta_0\rangle \rightarrow_{FR1}{}^{\mathcal{C}_1}$$
$$\langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{B_1}\boxed{\mathrm{R}_\Psi}, \overline{Y}, \underline{\boxed{\mathrm{L}_{\Psi'}}\ n\boxed{\mathrm{R}_{\Psi'}}}, \overline{Z})\theta_1; \theta_1\rangle \rightarrow_{FR3}$$
$$\langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{B_1}\boxed{\mathrm{R}_\Psi}, \overline{Y}, [\![\mathrm{et}_1]\!]'(q, n), \overline{Z})\theta_1; \theta_1\rangle$$

iff

$$\langle \overline{X}, A, \overline{Y}, \underline{\boxed{\mathrm{L}_{\Psi'}}\ n\boxed{\mathrm{R}_{\Psi'}}}, \overline{Z}; \theta_0\rangle \rightarrow_{FR3}$$
$$\langle \overline{X}, \underline{A}, \overline{Y}, [\![\mathrm{et}_1]\!]'(q, n), \overline{Z}; \theta_0\rangle \rightarrow_{FR1}{}^{\mathcal{C}_1}$$
$$\langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{B_1}\boxed{\mathrm{R}_\Psi}, \overline{Y}, [\![\mathrm{et}_1]\!]'(q, n), \overline{Z})\theta_1; \theta_1\rangle$$

as we wanted to prove.

(iv) First step with Rule 1 and second step with Rule 4.
Assume that: $\mathcal{C}_1 \equiv (H_1 \leftarrow \overline{B_1}; \Psi) \ll \mathcal{P}$ with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$, and $\Psi' \equiv \langle q, [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle$. Then,

$$\langle \overline{X}, \underline{A}, \overline{Y}, \boxed{\mathrm{L}_{\Psi'}}\ \overline{r}\boxed{\mathrm{R}_{\Psi'}}, \overline{Z}; \theta_0\rangle \rightarrow_{FR1}{}^{\mathcal{C}_1}$$
$$\langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{B_1}\boxed{\mathrm{R}_\Psi}, \overline{Y}, \underline{\boxed{\mathrm{L}_{\Psi'}}\ \overline{r}\boxed{\mathrm{R}_{\Psi'}}}, \overline{Z})\theta_1; \theta_1\rangle \rightarrow_{FR4}$$
$$\langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{B_1}\boxed{\mathrm{R}_\Psi}, \overline{Y}, [\![\mathrm{et}_2]\!]'(\overline{r}), \overline{Z})\theta_1; \theta_1\rangle$$

iff

$$\langle \overline{X}, A, \overline{Y}, \underline{\boxed{\mathrm{L}_{\Psi'}}\ \overline{r}\ \boxed{\mathrm{R}_{\Psi'}}}, \overline{Z}; \theta_0 \rangle \qquad \rightarrow_{FR4}$$

$$\langle \overline{X}, \underline{A}, \overline{Y}, [\![\mathrm{et}_2]\!]'(\overline{r}), \overline{Z}; \theta_0 \rangle \qquad \rightarrow_{FR1}{}^{\mathcal{C}_1}$$

$$\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{B_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y}, [\![\mathrm{et}_2]\!]'(\overline{r}), \overline{Z})\theta_1; \theta_1 \rangle$$

as we wanted to prove.

(v) First step with Rule 2 and second step with Rule 2.
Assume that: $\mathcal{C}_1 \equiv (H_1 \leftarrow; p) \ll \mathcal{P}$ and $\mathcal{C}_2 \equiv (H_2 \leftarrow; q) \ll \mathcal{P}$. Then,

$$\langle \overline{X}, \underline{A}, \overline{Y}, A', \overline{Z}; \theta_0 \rangle \qquad \rightarrow_{FR2}{}^{\mathcal{C}_1}$$

$$\langle (\overline{X}, p, \overline{Y}, \underline{A'}, \overline{Z})\theta_1; \theta_1 \rangle \rightarrow_{FR2}{}^{\mathcal{C}_2}$$

$$\langle (\overline{X}, p, \overline{Y}, q, \overline{Z})\theta_2; \theta_2 \rangle$$

iff

$$\langle \overline{X}, A, \overline{Y}, \underline{A'}, \overline{Z}; \theta_0 \rangle \qquad \rightarrow_{FR2}{}^{\mathcal{C}_2}$$

$$\langle (\overline{X}, \underline{A}, \overline{Y}, q, \overline{Z})\theta_1'; \theta_1' \rangle \rightarrow_{FR2}{}^{\mathcal{C}_1}$$

$$\langle (\overline{X}, p, \overline{Y}, q, \overline{Z})\theta_2'; \theta_2' \rangle$$

where, by Lemma 4.2 we have that $\theta_2 = \theta_2'$, which also implies that the first element of the final states in both derivations are syntactically identical, as we wanted to prove.

(vi) First step with Rule 2 and second step with Rule 3.
Assume that: $\mathcal{C}_1 \equiv (H_1 \leftarrow; q) \ll \mathcal{P}$ and $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!] \rangle$. Then,

$$\langle \overline{X}, \underline{A}, \overline{Y}, \boxed{\mathrm{L}_{\Psi}}\ n\ \boxed{\mathrm{R}_{\Psi}}, \overline{Z}; \theta_0 \rangle \qquad \rightarrow_{FR2}{}^{\mathcal{C}_1}$$

$$\langle (\overline{X}, q, \overline{Y}, \underline{\boxed{\mathrm{L}_{\Psi}}\ n\ \boxed{\mathrm{R}_{\Psi}}}, \overline{Z})\theta_1; \theta_1 \rangle \rightarrow_{FR3}$$

$$\langle (\overline{X}, q, \overline{Y}, [\![\mathrm{et}_1]\!](p, n), \overline{Z})\theta_1; \theta_1 \rangle$$

iff

$$\langle \overline{X}, A, \overline{Y}, \underline{\boxed{\mathrm{L}_{\Psi}}\ n\ \boxed{\mathrm{R}_{\Psi}}}, \overline{Z}; \theta_0 \rangle \rightarrow_{FR3}$$

$$\langle \overline{X}, \underline{A}, \overline{Y}, [\![\mathrm{et}_1]\!](p, n), \overline{Z}; \theta_0 \rangle \qquad \rightarrow_{FR2}{}^{\mathcal{C}_1}$$

$$\langle (\overline{X}, q, \overline{Y}, [\![\mathrm{et}_1]\!](p, n), \overline{Z})\theta_1; \theta_1 \rangle$$

as we wanted to prove.

(vii) First step with Rule 2 and second step with Rule 4.

Assume that: $\mathcal{C}_1 \equiv (H_1 \leftarrow \overline{B_1}; q) \ll \mathcal{P}$ and $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$. Then,

$$\langle \overline{X}, \underline{A}, \overline{Y}, \boxed{\mathrm{L}_\Psi}\ \overline{r}\ \boxed{\mathrm{R}_\Psi}, \overline{Z}; \theta_0\rangle \quad \rightarrow_{FR2}{}^{\mathcal{C}_1}$$
$$\langle (\overline{X}, q, \overline{Y}, \underline{\boxed{\mathrm{L}_\Psi}\ \overline{r}\ \boxed{\mathrm{R}_\Psi}}, \overline{Z})\theta_1; \theta_1\rangle \rightarrow_{FR4}$$
$$\langle (\overline{X}, q, \overline{Y}, [\![\mathrm{et}_2]\!](\overline{r}), \overline{Z})\theta_1; \theta_1\rangle$$

iff

$$\langle \overline{X}, A, \overline{Y}, \underline{\boxed{\mathrm{L}_\Psi}\ \overline{r}\ \boxed{\mathrm{R}_\Psi}}, \overline{Z}; \theta_0\rangle \rightarrow_{FR4}$$
$$\langle \overline{X}, \underline{A}, \overline{Y}, [\![\mathrm{et}_2]\!](\overline{r}), \overline{Z}; \theta_0\rangle \quad \rightarrow_{FR2}{}^{\mathcal{C}_1}$$
$$\langle (\overline{X}, q, \overline{Y}, [\![\mathrm{et}_2]\!](\overline{r}), \overline{Z})\theta_1; \theta_1\rangle$$

as we wanted to prove.

(viii) First step with Rule 3 and second step with Rule 3.
Assume that $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$ and $\Psi' \equiv \langle q, [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle$. Then,

$$\langle \overline{X}, \underline{\boxed{\mathrm{L}_\Psi}\ r\ \boxed{\mathrm{R}_\Psi}}, \overline{Y}, \boxed{\mathrm{L}_{\Psi'}}\ n\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Z}; \theta_0\rangle \rightarrow_{FR3}$$
$$\langle \overline{X}, [\![\mathrm{et}_1]\!](p, r), \overline{Y}, \underline{\boxed{\mathrm{L}_{\Psi'}}\ n\ \boxed{\mathrm{R}_{\Psi'}}}, \overline{Z}; \theta_0\rangle \quad \rightarrow_{FR3}$$
$$\langle \overline{X}, [\![\mathrm{et}_1]\!](p, r), \overline{Y}, [\![\mathrm{et}_1]\!]'(q, n), \overline{Z}; \theta_0\rangle$$

iff

$$\langle \overline{X}, \boxed{\mathrm{L}_\Psi}\ r\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y}, \underline{\boxed{\mathrm{L}_{\Psi'}}\ n\ \boxed{\mathrm{R}_{\Psi'}}}, \overline{Z}; \theta_0\rangle \rightarrow_{FR3}$$
$$\langle \overline{X}, \underline{\boxed{\mathrm{L}_\Psi}\ r\ \boxed{\mathrm{R}_{\Psi'}}}, \overline{Y}, [\![\mathrm{et}_1]\!]'(q, n), \overline{Z}; \theta_0\rangle \quad \rightarrow_{FR3}$$
$$\langle \overline{X}, [\![\mathrm{et}_1]\!](p, r), \overline{Y}, [\![\mathrm{et}_1]\!]'(q, n), \overline{Z}; \theta_0\rangle$$

as we wanted to prove.

