Fuzzy Prolog with Default Knowledge

Claudio Vaucheret
Departamento de Ciencias de la Computación -Fa.E.A.
Universidad Nacional del Comahue
Buenos Aires 1400 - 8300 Neuquén - Argentina
Tel/Fax (54) (299) 4490312/3
e-mail: cvaucher@uncoma.edu.ar

Abstract

Incomplete information is a problem in many aspects of actual environments. Furthermore, in many scenarios the knowledge is not represented in a crisp way. It is common to find fuzzy concepts or problems with some level of uncertainty. This work extends the semantics and implementation of fuzzy prolog presented in [VGM02, GMV04] in order to include Default Knowledge capability. The new semantic allows non-uniform default assumptions and has Closed World Assumption (CWA) and Open World Assumption (OWA) as particular cases.

1 Introduction

In [VGM02, GMV04] we presented a definition of a Fuzzy Prolog Language that models $\mathcal{B}([0,1])$-valued Fuzzy Logic, and subsumes former approaches because it uses a truth value representation based on a union of sub-intervals on [0,1] and it is defined using general operators that can model different logics. We also presented the implementation of an interpreter for this conceived language using Constraint Logic Programming over Real numbers CLP($\mathcal{R}$)). It was straightforward to extend the implementation in order to include Default Knowledge. In this paper we adapt the formal semantics given including Default Knowledge.

An assumption defines default knowledge to be used to complete the available knowledge provided by the facts and rules of a program. For example, the Closed World Assumption (CWA) asserts that any atom whose truth-value cannot be inferred from the facts and rules is supposed to be false, on the other hand, the Open World Assumption (OWA) asserts that every such atom is supposed to be unknown or undefined.

2 Language

The following definitions describe the language presented in [VGM02]. Membership functions assign to each element of the universal set one element of the Borel Algebra over the interval

$[0,1]$. These sets are defined by functions of the form $A : X \to \mathcal{B}([0,1])$, where an element in $\mathcal{B}([0,1])$ is a countable union of sub-intervals of $[0,1]$.

The truth value of a goal will depend on the truth value of the subgoals which are in the body of the clauses of its definition. We use *aggregation operators* [ET99] in order to propagate the truth value by means of the fuzzy rules. Fuzzy sets *aggregation* is done using the application of a numeric operator of the form $f : [0,1]^n \to [0,1]$. If it verifies $f(0,\ldots,0) = 0$ and $f(1,\ldots,1) = 1$, and in addition it is monotonic and continuous, then it is called *aggregation operator*. If we deal with the definition of fuzzy sets as intervals it is necessary to generalize from *aggregation operators* of numbers to *aggregation operators* of intervals. Following the theorem proven by Nguyen and Walker in [NW00] to extend T-norms and T-conorms [KMP] to intervals.

Definition 2.1 (interval-aggregation) *Given an aggregation $f : [0,1]^n \to [0,1]$, an interval-aggregation $F : \mathcal{E}([0,1])^n \to \mathcal{E}([0,1])$ is defined as follows:*

$$F([x_1^l, x_1^u], ..., [x_n^l, x_n^u]) = [f(x_1^l, ..., x_n^l), f(x_1^u, ..., x_n^u)].$$

Actually, we work with union of intervals and propose the definition:

Definition 2.2 (union-aggregation) *Given an interval-aggregation $F : \mathcal{E}([0,1])^n \to \mathcal{E}([0,1])$ defined over intervals, a union-aggregation $\mathcal{F} : \mathcal{B}([0,1])^n \to \mathcal{B}([0,1])$ is defined over union of intervals as follows:*

$$\mathcal{F}(B_1,\ldots,B_n) = \cup\{F(\mathcal{E}_1, ..., \mathcal{E}_n) \mid \mathcal{E}_i \in B_i\}.$$

A constraint is a Σ-*formula* where Σ is a signature that contains the real numbers, the binary function symbols $+$ and $*$, and the binary predicate symbols $=$, $<$ and $\leq$. If the constraint c has solution in the domain of real numbers in the interval $[0,1]$ then c is *consistent*, and is denoted as $solvable(c)$.

The alphabet of our language consists of the following kinds of symbols: *variables*, *constants*, *function symbols* and *predicate symbols*. A *term* is defined inductively as follows:

1. A *variable* is a *term*.
2. A *constant* is a *term*.
3. if f is an n-ary *function symbol* and $t_1,\ldots,t_n$ are *terms*, then $f(t_1,\ldots,t_n)$ is a *term*.