(ix) First step with Rule 3 and second step with Rule 4.
Assume that $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$ and $\Psi' \equiv \langle q, [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle$. Then,

$$\langle \overline{X}, \underline{\boxed{\mathrm{L}_\Psi}\ r\ \boxed{\mathrm{R}_\Psi}}, \overline{Y}, \boxed{\mathrm{L}_{\Psi'}}\ \overline{n}\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Z}; \theta_0\rangle \rightarrow_{FR3}$$
$$\langle \overline{X}, [\![\mathrm{et}_1]\!](p, r), \overline{Y}, \underline{\boxed{\mathrm{L}_{\Psi'}}\ \overline{n}\ \boxed{\mathrm{R}_{\Psi'}}}, \overline{Z}; \theta_0\rangle \quad \rightarrow_{FR4}$$
$$\langle \overline{X}, [\![\mathrm{et}_1]\!](p, r), \overline{Y}, [\![\mathrm{et}_2]\!]'(\overline{n}), \overline{Z}; \theta_0\rangle$$

iff

$$\langle \overline{X}, \boxed{\mathrm{L}_{\Psi}}\ r\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y}, \underline{\boxed{\mathrm{L}_{\Psi'}}\ \overline{n}\ \boxed{\mathrm{R}_{\Psi'}}}, \overline{Z}; \theta_0 \rangle \rightarrow_{FR4}$$

$$\langle \overline{X}, \underline{\boxed{\mathrm{L}_{\Psi}}\ r\ \boxed{\mathrm{R}_{\Psi}}}, \overline{Y}, [\![\mathrm{et}_2]\!]'(\overline{n}), \overline{Z}; \theta_0 \rangle \rightarrow_{FR3}$$

$$\langle \overline{X}, [\![\mathrm{et}_1]\!](p, r), \overline{Y}, [\![\mathrm{et}_2]\!]'(\overline{n}), \overline{Z}; \theta_0 \rangle$$

as we wanted to prove.

(x) First step with Rule 4 and second step with Rule 4.
Assume that $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!] \rangle$ and $\Psi' \equiv \langle q, [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]' \rangle$. Then,

$$\langle \overline{X}, \underline{\boxed{\mathrm{L}_{\Psi}}\ \overline{r}\ \boxed{\mathrm{R}_{\Psi}}}, \overline{Y}, \boxed{\mathrm{L}_{\Psi'}}\ \overline{n}\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Z}; \theta_0 \rangle \rightarrow_{FR4}$$

$$\langle \overline{X}, [\![\mathrm{et}_2]\!](\overline{r}), \overline{Y}, \underline{\boxed{\mathrm{L}_{\Psi'}}\ \overline{n}\ \boxed{\mathrm{R}_{\Psi'}}}, \overline{Z}; \theta_0 \rangle \rightarrow_{FR4}$$

$$\langle \overline{X}, [\![\mathrm{et}_2]\!](\overline{r}), \overline{Y}, [\![\mathrm{et}_2]\!]'(\overline{n}), \overline{Z}; \theta_0 \rangle$$

iff

$$\langle \overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{r}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y}, \underline{\boxed{\mathrm{L}_{\Psi'}}\ \overline{n}\ \boxed{\mathrm{R}_{\Psi'}}}, \overline{Z}; \theta_0 \rangle \rightarrow_{FR4}$$

$$\langle \overline{X}, \underline{\boxed{\mathrm{L}_{\Psi}}\ \overline{r}\ \boxed{\mathrm{R}_{\Psi}}}, \overline{Y}, [\![\mathrm{et}_2]\!]'(\overline{n}), \overline{Z}; \theta_0 \rangle \rightarrow_{FR4}$$

$$\langle \overline{X}, [\![\mathrm{et}_2]\!](\overline{r}), \overline{Y}, [\![\mathrm{et}_2]\!]'(\overline{n}), \overline{Z}; \theta_0 \rangle$$

as we wanted to prove.

□

Finally, we can formalize and prove the main result of this section.

Theorem 4.4 (Independence of the Fuzzy Computation Rule) *Let $\mathcal{P}$ be a* If-Prolog *program, and $\mathcal{G} \equiv \leftarrow \mathcal{Q}$ with $le_2 = [\![\mathrm{et}_2]\!]$ be a* If-Prolog *goal. Then, for any pair of different fuzzy computation rules $\mathcal{R}$ and $\mathcal{R}'$, we have that:*

$$\langle \boxed{\mathrm{L}_{\Psi}}\ \mathcal{Q}\ \boxed{\mathrm{R}_{\Psi}}; id \rangle \rightarrow_{FR^n_{\mathcal{R}}} \langle r; \theta \rangle \quad \textit{iff} \quad \langle \boxed{\mathrm{L}_{\Psi}}\ \mathcal{Q}\ \boxed{\mathrm{R}_{\Psi}}; id \rangle \rightarrow_{FR^n_{\mathcal{R}'}} \langle r; \theta \rangle$$

where $\Psi \equiv \langle 1, void, [\![\mathrm{et}_2]\!] \rangle$ and n is the (same) number of fuzzy SLD-resolution steps in both derivations.

Proof. Immediate by repeatedly applying the Switching Lemma 4.3. □

5 Unfolding-based Transformations for Fuzzy Programs

In essence, the unfolding transformation traditionally considered in pure logic programming consists in the replacement of a program clause C by the set of clauses obtained after applying a symbolic computation step in all its possible forms on the body of C [15]. In [5] we gave a first approach to the fuzzy

extension of unfolding, by considering the complete set of fuzzy SLD-resolution rules in Definition 3.1 (when performing symbolic computation steps) in order to generate all alternative clauses. However, a deeper look at Definition 3.1, reveals us that only rules 1 and 2 reproduce the essence of classical logic programming by exploiting atoms and generating unifiers and, in this sense, they are more appropriate to be used during the unfolding process to simulate the original definition. On the other hand, rules 3 and 4 neither reduce atoms nor produce unifiers, but simply perform numerical manipulations to produce truth degrees (what, in some way, reflects the fuzzy component of this enriched context). Therefore, rules 3 and 4 should be more appropriately used for defining other kind of transformations (as we will see in Definition 5.3).

In this paper we adopt this new point of view and in the next two sections, we define a set of program transformations based on (fuzzy variants of) the classical unfolding operation for pure logic programs defined in [17]. We also prove their strong correctness, i.e., they are sound and complete w.r.t. the semantics of fuzzy computed answers obtained by fuzzy SLD-resolution.

5.1 The Fuzzy Unfolding Transformation Rule

As we have seen in the previous sections, the differences between f-Prolog and lf-Prolog programs appear only at the syntactic level: whereas the body B of a (non unit) f-Prolog program clause (which in essence, is no more than a simple goal, that is, an atom or a conjunction of atoms) respects the grammar $B \rightarrow B, \ldots, B \mid atom$, we need to enrich this set of grammar rules with $B \rightarrow \boxed{\mathrm{L}_\Psi}\ B\ \boxed{\mathrm{R}_\Psi} \mid number$, if we really want to cope with the possibility of including marks and real numbers in the body of lf-Prolog clauses (which intuitively have the same structure of any initial, intermediate or final goal appearing in fuzzy SLD-derivations). This implies that any f-Prolog program is also a lf-Prolog program, although the contrary is not always true (i.e., the set of f-Prolog programs is a proper subclass of the set of lf-Prolog programs). Apart from this simple fact (which, on the other hand, is mandatory to define the fuzzy SLD-resolution principle) both languages share the same operational semantics.

On the other hand, note that the application of rules 1 and 2 always generates clauses whose bodies include marks or numbers which implies that an unfolding transformation based on these rules is able to preserve the syntactic structure of lf-Prolog programs but, even in the case that original programs be also f-Prolog programs, the transformed ones will never belong to this subclass: the marks or real numbers incorporated into the transformed clauses by unfolding steps force the lost of the original f-Prolog syntax. In order to

avoid this inconvenience, our transformation rules focus on the general framework of lf-Prolog programs instead of the more restricted subclass of f-Prolog programs. Note also that this fact does not suppose a problem in practice: the most important purpose of transformations rules, apart from preserving the program semantics, is to optimize code, independently of the object language. Classical fold/unfold based transformation systems optimize programs by returning code which uses the same source language, but unfolding has also played important roles in the design of compilers (see [6]) which generate an object code written in a target language. In this sense, our transformation system can be seen as a mixed technique that optimizes f-Prolog programs and compiles it into lf-Prolog programs, with the advantage in our case that both programs are executable with exactly the same operational principle.

In the following, we consider a fixed transformation sequence $(\mathcal{P}_0, \ldots, \mathcal{P}_k)$, $k \geq 0$, where we only require that the initial program $\mathcal{P}_0$ be an f-Prolog program since the remaining ones will necessarily adopt the lf-Prolog syntax.