If p is an n-ary *predicate symbol*, and $t_1,\ldots,t_n$ are *terms*, then $p(t_1,\ldots,t_n)$ is an *atomic formula* or, more simply an *atom*.

A *fuzzy program* is a finite set of *fuzzy facts*, and *fuzzy clauses* and we obtain information from the program through *fuzzy queries*. They are defined below:

Definition 2.3 (fuzzy fact) *If A is an atom,*

$$A \leftarrow v$$

is a fuzzy fact, *where v, a truth value, is an element in $\mathcal{B}([0,1])$ expressed as constraints over the domain $[0,1]$.*

Definition 2.4 (fuzzy clause) *Let* $A, B_1, \ldots, B_n$ *be atoms,*

$$A \leftarrow_F B_1, \ldots, B_n$$

is a fuzzy clause *where* F *is an interval-aggregation operator, which induces a union-aggregation, as by definition 2.2,* $\mathcal{F}$ *of truth values in* $\mathcal{B}([0,1])$ *represented as constraints over the domain* $[0,1]$.

Definition 2.5 (fuzzy query) *A* fuzzy query *is a tuple*

$$v \leftarrow A \ ?$$

where A *is an atom, and* v *is a variable (possibly instantiated) that represents a truth value in* $\mathcal{B}([0,1])$.

3 Semantics

3.1 Least Model Semantics

The *Herbrand Universe* U is the set of all ground *terms*, which can be made up with the constants and function symbols of a program, and the *Herbrand Base* B is the set of all ground atoms which can be formed by using the predicate symbols of the program with ground *terms* (of the *Herbrand Universe*) as arguments.

Definition 3.1 (default value) *We assume there is a function* default *which implement the Default Knowledge Assumptions. It assigns an element of* $\mathcal{B}([0,1])$ *to each element of the Herbrand Base. If the Closed World Assumption is used, then* $default(A) = [0,0]$ *for all* A *in Herbrand Base. If Open World Assumption is used instead,* $default(A) = [0,1]$ *for all* A *in Herbrand Base.*

Definition 3.2 (interpretation) *An* interpretation I *consists of the following:*

1. *a subset* B_I *of the* Herbrand Base,
2. *a mapping* V_I*, to assign*
   (a) *a truth value, in* $\mathcal{B}([0,1])$*, to each element of* B_I*, or*
   (b) $default(A)$*, if* A *does not belong to* B_I.

The Borel Algebra $\mathcal{B}([0,1])$ is a complete lattice under $\subseteq_{BI}$, that denotes Borel inclusion, and the Herbrand Base is a complete lattice under $\subseteq$, that denotes set inclusion, therefore a set of all *interpretations* forms a complete lattice under the relation $\sqsubseteq$ defined as follows.

Definition 3.3 (interval inclusion $\subseteq_{II}$) *Given two intervals* $I_1 = [a,b]$, $I_2 = [c,d]$ *in* $\mathcal{E}([0,1])$, $I_1 \subseteq_{II} I_2$ *if and only if* $c \leq a$ *and* $b \leq d$.

Definition 3.4 (Borel inclusion $\subseteq_{BI}$) *Given two unions of intervals $U = I_1 \cup \ldots \cup I_N$, $U' = I'_1 \cup \ldots \cup I'_M$ in $\mathcal{B}([0,1])$, $U \subseteq_{BI} U'$ if and only if $\forall I_i \in U$, $i \in 1..N$, $\exists I_{i1}, ..., I_{iL}$ intervals such that $I_{i1} \cup ... \cup I_{iL} = I_i$, $I_{i1} \cap ... \cap I_{iL} = \emptyset$ and for all $k \in 1..L$, $\exists I'_{jk} \in U'$. $I_{ik} \subseteq_{II} I'_{jk}$ where $jk \in 1..M$.*

Definition 3.5 (interpretation inclusion $\sqsubseteq$) *$I \sqsubseteq I'$ if and only if $B_I \subseteq B_{I'}$ and for all $B \in B_I$, $V_I(B) \subseteq_{BI} V_{I'}(B)$, where $I = \langle B_I, V_I \rangle$, $I' = \langle B_{I'}, V_{I'} \rangle$ are interpretations.*

Definition 3.6 (valuation) *A* valuation *σ of an atom A is an assignment of elements of U to variables of A. So $\sigma(A) \in B$ is a ground atom.*

Definition 3.7 (model) *Given an* interpretation *$I = \langle B_I, V_I \rangle$*

- *I is a* model *for a* fuzzy fact *$A \leftarrow v$, if for all valuation σ, $\sigma(A) \in B_I$ and $v \subseteq_{BI} V_I(\sigma(A))$.*
- *I is a* model *for a clause $A \leftarrow_F B_1, \ldots, B_n$ when the following holds: for all valuation σ, $\sigma(A) \in B_I$ and $v \subseteq_{BI} V_I(\sigma(A))$, where $v = \mathcal{F}(V_I(\sigma(B_1)), \ldots, V_I(\sigma(B_n)))$ and $\mathcal{F}$ is the union aggregation obtained from F.*
- *I is a* model *of a* fuzzy program*, if it is a* model *for the facts and clauses of the program.*

Every program has a least model which is usually regarded as the intended interpretation of the program since it is the most conservative model. Let $\cap$ be the meet operator on the lattice of interpretations $(I, \sqsubseteq)$, then we can prove the following result.

Theorema 3.1 (model intersection property) *Let $I_1 = \langle B_{I_1}, V_{I_1} \rangle$,$I_2 = \langle B_{I_1}, V_{I_1} \rangle$ be models of a fuzzy program P. Then $I_1 \cap I_2$ is a model of P.*

Proof. Let $M = \langle B_M, V_M \rangle = I_1 \cap I_2$. Since I_1 and I_2 are models of P, they are models for each fact and clause of P. Then for all valuation σ we have

- for all fact $A \leftarrow v$ in P,
  - $\sigma(A) \subseteq B_{I_1}$ and $\sigma(A) \in B_{I_2}$ then $\sigma(A) \in B_{I_1} \cap B_{I_2} = B_M$,
  - $v \subseteq_{BI} V_{I_1}(\sigma(A))$ and $v \subseteq_{BI} V_{I_2}(\sigma(A))$, then $v \subseteq_{BI} V_{I_1}(\sigma(A)) \cap V_{I_2}(\sigma(A)) = V_M(\sigma(A))$

  therefore M is a model for $A \leftarrow v$
- and for all clause $A \leftarrow_F B_1, \ldots, B_n$ in P
  - since $\sigma(A) \in B_{I_1}$ and $\sigma(A) \in B_{I_2}$, then $\sigma(A) \in B_{I_1} \cap B_{I_2} = B_M$.
  - if $v = \mathcal{F}(V_M(\sigma(B_1)), \ldots, V_M(\sigma(B_n)))$, since F is monotonic, $v \subseteq_{BI} V_{I_1}(\sigma(A))$ and $v \subseteq_{BI} V_{I_2}(\sigma(A))$, then $v \subseteq_{BI} V_{I_1}(\sigma(A)) \cap V_{I_2}(\sigma(A)) = V_M(\sigma(A))$

  therefore M is a model for $A \leftarrow_F B_1, \ldots, B_n$

and M is model of P.

Remark 3.1 (Least model semantic) *If we let* $\mathbf{M}$ *be the set of all models of a program* P*, the intersection of all of this models,* $\bigcap \mathbf{M}$*, is a model and it is the least model of* P*. We denote the least model of a program* P *by* $lm(P)$*.*

Example 3.1 *Let's see an example. Suppose we have the following program* P*:*

$tall(peter) \leftarrow [0.6, 0.7] \vee 0.8$
$tall(john) \leftarrow 0.7$
$swift(john) \leftarrow [0.6, 0.8]$
$good\_player(X) \leftarrow_{luka} tall(X), swift(X)$

Here, we have two facts, $tall(john)$ *and* $swift(john)$ *whose truth values are the unitary interval* $[0.7, 0.7]$ *and the interval* $[0.6, 0.8]$*, respectively, and a clause for the* $good\_player$ *predicate whose* aggregation operator *is the Lukasiewicz T-norm.*

The following interpretation $I = \langle B, V \rangle$ *is a model for* P*, where*
$B = \{tall(john), tall(peter), swift(john),$
$good\_player(john), good\_player(peter)\}$ *and*

$$\begin{aligned} V(tall(john)) &= [0.7, 1] \\ V(swift(john)) &= [0.5, 0.8] \\ V(tall(peter)) &= [0.6, 0.7] \vee [0.8, 0.8] \\ V(good\_player(john)) &= [0.2, 0.9] \\ V(good\_player(peter)) &= [0.5, 0.9] \end{aligned}$$

note that for instance if $V(good\_player(john)) = [0.2, 0.5]$ $I = \langle B, V \rangle$ *cannot be a model of* P*, the reason is that* $v = luka([0.7, 1], [0.5, 0.8]) = [0.7+0.5-1, 1+0.8-1] = [0.2, 0.8] \not\subseteq_{II} [0.2, 0.5]$.