Definition 5.1 [Fuzzy Unfolding]
Let $(\mathcal{P}_0, \ldots, \mathcal{P}_k)$ be a transformation sequence of lf-Prolog programs starting from an f-Prolog program $\mathcal{P}_0$. Assume that $\mathcal{C} \equiv (H \leftarrow \overline{B}$ with $\Psi) \in \mathcal{P}_k$ is a (non unit) lf-Prolog program clause, such that $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!] \rangle$ and A is an atom included in the body $\overline{B}$. Then, the next program $\mathcal{P}_{k+1}$ in the transformation sequence can be obtained by fuzzy unfolding of clause $\mathcal{C}$ at atom A in program $\mathcal{P}_k$ as follows: $\mathcal{P}_{k+1} = (\mathcal{P}_k - \{\mathcal{C}\}) \cup \{H\sigma \leftarrow \overline{B'}$ with $\Psi \mid \langle \boxed{\mathrm{L}_\Psi}\, \overline{B}\, \boxed{\mathrm{R}_\Psi}; id\rangle \rightarrow_{FR}{}^{A} \langle \boxed{\mathrm{L}_\Psi}\, \overline{B'}\, \boxed{\mathrm{R}_\Psi}; \sigma\rangle\}$.

There are some remarks to do regarding our definition. Similarly to the classical SLD-resolution based unfolding rule presented in [17], the substitutions computed by resolution steps during unfolding are incorporated to the transformed rules in a natural way, i.e., by applying them to the head of the clause. On the other hand, the propagation of truth degrees during the fuzzy unfolding process is done at two different levels: by directly assigning the complete tuple Ψ (containing the truth degree p and labels for $[\![\mathrm{et}_1]\!]$ and $[\![\mathrm{et}_2]\!]$) of the original clause to the transformed one, and by introducing marks and/or real numbers in its body. The following example illustrates these facts.

Example 5.2 Consider again the set of clauses of Example 3.3 as the initial program, $\mathcal{P}_0$, of a transformation sequence. It is easy to see that the unfolding of clause $\mathcal{C}_2$ w.r.t. $\mathcal{P}_0$ (exploiting the second rule of Definition 3.1) generates the new program $\mathcal{P}_1 = (\mathcal{P}_0 - \{\mathcal{C}_2\}) \cup \{\mathcal{C}_6\}$, where $\mathcal{C}_6$ is the new unfolded rule "$q(a,b) \leftarrow 0.9$ with $\langle 0.7, \texttt{product}, \texttt{void}\rangle$".

On the other hand, if we want to unfold now clause $\mathcal{C}_1$ in program $\mathcal{P}_1$, we

must firstly generate the following one-step Fuzzy SLD-derivations [5] :

$$\langle \boxed{\mathrm{L}_{\Psi_1}}\ \underline{q(X,Y)}, r(Y)\ \boxed{\mathrm{R}_{\Psi_1}}; id\rangle \rightarrow_{FR1}{}^{\mathcal{C}_6} \langle \boxed{\mathrm{L}_{\Psi_1}}\ \boxed{\mathrm{L}_{\Psi_6}}\ 0.9\ \boxed{\mathrm{R}_{\Psi_6}}, r(b)\ \boxed{\mathrm{R}_{\Psi_1}}; \sigma_1\rangle$$

$$\langle \boxed{\mathrm{L}_{\Psi_1}}\ \underline{q(X,Y)}, r(Y)\ \boxed{\mathrm{R}_{\Psi_1}}; id\rangle \rightarrow_{FR1}{}^{\mathcal{C}_3} \langle \boxed{\mathrm{L}_{\Psi_1}}\ \boxed{\mathrm{L}_{\Psi_3}}\ r(Y_1)\ \boxed{\mathrm{R}_{\Psi_3}}, r(a)\ \boxed{\mathrm{R}_{\Psi_1}}; \sigma_2\rangle$$

where $\Psi_1 \equiv \langle 0.8, \texttt{prod}, \texttt{min}\rangle$, $\Psi_6 \equiv \langle 0.7, \texttt{prod}, \texttt{max}\rangle$ and $\Psi_3 \equiv \langle 0.8, \texttt{luka}, \texttt{void}\rangle$, are the triples corresponding to clauses $\mathcal{C}_1$, $\mathcal{C}_6$ and $\mathcal{C}_3$ respectively; also the unifiers $\sigma_1 = \{X/a, Y/b\}$ and $\sigma_2 = \{X/Y_1, Y/a\}$.

So, the unfolded program $\mathcal{P}_2 = (\mathcal{P}_1 - \{\mathcal{C}_1\}) \cup \mathcal{U}$ where $\mathcal{U}$ contains the new clauses:

$$\mathcal{C}_7 : p(a) \leftarrow \boxed{\mathrm{L}_{\langle 0.7,\texttt{prod},\texttt{max}\rangle}}\ 0.9\ \boxed{\mathrm{R}_{\langle 0.7,\texttt{prod},\texttt{max}\rangle}}, r(b) \quad \text{with } \langle 0.8, \texttt{prod}, \texttt{min}\rangle$$

$$\mathcal{C}_8 : p(Y_1) \leftarrow \boxed{\mathrm{L}_{\langle 0.8,\texttt{luka},\texttt{min}\rangle}}\ r(Y_1)\ \boxed{\mathrm{R}_{\langle 0.8,\texttt{luka},\texttt{min}\rangle}}, r(a) \text{ with } \langle 0.8, \texttt{prod}, \texttt{min}\rangle$$

Finally, by performing a new resolution step with the second rule of Definition 3.1 on atom $r(b)$ in the body of clause $\mathcal{C}_7$, we obtain the new unfolded program $\mathcal{P}_3 = \{\mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5, \mathcal{C}_6, \mathcal{C}_8, \mathcal{C}_9\}$ (note that clauses $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_7$ have been removed after being unfolded) where $\mathcal{C}_9 \equiv p(a) \leftarrow \boxed{\mathrm{L}_{\langle 0.7,\texttt{prod},\texttt{max}\rangle}}\ 0.9\ \boxed{\mathrm{R}_{\langle 0.7,\texttt{prod},\texttt{max}\rangle}}, 0.7$ with $\langle 0.8, \texttt{prod}, \texttt{min}\rangle$.

It is important to note now that the application of this last rule to the goal "$\leftarrow p(X), r(a)$ with $\texttt{min}$" simulates the effects of the first four resolution steps shown in the derivation of Example 3.3, which evidences the improvements achieved by unfolding on transformed programs.

5.2 The T-Norm Replacement Transformation Rule

Although we have seen in the previous section that the actions performed by unfolding on the body of transformed clauses drastically rebound in the computation/propagation of truth degrees when solving goals against transformed programs, the 'compiled-in' information collected on the body of unfolded rules admits significant numerical manipulations to eliminate or, at least to simplify, marks and real numbers. The following transformation performs this task in a very similar way to rules 3 and 4 of Definition 3.1.

Definition 5.3 [T-Norm Replacement]

[5] Both steps are performed with Rule 1 of Definition 3.1 (the first one uses clause C_6 and the second one clause C_3) and exploit the same selected -underlined- atom $q(X, Y)$.

Let $(\mathcal{P}_0, \ldots, \mathcal{P}_k)$ be a transformation sequence of If-Prolog programs starting from an f-Prolog program $\mathcal{P}_0$. Assume that $\mathcal{C} \equiv (H \leftarrow \overline{B}$ with $\Psi) \in \mathcal{P}_k$ is a (non unit) If-Prolog program clause, such that $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$. Then, the next program $\mathcal{P}_{k+1}$ in the transformation sequence can be obtained by T-Norm Replacement of clause $\mathcal{C}$ in program $\mathcal{P}_k$ as follows: $\mathcal{P}_{k+1} = (\mathcal{P}_k - \{\mathcal{C}\}) \cup \{H \leftarrow \overline{B'}$ with $\Psi'\}$, such that:

(i) if $\overline{B} \equiv (\overline{X}, \boxed{\mathrm{L}_{\langle p', [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle}}\ r\ \boxed{\mathrm{R}_{\langle p', [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle}}, \overline{Y})$, where r is a real number, then $\overline{B'} \equiv (\overline{X}, [\![\mathrm{et}_1]\!]'(p', r), \overline{Y})$ and $\Psi' \equiv \Psi$.

(ii) if $\overline{B} \equiv (\overline{X}, \boxed{\mathrm{L}_{\langle p', [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle}}\ \overline{r}\ \boxed{\mathrm{R}_{\langle p', [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle}}, \overline{Y})$, where $\overline{r} \equiv r_1, \ldots, r_n (n > 1)$ are real numbers, $\overline{B'} \equiv (\overline{X}, \boxed{\mathrm{L}_{\langle p', [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle}} [\![\mathrm{et}_2]\!]'(\overline{r}) \boxed{\mathrm{R}_{\langle p', [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle}}, \overline{Y})$ and $\Psi' \equiv \Psi$.

(iii) if $\overline{B} \equiv r$, where r is a real number, then $\overline{B'}$ is empty and $\Psi' \equiv [\![\mathrm{et}_1]\!](p, r)$.

(iv) if $\overline{B} \equiv \overline{r}$, where $\overline{r} \equiv r_1, \ldots, r_n$ $(n > 1)$ are real numbers, then $\overline{B'} \equiv [\![\mathrm{et}_2]\!](\overline{r})$ and $\Psi' \equiv \langle p, [\![\mathrm{et}_1]\!], \mathit{void}\rangle$.

The previous definition remembers the so called "algebraic replacement" traditionally used in transformation systems for pure functional programs [3,15], where a functional expression, in a program rule, is replaced by other equivalent expression w.r.t. a given algebraic property. For instance, the pure functional rule $\mathtt{f(X,Y)} \rightarrow \mathtt{X + Y}$ can be transformed by algebraic replacement into $\mathtt{f(X,Y)} \rightarrow \mathtt{Y + X}$, thanks to the commutativity of the sum. Observe also that similarly to many other pure functional transformation rules (including algebraic replacement and pure functional unfolding) our T-Norm replacement replaces an original clause by a single new one, which contrasts with the unfolding rule defined for pure logic and/or fuzzy logic programs. All these facts are not surprising, since the fuzzy component of our logic language has a functional taste, in the sense that the numerical manipulations performed by rules 3 and 4 of Definition 3.1 recall functional evaluations. This property is also inherited by Definition 5.3 where we define some particular instances of algebraic replacement, but exclusively focused on t-norm operations. For that reason, in our fuzzy setting, we decided to use the name of "T-Norm replacement" instead of "algebraic replacement" to designate this kind of transformations.