The least model of P *is the intersection of all models of* P *which is* $M = \langle B_M, V_M \rangle$ *where*
$B_M = \{tall(john), tall(peter), swift(john),$
$good\_player(john)\}$ *and*

$$\begin{aligned} V_M(tall(john)) &= [0.7, 0.7] \\ V_M(swift(john)) &= [0.6, 0.8] \\ V_M(tall(peter)) &= [0.6, 0.7] \vee [0.8, 0.8] \\ V_M(good\_player(john)) &= [0.3, 0.5] \end{aligned}$$

Now, suppose we add to P *that* $default(swift(peter)) = [0.5, 1]$*. In this case* $V(swift(peter)) = [0.5, 1]$ *and* I *is not a model for* P *because* $v = luka([0.6, 0.7] \vee [0.8, 0.8], [0.5, 1]) = [0.6 + 0.5 - 1, 0.7 + 1 - 1] \vee [0.8 + 0.5 - 1, 0.8 + 1 - 1] = [0.1, 0.7] \vee [0.3, 0.8] \not\subseteq_{II} [0.5, 0.9]$.

If we add to P *that* $default(swift(peter)) = [0.5, 1]$ *then the least model of* P *is*
$M = \langle B_M, V_M \rangle$ *where*
$B_M = \{tall(john), tall(peter), swift(john),$

$good\_player(john), good\_player(peter)\}$ *and*

$$\begin{aligned} V_M(tall(john)) &= [0.7, 0.7] \\ V_M(swift(john)) &= [0.6, 0.8] \\ V_M(tall(peter)) &= [0.6, 0.7] \vee [0.8, 0.8] \\ V_M(good\_player(john)) &= [0.3, 0.5] \\ V_M(good\_player(peter)) &= [0.1, 0.7] \vee [0.3, 0.8] \end{aligned}$$

3.2 Fixed-Point Semantics

The fixed-point semantics we present is based on a one-step consequence operator T_P. The least fixed-point $lfp(T_P) = I$ (i.e. $T_P(I) = I$) is the declarative meaning of the program P, so is equal to $lm(P)$.

Let P be a fuzzy program and B_P the Herbrand base of P; then the *mapping* T_P over *interpretations* is defined as follows:

Let $I = \langle B_I, V_I \rangle$ be a fuzzy *interpretation*, then $T_P(I) = I'$, $I' = \langle B_{I'}, V_{I'} \rangle$

$$B_{I'} = \{A \in B_P \mid Cond\}$$
$$V_{I'}(A) = \bigcup \{v \in \mathcal{B}([0,1]) \mid Cond\}$$

where $Cond = (A \leftarrow v$ is a ground instance of a fact in P and $solvable(v))$ or $(A \leftarrow_F A_1, \ldots, A_n$ is a ground instance of a clause in P, and $solvable(v), v = \mathcal{F}(V_I(A_1), \ldots, V_I(A_n)))$. Note that since I' must be an interpretation, $V_{I'}(A) = default(A)$ for all $A \notin B_{I'}$.

The set of interpretations forms a complete lattice so that, T_P it is continuous. Recall the definition of the *ordinal powers* of a function G over a complete lattice X:

$$G \uparrow \alpha = \begin{cases} \bigcup \{G \uparrow \alpha' \mid \alpha' < \alpha\} \\ \qquad \text{if } \alpha \text{ is a limit ordinal,} \\ G(G \uparrow (\alpha - 1)) \\ \qquad \text{if } \alpha \text{ is a successor ordinal,} \end{cases} \quad \text{and dually,} \quad G \downarrow \alpha = \begin{cases} \bigcap \{G \downarrow \alpha' \mid \alpha' < \alpha\} \\ \qquad \text{if } \alpha \text{ is a limit ordinal,} \\ G(G \downarrow (\alpha - 1)) \\ \qquad \text{if } \alpha \text{ is a successor ordinal,} \end{cases}$$

Since the first limit ordinal is 0, it follows that in particular, $G \uparrow 0 = \bot_X$ (the bottom element of the lattice X) and $G \downarrow 0 = \top_X$ (the top element). From Kleene's fixed point theorem we know that the least fixed-point of any continuous operator is reached at the first infinite ordinal ω. Hence $lfp(T_P) = T_P \uparrow \omega$.