Apart from algebraic replacement, the T-Norm replacement transformation has also some similarities with fuzzy unfolding. In fact, note that the following unfolding-like definition subsumes the first kind of T-Norm replacement: $\mathcal{P}_{k+1} = (\mathcal{P}_k - \{\mathcal{C}\}) \cup \{H \leftarrow \overline{B'}$ with $\Psi \mid \langle \boxed{\mathrm{L}_\Psi}\ \overline{B}\ \boxed{\mathrm{R}_\Psi}; id\rangle \rightarrow_{FR3} \langle \boxed{\mathrm{L}_\Psi}\ \overline{B'}\ \boxed{\mathrm{R}_\Psi}; \sigma\rangle\}$, where a simple fuzzy SLD-resolution step of kind 3 have been applied for performing a T-Norm replacement of type $[\![\mathrm{et}_1]\!]$. Observe

also that, by replacing $\rightarrow_{FR3}$ with $\rightarrow_{FR4}$ in the previous definition, we obtain an alternative, unfolding-like definition, of the second and fourth kinds of T-Norm replacement (depending on which is the $[\![\mathrm{et}_2]\!]$-based lf-expression exploited: inside of $\overline{B}$ or directly the whole $\boxed{\mathrm{L}_\Psi}\,\overline{B}\,\boxed{\mathrm{R}_\Psi}$ expression). Anyway, the T-Norm replacement of kind 3 has never been considered in the literature, neither implicitly nor explicitly (although it was proposed as future work in [5]), and its application is able to transform a non unit program clause into a fact (i.e., a clause with empty body). Observe that no other transformation has this capability, which indirectly imply that, in the best case, although programs to be transformed by this transformation must necessarily belong to the lf-Prolog superclass, once transformed by T-Norm replacement of kind 3, they may recover the f-Prolog syntax.

The following example illustrates the application of transformations based on T-Norm replacement and some of their advantages.

Example 5.4 Let us continue with the transformation sequence started in example 5.2 by performing now some T-Norm replacements. So, the next program in the sequence is $\mathcal{P}_4 = (\mathcal{P}_3 - \{\mathcal{C}_9\}) \cup \{\mathcal{C}_{10}\}$, where $\mathcal{C}_{10} \equiv p(a) \leftarrow 0.63, 0.7$ with $\langle 0.8, \mathtt{prod}, \mathtt{min}\rangle$. It has been obtained by T-Norm replacement of kind 1 on clause $\mathcal{C}_9$ (note that $\mathtt{prod}(0.9, 0.7) = 0.63$). Moreover, since $\mathtt{min}(0.63, 0.7) = 0.63$, a T-Norm replacement of kind 4 on this last clause generates the new one $\mathcal{C}_{11} \equiv p(a) \leftarrow 0.63$ with $\langle 0.8, \mathtt{prod}, \mathtt{void}\rangle$. Finally clause $\mathcal{C}_{11}$ becomes in the fact $\mathcal{C}_{12} \equiv p(a) \leftarrow$ with 0.504 (since $\mathtt{prod}(0.63, 0.8) = 0.504$) after the last T-Norm replacement of kind 3. Hence, the final program is $\mathcal{P}_6 = \{\mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5, \mathcal{C}_6, \mathcal{C}_8, \mathcal{C}_{12}\}$ and now the derivation shown in example 3.3 can reduce its length in six steps thanks to the use of clause $\mathcal{C}_{12}$, which evidences once again the improvements achieved not only by unfolding, but also by T-Norm replacement, on transformed programs.

The following section is devoted to formalize the best properties one can expect of a transformation system like the our, namely:

- on the theoretical side, the exact and total correspondence between fuzzy computed answers for goals executed against the original and the transformed programs, and
- ii) on the practical side, the gains in efficiency when executing transformed programs by reducing the number of fuzzy SLD-resolution steps needed to solve a goal.

6 Properties of Fuzzy Transformations

The following Lemma, which can be seen as the counterpart of Lemma 4.2 (preservation of substitutions in f.c.a.'s on interchangeable derivation steps) is auxiliary. Intuitively it shows that, even in the case that two derivation steps can not be switched since the second one exploits an lf-expression introduced on the considered goal by the first one, its effect (w.r.t. fuzzy computed answer substitutions) can be simulated by a single step performed with a transformed clause obtained by fuzzy unfolding.

Lemma 6.1 *Let $\mathcal{P}$ be a* lf-Prolog *program, $\mathcal{G} \equiv \leftarrow \mathcal{Q}$ `with` $le_2 = [\![\mathrm{et}_2]\!]^0$ a* lf-Prolog *goal, $\Psi_0 \equiv \langle 1, void, [\![et_2]\!]^0 \rangle$ and $\mathcal{C}_1, \mathcal{C}_2 \ll \mathcal{P}$, where $\mathcal{C}_1$ is a non unit clause. Then, $\langle \boxed{\mathrm{L}_{\Psi_0}} \mathcal{Q} \boxed{\mathrm{R}_{\Psi_0}} ; \theta_0 \rangle \rightarrow_{FR1}^{\mathcal{C}_1} \langle \boxed{\mathrm{L}_{\Psi_0}} \mathcal{Q}_1 \boxed{\mathrm{R}_{\Psi_0}} ; \theta_0\theta_1 \rangle \rightarrow_{FR}^{\mathcal{C}_2}$ $\langle \boxed{\mathrm{L}_{\Psi_0}} \mathcal{Q}_2 \boxed{\mathrm{R}_{\Psi_0}} ; \theta_0\theta_1\theta_2 \rangle$ where the second steps exploits an atom introduced in $\mathcal{Q}_1$ by the first step, iff $\langle \boxed{\mathrm{L}_{\Psi_0}} \mathcal{Q} \boxed{\mathrm{R}_{\Psi_0}} ; \theta_0 \rangle \rightarrow_{FR1}^{\mathcal{C}_3} \langle \boxed{\mathrm{L}_{\Psi_0}} \mathcal{Q}_3 \boxed{\mathrm{R}_{\Psi_0}} ; \theta_0\theta_3 \rangle$, where $\mathcal{C}_3$ is obtained by fuzzy unfolding of C_1 with C_2 and $\theta_0\theta_1\theta_2 = \theta_0\theta_3$ $[\mathcal{V}ar(\mathcal{Q})]$.*

Proof.
($\Rightarrow$) Let $\mathcal{C}_1 \equiv (H_1 \leftarrow \overline{X_1}, A_1, \overline{Y_1}; \Psi)$ where $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!] \rangle$, let H_2 be the atom at the head of clause C_2 and let $\mathcal{Q} \equiv \overline{X}, A, \overline{Y}$ where $\overline{X}, \overline{Y}, \overline{X_1}$ and $\overline{Y_1}$ are arbitrary sequences of valid lf-expressions and A is the atom selected in $\mathcal{Q}$ by the considered computation rule. Then, in the f-derivation $\langle \boxed{\mathrm{L}_{\Psi_0}} \mathcal{Q} \boxed{\mathrm{R}_{\Psi_0}} ; \theta_0 \rangle \rightarrow_{FR1}^{\mathcal{C}_1} \langle \boxed{\mathrm{L}_{\Psi_0}} \mathcal{Q}_1 \boxed{\mathrm{R}_{\Psi_0}} ; \theta_0\theta_1 \rangle \rightarrow_{FR}^{\mathcal{C}_2}$ $\langle \boxed{\mathrm{L}_{\Psi_0}} \mathcal{Q}_2 \boxed{\mathrm{R}_{\Psi_0}} ; \theta_0\theta_1\theta_2 \rangle$ we have that: $\theta_1 = mgu(\{A = H_1\})$, $\mathcal{Q}_1 \equiv (\overline{X}, \boxed{\mathrm{L}_{\Psi}} \overline{X_1}, A_1, \overline{Y_1} \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\theta_1$. Moreover, if $A_1\theta_1$ is the atom selected in $\mathcal{Q}_1$ by the considered computation rule, then $\theta_2 = mgu(\{A_1\theta_1 = H_2\})$. Now, consider $\sigma = mgu(\{A_1 = H_2\})$. Then, the following equalities hold:

$$\begin{aligned} \theta_1\theta_2 &= \\ \theta_1 mgu(\{A_1\theta_1 = H_2\}) &= \qquad (\text{since } Dom(\theta_1) \cap \mathcal{V}ar(\mathcal{C}_2) = \emptyset) \end{aligned}$$

$$
\begin{array}{lcl}
\theta_1 mgu(\widehat{mgu}(\{A_1 = H_2\})\theta_1) = & & \\
\theta_1 mgu(\widehat{\sigma}\theta_1) & = & \text{(by Lemma 4.1)} \\
\theta_1 \Uparrow \sigma & = & \text{(by Lemma 4.1)} \\
\sigma mgu(\widehat{\theta_1}\sigma) & = & \\
\sigma mgu(\widehat{mgu}(\{A = H_1\})\sigma) & = & \text{(since } Dom(\sigma) \cap \mathcal{V}ar(\mathcal{Q}) = \emptyset\text{)} \\
\sigma mgu(\{A = H_1\sigma\}). & &
\end{array}
$$