Example 3.2 *Consider the same program P of the example 3.1, with $default(swift(peter)) = [0.5, 1]$, the ordinal powers of T_P are*
$T_P \uparrow 0 = \{\}$
$T_P \uparrow 1 = \{tall(john), swift(john),$
$tall(peter)\}$ *and*

$$\begin{aligned} V(tall(john)) &= [0.7, 0.7] \\ V(swift(john)) &= [0.6, 0.8] \\ V(tall(peter)) &= [0.6, 0.7] \vee [0.8, 0.8] \end{aligned}$$

Since $swift(peter)$ *does not belong to* $B_{T_P \uparrow 1}$,
$V_{T_P \uparrow 1}(swift(peter) = default(swift(peter) = [0.5, 1]$ *then*

$T_P \uparrow 2 = \{tall(john), swift(john), tall(peter),$
$good\_player(john), good\_player(peter)\}$ *and*

$$\begin{array}{lcl} V(tall(john)) & = & [0.7, 0.7] \\ V(swift(john)) & = & [0.6, 0.8] \\ V(tall(peter)) & = & [0.6, 0.7] \vee [0.8, 0.8] \\ V(good\_player(john)) & = & [0.3, 0.5] \\ V_M(good\_player(peter)) & = & [0.1, 0.7] \vee [0.3, 0.8] \end{array}$$

$T_P \uparrow 3 = T_P \uparrow 2$.

Lemma 3.1 *Let* P *a fuzzy program,* M *is a model of* P *if and only if* M *is a prefixpoint of* T_P*, that is* $T_P(M) \sqsubseteq M$.

Proof. Let $M = \langle B_M, V_M \rangle$ and $T_P(M) = \langle B_{T_P}, V_{T_P} \rangle$.

We first prove the "if" direction. Let A be an element of Herbrand Base, if $A \in B_{T_P}$, then by definition of T_P there exists a ground instance of a fact of P, $A \leftarrow v$, or a ground instance of a clause of P, $A \leftarrow_F A_1, \ldots, A_n$ where $\{A_1, \ldots, A_n\} \subseteq B_M$ and $v = \mathcal{F}(V_M(A_1), \ldots, V_M(A_n))$. Since M is a model of P, $A \in B_M$, and each $v \subseteq_{BI} V_M(A)$, then $V_{T_P}(A) \subseteq_{BI} V_M(A)$ and then $T_P(M) \sqsubseteq M$. $\square$. If $A \notin B_{T_P}$ then $V_{T_P}(A) = default(A) \subseteq_{BI} V_M(A)$.

Analogously, for the "only if" direction, for each ground instance $v = \mathcal{F}(V_M(A_1), \ldots, V_M(A_n))$, $A \in B_{T_P}$ and $v \subseteq_{BI} V_{T_P}(A)$, but as $T_P(M) \subseteq M$, $B_{T_P} \subseteq B_M$ and $V_{T_P}(A) \subseteq_{BI} V_M(A)$. Then $A \in B_M$ and $v \subseteq_{BI} V_M(A)$ therefore M is a model of P. $\square$

Given this relationship, it is straightforward to prove that the least model of a program P is also the least fixed-point of T_P.

Theorema 3.2 *Let* P *be a fuzzy program,* $lm(P) = lfp(T_P)$.

Proof.

$$\begin{array}{rcl} lm(P) & = & \bigcap\{M \mid M \text{ is a model of } P\} \\ & = & \bigcap\{M \mid M \text{ is a pre-fixpoint of } P\} \quad \text{from lemma 3.1} \\ & = & lfp(T_P) \quad \text{by the Knaster-Tarski Fixpoint Theorem [Tar55]}\square \end{array}$$

3.3 Operational Semantics

The procedural semantics is interpreted as a sequence of transitions between different states of a system. We represent the state of a *transition system* in a computation as a tuple $\langle A, \sigma, S \rangle$ where A is the goal, σ is a substitution representing the instantiation of variables needed to get

to this state from the initial one and S is a constraint that represents the truth value of the goal at this state.

When computation starts, A is the initial goal, $\sigma = \emptyset$ and S is true (if there are neither previous instantiations nor initial constraints). When we get to a state where the first argument is empty then we have finished the computation and the other two arguments represent the answer.

A *transition* in the *transition system* is defined as:

1. $\langle A \cup a, \sigma, S\rangle \rightarrow \langle A\theta, \sigma \cdot \theta, S \wedge \mu_a = v\rangle$

   if $h \leftarrow v$ is a fact of the program P, θ is the mgu of a and h, μ_a is the truth value for a and $solvable(S \wedge \mu_a = v)$.

2. $\langle A \cup a, \sigma, S\rangle \rightarrow \langle (A \cup B)\theta, \sigma \cdot \theta, S \wedge c\rangle$

   if $h \leftarrow_F B$ is a rule of the program P, θ is the mgu of a and h, c is the constraint that represents the truth value obtained applying the union-aggregation $\mathcal{F}$ to the truth values of B, and $solvable(S \wedge c)$.