Moreover, since $\theta_1\theta_2 \not\equiv fail$, then $\sigma \not\equiv fail$ and thus there exists a clause $\mathcal{C}_3$ obtained by unfolding (atom A_1 in the body of) $\mathcal{C}_1$ by using $\mathcal{C}_2$, such that the head of $\mathcal{C}_3$ is the atom $H_1\sigma$. Now, since $mgu(\{A = H_1\sigma\}) \not\equiv fail$, the following f-resolution step done on the selected atom A in $\mathcal{Q} \equiv \overline{X}, A, \overline{Y}$ can be proved: $\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}\boxed{\mathrm{R}_{\Psi_0}}; \theta_0\rangle \rightarrow_{FR1}^{\mathcal{C}_3} \langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}_3\boxed{\mathrm{R}_{\Psi_0}}; \theta_0\theta_3\rangle$, where $\theta_3 = mgu(\{A = H_1\sigma\})$. Finally, since $\theta_1\theta_2 = \sigma\theta_3$, then $\theta_0\theta_1\theta_2 = \theta_0\sigma\theta_3$, and since $Dom(\sigma) \cap \mathcal{V}ar(\mathcal{Q}) = \emptyset$ and $Dom(\sigma) \cap Dom(\theta_0) = \emptyset$, we have that $\theta_0\theta_1\theta_2 = \theta_0\theta_3$ $[\mathcal{V}ar(\mathcal{Q})]$, as we wanted to prove.
($\Leftarrow$) This case can be easily proved in a similar way as the previous one, by also exploiting the equivalence between $\theta_1\theta_2$ and $\sigma\theta_3$. □

In order to prove Theorem 6.4, we treat separately both claims of the double implication, since the strong correctness of fuzzy unfolding or T-Norm Replacement implies both strong soundness ($\Leftarrow$) and strong completeness ($\Rightarrow$).

Theorem 6.2 (Strong Soundness) *Let $\mathcal{P}$ be a* If-Prolog *program and let $\mathcal{G} \equiv \leftarrow\mathcal{Q}$ with $le_2 = [\![\mathrm{et}_2]\!]^0$ be a* If-Prolog *goal. If $\mathcal{P}'$ is an* If-Prolog *program obtained by fuzzy unfolding or by T-Norm Replacement of $\mathcal{P}$, then $\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}\boxed{\mathrm{R}_{\Psi_0}}; id\rangle \rightarrow_{FR}{}^*\langle r; \theta\rangle$ in $\mathcal{P}$ if $\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}\boxed{\mathrm{R}_{\Psi_0}}; id\rangle \rightarrow_{FR}{}^*\langle r; \theta'\rangle$ in $\mathcal{P}'$, where $\theta = \theta'[\mathcal{V}ar(\mathcal{Q})]$ and $\Psi_0 \equiv \langle 1, void, [\![et_2]\!]^0\rangle$.*

Proof. Let $\mathcal{D}' \equiv [\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}\boxed{\mathrm{R}_{\Psi_0}}; id\rangle \rightarrow_{FR}{}^*\langle r; \theta\rangle]$ be the (generic) successful derivation for $\mathcal{G}$ in $\mathcal{P}'$ that we plan to simulate by constructing a new derivation $\mathcal{D}$ in $\mathcal{P}$. The construction of $\mathcal{D}$ is done by induction on the length of $\mathcal{D}'$, n. Since the case base, i.e. $n = 0$, is trivial, we proceed with the general case when $n > 0$. Then, $\mathcal{D}' \equiv [\langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}\boxed{\mathrm{R}_{\Psi_0}}; id\rangle \rightarrow_{FR} \langle \boxed{\mathrm{L}_{\Psi_0}}\mathcal{Q}'\boxed{\mathrm{R}_{\Psi_0}}; \vartheta\rangle \rightarrow_{FR}{}^*\langle r; \theta'\rangle]$. If the first step of $\mathcal{D}'$ has been given with the second, third or fourth rule of Definition 3.1, or, even it is has been performed with the first one but using a clause also belonging to $\mathcal{P}$, then the claim follows by the inductive hypothesis. Otherwise, this initial step is done with rule 1 using a clause $\mathcal{C}'$ that has been obtained by unfolding or T-Norm Replacement other clause $\mathcal{C} \in \mathcal{P}$. Since the

unfolding step has been performed with one of the two rules of Definition 3.1, and the T-Norm Replacement step has been performed with one of the three rules of Definition 5.3 we treat each case separately.

(i) Unfolding based on Rule 1.
Let $\mathcal{C} \equiv (H_1 \leftarrow \overline{X_1}, A_1, \overline{Y_1}; \Psi) \in \mathcal{P}$ with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$ and $\mathcal{C}_2 \equiv (H_2 \leftarrow \overline{B_2}; \Psi') \in \mathcal{P}$ with $\Psi' \equiv \langle q, [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle$ such that, by unfolding $\mathcal{C}$ w.r.t. $\mathcal{C}_2$ using the Definition 5.1, we obtain the clause $\mathcal{C}' \equiv ((H_1 \leftarrow \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ \overline{B_2}\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1})\sigma; \Psi) \in \mathcal{P}'$. Now, assume that $\mathcal{Q} \equiv \overline{X}, A, \overline{Y}$ where $\overline{X}$ and $\overline{Y}$ are arbitrary sequences of valid lf-expressions and A is the selected atom. Then, $\mathcal{D}'$ has the following form:

$$\begin{array}{ll} \langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}'} \\ \langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ \overline{B_2}\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\sigma\gamma; \sigma\gamma\rangle & \rightarrow_{FR}{}^{*} \\ \langle r; \theta'\rangle & \end{array}$$

And now, the first step of $\mathcal{D}'$ can be simulated in derivation $\mathcal{D}$ by using clauses $\mathcal{C}$ and $\mathcal{C}_2$ of $\mathcal{P}$ as follows:

$$\begin{array}{ll} \langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}} \\ \langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, A_1, \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR1}{}^{\mathcal{C}_2} \\ \langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ \overline{B_2}\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\alpha\beta; \alpha\beta\rangle & \rightarrow_{FR}{}^{*} \\ \langle r; \theta\rangle & \end{array}$$

By Lemma 6.1 we can conclude that $\alpha\beta = \sigma\gamma[\mathcal{V}ar(\mathcal{Q})]$, and hence the third state in $\mathcal{D}$ coincides syntactically with the second one in $\mathcal{D}'$. Moreover, by the inductive hypothesis $\theta = \theta'[\mathcal{V}ar(\mathcal{Q})]$ and hence the entire f-derivations $\mathcal{D}$ and $\mathcal{D}'$ are equivalents, as we wanted to prove.

(ii) Unfolding based on Rule 2.
Let $\mathcal{C} \equiv (H_1 \leftarrow \overline{X_1}, A_1, \overline{Y_1}; \Psi) \in \mathcal{P}$ with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$ and $\mathcal{C}_2 \equiv (H_2 \leftarrow; q) \in \mathcal{P}$ such that, by unfolding $\mathcal{C}$ w.r.t. $\mathcal{C}_2$ using the Definition 5.1, we obtain: $\mathcal{C}' \equiv ((H_1 \leftarrow \overline{X_1}, q, \overline{Y_1})\sigma; \Psi) \in \mathcal{P}'$. Then, $\mathcal{D}'$ has the

following form:

$$\begin{array}{ll}\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}'}\\ \langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{X_1}, q, \overline{Y_1}\ \boxed{\mathrm{R}_\Psi}, \overline{Y})\sigma\gamma; \sigma\gamma\rangle & \rightarrow_{FR}{}^{*}\\ \langle r; \theta'\rangle & \end{array}$$

And now, the first step of $\mathcal{D}'$ can be simulated in derivation $\mathcal{D}$ by using clauses $\mathcal{C}$ and $\mathcal{C}_2$ of $\mathcal{P}$ as follows:

$$\begin{array}{ll}\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}}\\ \langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{X_1}, A_1, \overline{Y_1}\ \boxed{\mathrm{R}_\Psi}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR2}{}^{\mathcal{C}_2}\\ \langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{X_1}, q, \overline{Y_1}\ \boxed{\mathrm{R}_\Psi}, \overline{Y})\alpha\beta; \alpha\beta\rangle & \rightarrow_{FR}{}^{*}\\ \langle r; \theta\rangle & \end{array}$$

By Lemma 6.1 we can conclude that $\alpha\beta = \sigma\gamma[\mathcal{V}ar(\mathcal{Q})]$, and hence the third state in $\mathcal{D}$ coincides syntactically with the second one in $\mathcal{D}'$. Moreover, by the inductive hypothesis $\theta = \theta'[\mathcal{V}ar(\mathcal{Q})]$ and hence the entire f-derivations $\mathcal{D}$ and $\mathcal{D}'$ are equivalents, as we wanted to prove.

(iii) T-Norm Replacement of kind 1.

Let $\mathcal{C} \equiv (H_1 \leftarrow \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ n\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}; \Psi) \in \mathcal{P}$ with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$, and let $\Psi' \equiv \langle q, [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle$. By T-Norm Replacement of clause $\mathcal{C}$ using the rule of kind 1 in Definition 5.3, we obtain the clause $\mathcal{C}' \equiv (H_1 \leftarrow \overline{X_1}, [\![\mathrm{et}_1']\!](q, n), \overline{Y_1}; \Psi) \in \mathcal{P}'$. Then, $\mathcal{D}'$ has the following form:

$$\begin{array}{ll}\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}'}\\ \langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{X_1}, [\![\mathrm{et}_1]\!]'(q, n), \overline{Y_1}\ \boxed{\mathrm{R}_\Psi}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*}\\ \langle r; \theta'\rangle & \end{array}$$

And now, the first step of $\mathcal{D}'$ can be simulated in $\mathcal{P}$ by giving two resolution steps in $\mathcal{D}$: the first one with rule 1 using clause $\mathcal{C}$ and the second one with rule 3, as follows:

$$\begin{array}{ll}\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}}\\ \langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ n\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}\ \boxed{\mathrm{R}_\Psi}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR3}\\ \langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{X_1}, [\![\mathrm{et}_1]\!]'(q, n), \overline{Y_1}\ \boxed{\mathrm{R}_\Psi}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*}\\ \langle r; \theta\rangle & \end{array}$$

Since the third state in $\mathcal{D}$ coincides syntactically with the second one in $\mathcal{D}'$, our claim holds by the inductive hypothesis.

(iv) T-Norm Replacement of kind 2.

Let $\mathcal{C} \equiv (H_1 \leftarrow \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ \overline{n}\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}; \Psi) \in \mathcal{P}$, with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$ and let $\Psi' \equiv \langle q, [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle$. By T-Norm Replacement of $\mathcal{C}$ using the rule of kind 2 in Definition 5.3, we obtain the new clause: $\mathcal{C}' \equiv (H_1 \leftarrow \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ [\![\mathrm{et}_2]\!]'(\overline{n})\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}; \Psi) \in \mathcal{P}'$. Then, $\mathcal{D}'$ has the following form:

$$\begin{array}{ll} \langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}'} \\ \langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ [\![\mathrm{et}_2]\!]'(\overline{n})\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*} \\ \langle r; \theta'\rangle & \end{array}$$

And now, the first step of $\mathcal{D}'$ can be simulated with clauses of $\mathcal{P}$ by giving two resolution steps in $\mathcal{D}$: the first one with rule 1 using clause $\mathcal{C}$ and the second one with rule 4, as follows:

$$\begin{array}{ll} \langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}} \\ \langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ \overline{n}\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR4} \\ \langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ [\![\mathrm{et}_2]\!]'(\overline{n})\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*} \\ \langle r; \theta\rangle & \end{array}$$

Since the third state in $\mathcal{D}$ coincides syntactically with the second one in $\mathcal{D}'$, our claim holds by the inductive hypothesis.

(v) T-Norm Replacement of kind 3.

Let $\mathcal{C} \equiv (H_1 \leftarrow n; \Psi) \in \mathcal{P}$, with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$, such that, by T-Norm Replacement of $\mathcal{C}$ using the rule of kind 3 in Definition 5.3, we obtain the new clause $\mathcal{C}' \equiv (H_1 \leftarrow\ ; [\![\mathrm{et}_1]\!](p, n)) \in \mathcal{P}'$. Then, $\mathcal{D}'$ has the following form:

$$\begin{array}{ll} \langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR2}{}^{\mathcal{C}'} \\ \langle (\overline{X}, [\![\mathrm{et}_1]\!](p, n), \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*} \\ \langle r; \theta'\rangle & \end{array}$$

And now, the first step of $\mathcal{D}'$ can be simulated with clauses of $\mathcal{P}$ by giving two resolution steps in $\mathcal{D}$: the first one with rule 1 using clause $\mathcal{C}$ and

the second one with rule 3, as follows:

$$\begin{array}{ll}
\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}} \\
\langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ n\ \boxed{\mathrm{R}_\Psi}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR3} \\
\langle (\overline{X}, [\![\mathrm{et}_1]\!](p,n), \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*} \\
\langle r; \theta\rangle &
\end{array}$$

Since the third state in $\mathcal{D}$ coincides syntactically with the second one in $\mathcal{D}'$, our claim holds by the inductive hypothesis.

(vi) T-Norm Replacement of kind 4.
Let $\mathcal{C} \equiv (H_1 \leftarrow \overline{n}; \Psi) \in \mathcal{P}$, with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$, such that, by T-Norm Replacement of $\mathcal{C}$ using the rule of kind 4 in Definition 5.3, we obtain the new clause $\mathcal{C}' \equiv (H_1 \leftarrow [\![\mathrm{et}_2]\!](\overline{n}); \Psi) \in \mathcal{P}'$. Then, $\mathcal{D}'$ has the following form:

$$\begin{array}{ll}
\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}'} \\
\langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ [\![\mathrm{et}_2]\!](\overline{n})\ \boxed{\mathrm{R}_\Psi}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*} \\
\langle r; \theta'\rangle &
\end{array}$$

And now, the first step of $\mathcal{D}'$ can be simulated with clauses of $\mathcal{P}$ by giving two resolution steps in $\mathcal{D}$: the first one with rule 1 using clause $\mathcal{C}$ and the second one with rule 3, as follows:

$$\begin{array}{ll}
\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}} \\
\langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{n}\ \boxed{\mathrm{R}_\Psi}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR4} \\
\langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ [\![\mathrm{et}_2]\!](\overline{n})\ \boxed{\mathrm{R}_\Psi}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*} \\
\langle r; \theta\rangle &
\end{array}$$

Since the third state in $\mathcal{D}$ coincides syntactically with the second one in $\mathcal{D}'$, our claim holds by the inductive hypothesis.

□

Now, we proceed with the counterpart of the previous Theorem, that is, the strong completeness.

Theorem 6.3 (Strong Completeness) *Let $\mathcal{P}$ be a* If-Prolog *program and let $\mathcal{G} \equiv \leftarrow \mathcal{Q}$ with $le_2 = [\![\mathrm{et}_2]\!]^0$ be a* If-Prolog *goal. If $\mathcal{P}'$ is an* If-Prolog *program obtained by fuzzy unfolding or T-Norm Replacement of $\mathcal{P}$, then,*

$\langle \boxed{\mathrm{L}_{\Psi_0}}\,\mathcal{Q}\,\boxed{\mathrm{R}_{\Psi_0}}; id\rangle \rightarrow_{FR}{}^{*} \langle r; \theta'\rangle$ *in* $\mathcal{P}'$ *if* $\langle \boxed{\mathrm{L}_{\Psi_0}}\,\mathcal{Q}\,\boxed{\mathrm{R}_{\Psi_0}}; id\rangle \rightarrow_{FR}{}^{*} \langle r; \theta\rangle$ *in* $\mathcal{P}$, *where* $\theta' = \theta[\mathcal{V}ar(\mathcal{Q})]$ *and* $\Psi_0 \equiv \langle 1, void, [\![et_2]\!]^0\rangle$.

Proof. Our proof consists in simulating in $\mathcal{P}'$ a re-ordered successful f-derivation originally performed in $\mathcal{P}$. So, let $\mathcal{D}_0 \equiv [\langle \boxed{\mathrm{L}_{\Psi_0}}\,\mathcal{Q}\,\boxed{\mathrm{R}_{\Psi_0}}; id\rangle \rightarrow_{FR}{}^{n} \langle r; \theta_0\rangle]$ be a (generic) n-steps successful f-derivation for $\mathcal{G}$ in $\mathcal{P}$. Assume now that $\mathcal{C} \in \mathcal{P}$ is the clause unfolded in $\mathcal{P}$ which obviously does not belong to $\mathcal{P}'$. Any existing step done with clause $\mathcal{C}$ in $\mathcal{D}_0$ introduces an instance of the body of $\mathcal{C}$ in the next state of the derivation. Since we are dealing which a successful derivation, this *instanciated* body of $\mathcal{C}$ must necessarily be reduced in the immediately next step or in subsequent ones. For the second case, we can safely interchange the step done with clause $\mathcal{C}$ and the next one, by application of the Switching Lemma 4.3. Moreover, by repeated application of this Lemma, we can obtain a new n-steps f-derivation $\mathcal{D} \equiv [\langle \boxed{\mathrm{L}_{\Psi_0}}\,\mathcal{Q}\,\boxed{\mathrm{R}_{\Psi_0}}; id\rangle \rightarrow_{FR}{}^{n} \langle r; \theta\rangle]$ in $\mathcal{P}$ verifying $\theta = \theta_0[\mathcal{V}ar(\mathcal{Q})]$, where any step (if it exists) using the clause $\mathcal{C}$ unfolded in $\mathcal{P}$, is followed by other step exploiting an lf-expression just introduced by the previous step (i.e., belonging to the instanciated body of $\mathcal{C}$). We say that $\mathcal{D}$ is an successful f-derivation re-ordered w.r.t. clause $\mathcal{C}$.

Now, and similarly to the previous theorem, we are going to simulate $\mathcal{D}$ in $\mathcal{P}'$ by constructing a new derivation $\mathcal{D}'$ using the clauses of $\mathcal{P}'$ and following an schema perfectly analogous to the one used in Theorem 6.2, but inverting now the use of terms $\mathcal{P}$ and $\mathcal{P}'$ (and related ones). The construction of $\mathcal{D}'$ is done by induction on the length of $\mathcal{D}$, n. Since the case base, i.e. $n = 0$, is trivial, we proceed with the general case when $n > 0$. Then, $\mathcal{D} \equiv [\langle \boxed{\mathrm{L}_{\Psi_0}}\,\mathcal{Q}\,\boxed{\mathrm{R}_{\Psi_0}}; id\rangle \rightarrow_{FR} \langle \boxed{\mathrm{L}_{\Psi_0}}\,\mathcal{Q}'\,\boxed{\mathrm{R}_{\Psi_0}}\vartheta\rangle \rightarrow_{FR}{}^{*} \langle r; \theta\rangle]$. If the first step of $\mathcal{D}$ has been given with the second, third or fourth rule of Definition 3.1, or, even it is has been performed with the first one but using a clause also belonging to $\mathcal{P}'$, then the claim follows by the inductive hypothesis. Otherwise, this initial step is done with rule 1 using a clause $\mathcal{C}$ that, once it is transformed, generates the new clause $\mathcal{C}' \in \mathcal{P}'$. Since the unfolding step has been performed with one of the four rules of Definition 5.1, and the T-Norm Replacement step fulfil one of de four rules of Definition 5.3 we treat each case separately.