3. $\langle A \cup a, \sigma, S\rangle \rightarrow \langle A, \sigma, S \wedge \mu_a = v\rangle$

   if none of the above are applicable and $solvable(S \wedge \mu_a = v)$ where $\mu_a = default(a)$.

The success set $SS(P)$ collects the answers to simple goals $p(\widehat{x})$. It is defined as follows:

$$SS(P) = \langle B, V\rangle$$

where $B = \{p(\widehat{x})\sigma | \langle p(\widehat{x}), \emptyset, true\rangle \rightarrow^* \langle \emptyset, \sigma, S\rangle\}$ is the set of elements of the Herbrand Base that are instantiated and that have succeeded; and $V(p(\widehat{x})) = \cup\{v | \langle p(\widehat{x}), \emptyset, true\rangle \rightarrow^* \langle \emptyset, \sigma, S\rangle$, and v is the solution of $S\}$ is the set of truth values of the elements of B that is the union (got by backtracking) of truth values that are obtained from the set of constraints provided by the program P while query $p(\widehat{x})$ is computed.

Example 3.3 *Let P be the program of example 3.1. Consider the fuzzy goal*

$$\mu \leftarrow good\_player(X) \quad ?$$

the first transition *in the computation is*

$$\langle \{(good\_player(X)\}, \epsilon, true\rangle \rightarrow$$
$$\langle \{tall(X), swift(X)\}, \epsilon,$$
$$\mu = max(0, \mu_{tall} + \mu_{swift} - 1)\rangle$$

unifying the goal with the clause and adding the constraint corresponding to Lukasiewicz T-norm. The next transition *leads to the state:*

$$\langle \{swift(X)\}, \{X = john\}, \mu = max(0, \mu_{tall} + \mu_{swift} - 1) \wedge \mu_{tall} = 0.7\rangle$$

after unifying $tall(X)$ *with* $tall(john)$ *and adding the constraint regarding the truth value of the fact. The computation ends with:*

$$\langle\{\}, \{X = john\}, \mu = max(0, \mu_{tall} + \mu_{swift} - 1) \wedge \mu_{tall} = 0.7 \wedge 0.6 \leq \mu_{swift} \wedge\ \mu_{swift} \leq 0.8\rangle$$

As $\mu = max(0, \mu_{tall} + \mu_{swift} - 1) \wedge \mu_{tall} = 0.7 \wedge 0.6 \leq \mu_{swift} \wedge \mu_{swift} \leq 0.8$ *entails* $\mu \in [0.3, 0.5]$, *the answer to the query* $good\_player(X)$ *is* $X = john$ *with truth value the interval* $[0.3, 0.5]$.

Other sequences of transitions are:

1. $\langle\{(good\_player(X)\}, \epsilon, true\rangle \rightarrow$
   $\langle\{tall(X), swift(X)\}, \epsilon,$
   $\mu = max(0, \mu_{tall} + \mu_{swift} - 1)\rangle \rightarrow$
   $\langle\{swift(X)\}, \{X = peter\},$
   $\mu = max(0, \mu_{tall} + \mu_{swift} - 1) \wedge 0.6 \leq \mu_{tall} \wedge\ \mu_{tall} \leq 0.7\rangle \rightarrow$
   $\langle\{swift(X)\}, \{X = peter\}, \mu = max(0, \mu_{tall} + \mu_{swift} - 1) \wedge$
   $0.6 \leq \mu_{tall} \wedge\ \mu_{tall} \leq 0.7 \wedge 0.5 \leq \mu_{swift} \wedge\ \mu_{swift} \leq 1\rangle$
   As $\mu = max(0, \mu_{tall} + \mu_{swift} - 1) \wedge 0.6 \leq \mu_{tall} \wedge \mu_{tall} \leq 0.8 \wedge 0.5 \leq \mu_{swift} \wedge \mu_{swift} \leq 1$ *entails* $\mu \in [0.1, 0.7]$, *the answer to the query* $good\_player(X)$ *is* $X = peter$ *with truth value the interval* $[0.1, 0.7]$.