(i) Unfolding based on Rule 1.
Let $\mathcal{C} \equiv (H_1 \leftarrow \overline{X_1}, A_1, \overline{Y_1}; \Psi) \in \mathcal{P}$ with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$ and $\mathcal{C}_2 \equiv (H_2 \leftarrow \overline{B_2}; \Psi') \in \mathcal{P}$ with $\Psi' \equiv \langle q, [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle$. By unfolding $\mathcal{C}$ w.r.t. $\mathcal{C}_2$ we obtain: $\mathcal{C}' \equiv ((H_1 \leftarrow \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\,\overline{B_2}\,\boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1})\sigma; \Psi) \in \mathcal{P}'$. Now, assume that $\mathcal{Q} \equiv \overline{X}, A, \overline{Y}$ where $\overline{X}$ and $\overline{Y}$ are arbitrary sequences of valid

lf-expressions and A is the selected atom. Then, since $\mathcal{D}$ is a successful f-derivation reordered w.r.t. clause $\mathcal{C}$, then it has the following form:

$$\begin{array}{ll}
\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}} \\
\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, A_1, \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR1}{}^{\mathcal{C}_2} \\
\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ \overline{B_2}\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\alpha\beta; \alpha\beta\rangle & \rightarrow_{FR}{}^{*} \\
\langle r; \theta\rangle &
\end{array}$$

And now, the first two steps of $\mathcal{D}$ can be simulated in $\mathcal{P}'$ by using clause $\mathcal{C}'$ in derivation $\mathcal{D}'$ as follows:

$$\begin{array}{ll}
\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}'} \\
\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ \overline{B_2}\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y})\sigma\gamma; \sigma\gamma\rangle & \rightarrow_{FR}{}^{*} \\
\langle r; \theta'\rangle &
\end{array}$$

By Lemma 6.1 we can conclude that $\alpha\beta = \sigma\gamma[\mathcal{V}ar(\mathcal{Q})]$, and hence the third state in $\mathcal{D}$ coincides syntactically with the second one in $\mathcal{D}'$. Moreover, by the inductive hypothesis $\theta = \theta'[\mathcal{V}ar(\mathcal{Q})]$ and hence the entire f-derivations $\mathcal{D}$ and $\mathcal{D}'$ are equivalents, as we wanted to prove.

(ii) Unfolding based on Rule 2.
Let $\mathcal{C} \equiv (H_1 \leftarrow \overline{X_1}, A_1, \overline{Y_1}; \Psi) \in \mathcal{P}$ with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$ and $\mathcal{C}_2 \equiv (H_2 \leftarrow; q) \in \mathcal{P}$ such that, by unfolding $\mathcal{C}$ w.r.t. $\mathcal{C}_2$ using the Definition 5.1, we obtain: $\mathcal{C}' \equiv ((H_1 \leftarrow \overline{X_1}, q, \overline{Y_1})\sigma; \Psi) \in \mathcal{P}'$. Then, since $\mathcal{D}$ is a successful f-derivation reordered w.r.t. clause $\mathcal{C}$, then it has the following form:

$$\begin{array}{ll}
\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}} \\
\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, A_1, \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR2}{}^{\mathcal{C}_2} \\
\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, q, \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\alpha\beta; \alpha\beta\rangle & \rightarrow_{FR}{}^{*} \\
\langle r; \theta\rangle &
\end{array}$$

And now, the first two steps in $\mathcal{D}$ can be simulated in $\mathcal{P}'$ by using clause $\mathcal{C}'$ as follows:

$$\begin{array}{ll}
\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}'} \\
\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, q, \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\sigma\gamma; \sigma\gamma\rangle & \rightarrow_{FR}{}^{*} \\
\langle r; \theta'\rangle &
\end{array}$$

By Lemma 6.1 we can conclude that $\alpha\beta = \sigma\gamma[\mathcal{V}ar(\mathcal{Q})]$, and hence the third state in $\mathcal{D}$ coincides syntactically with the second one in $\mathcal{D}'$. Moreover, by the inductive hypothesis $\theta = \theta'[\mathcal{V}ar(\mathcal{Q})]$ and hence the entire f-derivations $\mathcal{D}$ and $\mathcal{D}'$ are equivalents, as we wanted to prove.

(iii) T-Norm Replacement of kind 1.
Let $\mathcal{C} \equiv (H_1 \leftarrow \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ n\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}; \Psi) \in \mathcal{P}$ with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$ and let $\Psi' \equiv \langle q, [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle$. By T-Norm Replacement of kind 1 of $\mathcal{C}$ (see the Definition 5.3), we obtain $\mathcal{C}' \equiv (H_1 \leftarrow \overline{X_1}, [\![\mathrm{et}_1]\!](q,n), \overline{Y_1}; \Psi) \in \mathcal{P}'$. Moreover, since $\mathcal{D}$ is a successful f-derivation reordered w.r.t. clause $\mathcal{C}$, then it has the following form:

$$
\begin{array}{ll}
\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}} \\
\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ n\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR3} \\
\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, [\![\mathrm{et}_1]\!]'(q,n), \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*} \\
\langle r; \theta\rangle &
\end{array}
$$

And now, the first two steps in $\mathcal{D}$ can be simulated in $\mathcal{P}'$ by giving a unique resolution step using clause $\mathcal{C}'$ in $\mathcal{D}'$ as follows:

$$
\begin{array}{ll}
\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}'} \\
\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, [\![\mathrm{et}_1]\!]'(q,n), \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}, \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*} \\
\langle r; \theta'\rangle &
\end{array}
$$

Since the third state in $\mathcal{D}$ coincides syntactically with the second one in $\mathcal{D}'$, our claim holds by the inductive hypothesis.

(iv) T-Norm Replacement of kind 2.
Let $\mathcal{C} \equiv (H_1 \leftarrow \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ \overline{n}\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}; \Psi) \in \mathcal{P}$ with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$ and $\Psi' \equiv \langle q, [\![\mathrm{et}_1]\!]', [\![\mathrm{et}_2]\!]'\rangle$. By T-Norm Replacement of kind 2 of $\mathcal{C}$ using the Definition 5.3, we get $\mathcal{C}' \equiv (H_1 \leftarrow \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ [\![\mathrm{et}_2]\!](\overline{n})\ \boxed{\mathrm{R}_{\Psi'}}, \overline{Y_1}; \Psi) \in \mathcal{P}'$. Moreover, since $\mathcal{D}$ is a successful f-derivation reordered w.r.t.

clause $\mathcal{C}$, then it has the following form:

$$\begin{array}{ll}
\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}} \\
\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ \overline{n}\ \boxed{\mathrm{R}_{\Psi'}}\ , \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}\ , \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR4} \\
\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ [\![\mathrm{et}_2]\!]'(\overline{n})\ \boxed{\mathrm{R}_{\Psi'}}\ , \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}\ , \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*} \\
\langle r; \theta\rangle &
\end{array}$$

And now, the first two steps in $\mathcal{D}$ can be simulated in $\mathcal{P}'$ by giving a unique resolution step using clause $\mathcal{C}'$ in $\mathcal{D}'$ as follows:

$$\begin{array}{ll}
\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}'} \\
\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ \overline{X_1}, \boxed{\mathrm{L}_{\Psi'}}\ [\![\mathrm{et}_2]\!]'(\overline{n})\ \boxed{\mathrm{R}_{\Psi'}}\ , \overline{Y_1}\ \boxed{\mathrm{R}_{\Psi}}\ , \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*} \\
\langle r; \theta'\rangle &
\end{array}$$

Since the third state in $\mathcal{D}$ coincides syntactically with the second one in $\mathcal{D}'$, our claim holds by the inductive hypothesis.

(v) T-Norm Replacement of kind 3.
Let $\mathcal{C} \equiv (H_1 \leftarrow n; \Psi) \in \mathcal{P}$, with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!]\rangle$, such that, by T-Norm Replacement of $\mathcal{C}$ using the rule of kind 3 in Definition 5.3, we obtain the new clause $\mathcal{C}' \equiv (H_1 \leftarrow\ ; [\![\mathrm{et}_1]\!](p, n)) \in \mathcal{P}'$. Moreover, since $\mathcal{D}$ is a successful f-derivation reordered w.r.t. clause $\mathcal{C}$, then it has the following form:

$$\begin{array}{ll}
\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR1}{}^{\mathcal{C}} \\
\langle (\overline{X}, \boxed{\mathrm{L}_{\Psi}}\ n\ \boxed{\mathrm{R}_{\Psi}}\ , \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR3} \\
\langle (\overline{X}, [\![\mathrm{et}_1]\!](p, n), \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*} \\
\langle r; \theta\rangle &
\end{array}$$

Now, the first two steps in $\mathcal{D}$ (the first one with rule 1 using clause $\mathcal{C}$ and the second one with rule 3) can be simulated in $\mathcal{P}'$ by giving a unique resolution step using clause $\mathcal{C}'$ in $\mathcal{D}'$ as follows:

$$\begin{array}{ll}
\langle \overline{X}, A, \overline{Y}; id\rangle & \rightarrow_{FR2}{}^{\mathcal{C}'} \\
\langle (\overline{X}, [\![\mathrm{et}_1]\!](p, n), \overline{Y})\alpha; \alpha\rangle & \rightarrow_{FR}{}^{*} \\
\langle r; \theta'\rangle &
\end{array}$$

Since the third state in $\mathcal{D}$ coincides syntactically with the second one

in $\mathcal{D}'$, our claim holds by the inductive hypothesis.

(vi) T-Norm Replacement of kind 4.
Let $\mathcal{C} \equiv (H_1 \leftarrow \overline{n}; \Psi) \in \mathcal{P}$, with $\Psi \equiv \langle p, [\![\mathrm{et}_1]\!], [\![\mathrm{et}_2]\!] \rangle$, such that, by T-Norm Replacement of $\mathcal{C}$ using the rule of kind 4 in Definition 5.3, we obtain the new clause $\mathcal{C}' \equiv (H_1 \leftarrow [\![\mathrm{et}_2]\!](\overline{n}); \Psi) \in \mathcal{P}'$. Moreover, since $\mathcal{D}$ is a successful f-derivation reordered w.r.t. clause $\mathcal{C}$, then it has the following form:

$$\begin{array}{ll} \langle \overline{X}, A, \overline{Y}; id \rangle & \rightarrow_{FR1}{}^{\mathcal{C}} \\ \langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ \overline{n}\ \boxed{\mathrm{R}_\Psi}, \overline{Y})\alpha; \alpha \rangle & \rightarrow_{FR4} \\ \langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ [\![\mathrm{et}_2]\!](\overline{n})\ \boxed{\mathrm{R}_\Psi}, \overline{Y})\alpha; \alpha \rangle & \rightarrow_{FR}{}^{*} \\ \langle r; \theta \rangle & \end{array}$$

Now, the first two steps in $\mathcal{D}$ (the first one with rule 1 using clause $\mathcal{C}$ and the second one with rule 4) can be simulated in $\mathcal{P}'$ by giving a unique resolution step using clause $\mathcal{C}'$ in $\mathcal{D}'$ as follows:

$$\begin{array}{ll} \langle \overline{X}, A, \overline{Y}; id \rangle & \rightarrow_{FR1}{}^{\mathcal{C}'} \\ \langle (\overline{X}, \boxed{\mathrm{L}_\Psi}\ [\![\mathrm{et}_2]\!](\overline{n})\ \boxed{\mathrm{R}_\Psi}, \overline{Y})\alpha; \alpha \rangle & \rightarrow_{FR}{}^{*} \\ \langle r; \theta' \rangle & \end{array}$$

Since the third state in $\mathcal{D}$ coincides syntactically with the second one in $\mathcal{D}'$, our claim holds by the inductive hypothesis.

□

Finally, we are able to formalize and prove the best properties of our fuzzy transformations (namely, its strong correctness and the guarantee for producing improvements on transformed programs) as follows:

Theorem 6.4 (Strong Correctness of the Transformation System) *Let $(\mathcal{P}_0, \ldots, \mathcal{P}_k)$ be a transformation sequence of* lf-Prolog *programs where each program in the sequence, except the initial* f-Prolog *program $\mathcal{P}_0$, is obtained from the immediately preceding one by applying fuzzy unfolding or T-Norm replacement. Then, for any* lf-Prolog *goal $\mathcal{G} \equiv \leftarrow \mathcal{Q}$ with $le_2 = [\![\mathrm{et}_2]\!]$, we have:*

$$\langle \boxed{\mathrm{L}_\Psi}\mathcal{Q}\boxed{\mathrm{R}_\Psi}; id \rangle \rightarrow_{FR}{}^{n} \langle r; \theta \rangle \text{ in } \mathcal{P}_0 \text{ iff } \langle \boxed{\mathrm{L}_\Psi}\mathcal{Q}\boxed{\mathrm{R}_\Psi}; id \rangle \rightarrow_{FR}{}^{m} \langle r; \theta \rangle \text{ in } \mathcal{P}_k$$

where $\Psi \equiv \langle 1, void, [\![\mathrm{et}_2]\!] \rangle$ and the number of fuzzy SLD-resolution steps in each derivation verify that $m \leq n$.

Proof. The two claims of Theorem 6.4 can be easily proved as follows:

- The strong correctness of the transformation system is immediate by simply applying Theorems 6.2 and 6.3, since both theorems prove the equivalence between any pair of consecutive programs inside the transformation sequence $(\mathcal{P}_0, \ldots, \mathcal{P}_k)$, thus implying the final equivalence between $\mathcal{P}_0$ and $\mathcal{P}_k$.
- Regarding the reduction of the length of successful derivation in transformed programs, we have seen in proofs of both theorems 6.2 and 6.3 that any fuzzy SLD-resolution step done with a new clause obtained after applying fuzzy unfolding or t-norm replacement subsumes two fuzzy SLD-resolution steps done against the original program, which confirms that $m \leq n$, as we wanted to prove.

□

7 Conclusions

This work introduces two safe unfolding-based transformation rules for optimizing fuzzy logic programs. To the best of our knowledge, this is the first time the issue, of integrating transformation techniques in the context of fuzzy logic languages, is treated in the literature.

After an inspection of the main proposals for the inclusion of fuzzy logic into a logic programming setting, we have selected an extension of the language described in [21], that we call If-Prolog, since we think it is the best suited to deal with the problems that may arise in the transformation process of logic programs. It is remarkable that If-Prolog is provided with a labeled mark language. Inspired in [19], we have extended this language in order to be able to code different fuzzy logics inside the same program, which greatly enhances the expressive power of the former language. Also, as an auxiliary result, we have established the independence of the Fuzzy Computation Rule for If-Prolog programs and goals (Theorem 4.4).

We have defined the fuzzy unfolding and T-Norm replacement of If-Prolog programs (Definition 5.1 and 5.3) and we have demonstrated the (strong) correctness (Theorem 6.4) of the transformation system. Moreover, we have proved that transformation sequences can be guided in a blind way since any transformation step based on fuzzy unfolding and T-Norm replacement always produces an improvement on transformed programs. This contrasts with other transformation rules, like definition introduction or folding that may degrade the efficiency of programs, if appropriate "transformation strategies" are not used to drive the construction of the transformation sequence.

Finally, it is important to say that the results in this paper can be thought

as a basis to optimize fuzzy prolog programs and they are the first step in the construction of a global fold/unfold framework (including more transformation rules and strategies) for optimizing this class of programs.

References

[1] M. Alpuente, M. Falaschi, G. Moreno, and G. Vidal. Rules + Strategies for Transforming Lazy Functional Logic Programs. *Theoretical Computer Science*, 311:479–525, 2004.

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

[3] R.M. Burstall and J. Darlington. A Transformation System for Developing Recursive Programs. *Journal of the ACM*, 24(1):44–67, 1977.

[4] M. Ishizuka and N. Kanai. Prolog-ELF Incorporating Fuzzy Logic. In Aravind K. Joshi, editor, *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI'85). Los Angeles, CA, August 1985.*, pages 701–703. Morgan Kaufmann, 1985.

[5] P. Julian, G. Moreno, and J. Penabad. Unfolding Fuzzy Logic Programs. In *Proc. of the Fourth International Conference on Intelligent Systems Design and Applications, ISDA'04, Budapest, Hungary*, pages 595–600. (Sponsored by IEEE), 2004.

[6] P. Julian and C. Villamizar. Analizing Definitional Trees: Looking for Determinism. In Yukiyoshi Kameyama and M. Peter J. Stuckey, editors, *Proc. of the 7th Fuji International Symposyum on Functional and Logic Programming, FLOPS'04, Nara (Japan)*, page 25. Springer-Verlag LNCS (To appear), 2004.

[7] Frank Klawonn and Rudolf Kruse. A Łukasiewicz logic based Prolog. *Mathware & Soft Computing*, 1(1):5–29, 1994.

[8] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic*. Prentice-Hall, 1995.

[9] J.-L. Lassez, M. J. Maher, and K. Marriott. Unification Revisited. In J. Minker, editor, *Foundations of Deductive Databases and Logic Programming*, pages 587–625. Morgan Kaufmann, Los Altos, Ca., 1988.

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

[11] Deyi Li and Dongbo Liu. *A fuzzy Prolog database system*. John Wiley & Sons, Inc., 1990.

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

[13] H.T. Nguyen and E.A. Walker. *A First Course in Fuzzy Logic*. Chapman & Hall/CRC, Boca Ratón, Florida, 2000.

[14] C. Palamidessi. Algebraic Properties of Idempotent Substitutions. In M.S. Paterson, editor, *Proc. of 17th Int'l Colloquium on Automata, Languages and Programming*, pages 386–399. Springer LNCS 443, 1990.

[15] A. Pettorossi and M. Proietti. Rules and Strategies for Transforming Functional and Logic Programs. *ACM Computing Surveys*, 28(2):360–414, 1996.

[16] B. Schweizer and A. Sklar. *Probabilistic Metric Spaces*. North-Holland, New York, 1983.

[17] H. Tamaki and T. Sato. Unfold/Fold Transformations of Logic Programs. In S. Tärnlund, editor, *Proc. of Second Int'l Conf. on Logic Programming*, pages 127–139, 1984.

[18] E. Trillas, C. del Campo, and S. Cubillo. When QM-Operators Are Implication Functions and Conditional Fuzzy Relations. *International Journal of Intelligent Systems*, 15:647–655, 2000.

[19] C. Vaucheret, S. Guadarrama, and S. Muñoz. Fuzzy prolog: A simple general implementation using $clp(r)$. In M. Baaz and A. Voronkov, editors, *Proc. of Logic for Programming, Artificial Intelligence and Reasoning (LPAR 2002), Tbilisi, Georgia*, pages 450–463. Lectures Notes in Artificial Intelligence 2514, Springer Verlag, 2002.

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

[21] P. Vojtas and L. Paulík. Soundness and completeness of non-classical extended SLD-resolution. In R. Dyckhoff et al, editor, *Proc. ELP'96 Leipzig*, pages 289–301. LNCS 1050, Springer Verlag, 1996.

[22] L. Zadeh. Calculus of fuzzy restrictions. *Fuzzy Sets and Their Applications to Cognitive and Decision Processes*, pages 1–39, 1975.

[23] L. A. Zadeh. Fuzzy Sets. *Information and Control*, 8(3):338–353, 1965.