2. $\langle\{(good\_player(X)\}, \epsilon, true\rangle \rightarrow$
   $\langle\{tall(X), swift(X)\}, \epsilon,$
   $\mu = max(0, \mu_{tall} + \mu_{swift} - 1)\rangle \rightarrow$
   $\langle\{swift(X)\}, \{X = peter\},$
   $\mu = max(0, \mu_{tall} + \mu_{swift} - 1) \wedge \mu_{tall} = 0.8\rangle \rightarrow$
   $\langle\{swift(X)\}, \{X = peter\}, \mu = max(0, \mu_{tall} + \mu_{swift} - 1) \wedge$
   $\mu_{tall} = 0.8 \wedge\ 0.5 \leq \mu_{swift} \wedge \mu_{swift} \leq 1\rangle$
   As $\mu = max(0, \mu_{tall} + \mu_{swift} - 1) \wedge \mu_{tall} = 0.8 \wedge 0.5 \leq \mu_{swift} \wedge \mu_{swift} \leq 1$ *entails* $\mu \in [0.3, 0.8]$, *the answer to the query* $good\_player(X)$ *is* $X = peter$ *with truth value the interval* $[0.3, 0.8]$.

In order to prove the equivalence between operational semantic and fixed-point semantic, it is useful to introduce a type of canonical top-down evaluation strategy. In this strategy all literals are reduced at each step in a derivation. For obvious reasons, such a derivation is called *breadth-first*.

Definition 3.8 (Breadth-first transition) *Given the following set of valid transitions:*

$$\begin{array}{l}
\langle\{\{A_1, \ldots, A_n\}, \sigma, S\rangle \rightarrow \langle\{\{A_2, \ldots, A_n\} \cup B_1, \sigma \cdot \theta_1, S \wedge c_1\rangle \\
\langle\{\{A_1, \ldots, A_n\}, \sigma, S\rangle \rightarrow \langle\{\{A_1, A_3 \ldots, A_n\} \cup B_2, \sigma \cdot \theta_2, S \wedge c_2\rangle \\
\qquad\qquad\qquad\qquad \vdots \\
\langle\{\{A_1, \ldots, A_n\}, \sigma, S\rangle \rightarrow \langle\{\{A_1, \ldots, A_{n-1}\} \cup B_n, \sigma \cdot \theta_n, S \wedge c_n\rangle
\end{array}$$

a breadth-first transition *is defined as*

$\langle\{A_1,\ldots,A_n\},\sigma,S\rangle \rightarrow_{BF} \langle B_1\cup\ldots\cup B_n,\sigma\cdot\theta_1\cdot\ldots\cdot\theta_n,S\wedge c_1\wedge\ldots\wedge c_n\rangle$

in which all literals are reduced at one step.

Theorema 3.3 *Given a ordinal number n and $T_P\uparrow n=\langle B_{T_{P_n}},V_{T_{P_n}}\rangle$. there is a successful breadth-first derivation of lengh less or equal to $n+1$ for a program P, $\langle\{A_1,\ldots,A_k\},\sigma,S_1\rangle\rightarrow^*_{BF}$ $\langle\emptyset,\theta,S_2\rangle$ iff $A_i\theta\in B_{T_{P_n}}$ and $solvable(S\wedge\mu_{A_i}=v_i)$ and $v_i\subseteq_{BI} V_{T_{P_n}}(A_i\theta)$.*

Proof. The proof is by induction on n. For the base case, all the literals are reduced using the first type of transitions or the last one, that is, for each literal A_i, it exits a fact $h_i\leftarrow v_i$ such that θ_i is the mgu of A_i and h_i, and μ_{A_i} is the truth variable for A_i, and $solvable(S_1\wedge\mu_{A_i}=v_i)$ or $\mu_{A_i}=default(A_i)$. By definition of T_P, each $v_i\subseteq_{BI} V_{T_{P_1}}(A_i\theta)$ where $\langle B_{T_{P_1}},V_{T_{P_1}}\rangle=T_P\uparrow 1$.

For the general case, consider the successful derivation,

$\langle\{A_1,\ldots,A_k\},\sigma_1,S_1\rangle\rightarrow_{BF}\langle B,\sigma_2,S_2\rangle\rightarrow_{BF}\ldots\rightarrow_{BF}\langle\emptyset,\sigma_n,S_n\rangle$

the transition $\langle\{A_1,\ldots,A_k\},\sigma_1,S_1\rangle\rightarrow_{BF}\langle B,\sigma_2,S_2\rangle$

When a literal A_i is reduced using a fact or there is not rule for A_i the result is the same as in the base case, otherwise there is a clause $h_i\leftarrow_F B_{1_i},\ldots,B_{m_i}$ in P such that θ_i is the mgu of A_i and $h_i\in B\sigma_2$ and $B_{j_i}\theta_i\in B\sigma_2$, by the induction hypothesis $B\sigma_2\subseteq B_{T_{P_{n-1}}}$ and $solvable(S_2\wedge\mu_{B_{j_i}}=v_{j_i})$ and $v_{j_i}\subseteq_{BI} V_{T_{P_{n-1}}}(B_{j_i}\sigma_2)$ then $B_{j_i}\theta_i\subseteq B_{T_{P_{n-1}}}$ and by definition of T_P, $A_i\theta_i\in B_{T_{P_n}}$ and $solvable(S_1\wedge\mu_{A_i}=v_i)$ and $v_i=\subseteq_{BI} V_{T_{P_n}}(A_i\sigma_1)$. □

Theorema 3.4 *For a program P there is a successful derivation*

$$\langle p(\widehat{x}),\emptyset,true\rangle\rightarrow^*\langle\emptyset,\sigma,S\rangle$$

iff $p(\widehat{x})\sigma\in B$ and v is the solution of S and $v\subseteq_{BI} V(p(\widehat{x})\sigma)$ where $lfp(T_P)=\langle B,V\rangle$

Proof. It follows from the fact that $lfp(T_P)=T_P\uparrow\omega$ and from the Theorem 3.3. □

Theorema 3.5 *For a fuzzy program P the three semantics are equivalent, i.e.*

$$SS(P)=lfp(TP)=lm(P)$$

Proof. the first equivalence follows from Theorem 3.4 and the second from Theorem 3.2. □

4 Implementation and Syntax

4.1 CLP($\mathcal{R}$)

Constraint Logic Programming [JL87] began as a natural merging of two declarative paradigms: constraint solving and logic programming. This combination helps make CLP programs both expressive and flexible, and in some cases, more efficient than other kinds of logic programs. CLP($\mathcal{R}$) [JMSY92] has linear arithmetic constraints and computes over the real numbers.

Fuzzy Prolog was implemented in [GMV04] as a syntactic extension of a CLP($\mathcal{R}$) system. CLP($\mathcal{R}$) was incorporated as a library [CH00] in the Ciao Prolog system [HBC+99].

The *fuzzy* library (or *package* in the Ciao Prolog terminology) which implements the interpreter of our fuzzy Prolog language has been modified to handle default reasoning.

4.2 Syntax

Each fuzzy Prolog clause has an additional argument in the head which represents its truth value in terms of the truth values of the subgoals of the body of the clause. A fact $A \leftarrow v$ is represented by a Fuzzy Prolog fact that describes the range of values of v with a union of intervals (that can be only an interval or even a real number in particular cases). The following examples illustrate the concrete syntax of programs:

$youth(45) \leftarrow$ $\quad [0.2, 0.5] \bigcup [0.8, 1]$	`youth(45) :~` `  [0.2,0.5] v [0.8,1].`
$tall(john) \leftarrow 0.7$	`tall(john) :~ 0.7.`
$swift(john) \leftarrow$ $\quad [0.6, 0.8]$	`tall(john) :~` `  [0.6,0.8].`

$good\_player(X) \leftarrow_{min}$
$\quad tall(X),$
$\quad swift(X)$
	`good_player(X) :~ min`
	`  tall(X),`
	`  swift(X).`

These clauses are expanded at compilation time to constrained clauses that are managed by CLP($\mathcal{R}$) at run-time. Predicates . = ./2, . < ./2, . <= ./2, . > ./2 and . >= ./2 are the Ciao CLP($\mathcal{R}$) operators for representing constraint inequalities. For example the first fuzzy fact is expanded to these Prolog clauses with constraints

```
youth(45,V):-   V .>=. 0.2,
                V .<=. 0.5.
youth(45,V):-   V .>=. 0.8,
                V .<.  1.
```

And the fuzzy clause

```
    :- default(good_layer/1,[0.5,0.7]).
  good_player(X) :~ min tall(X),swift(X).
```

is expanded to

```
good_player(X,Vp) :-
           tall(X,Vq),
           swift(X,Vr),
           minim([Vq,Vr],Vp),
           Vp .>=. 0, Vp .=<. 1.
good_player(X,Vp) :-
           Vp .>=. 0.5, Vp .=<.0.7.
```

The predicate `minim/2` is included as run-time code by the library. Its function is adding constraints to the truth value variables in order to implement the T-norm *min*. We have implemented several *aggregation operators* as `prod`, `max`, `luka`, etc. and in a similar way any other operator can be added to the system without any effort. The system is extensible by the user simply adding the code for new *aggregation operators* to the library.

5 Conclusion

We have presented different semantics of our fuzzy language, and it is proved the equivalence between them. These semantics support non-uniform default assumptions extending the formalization given in [GMV04]. The Ciao system including our Fuzzy Prolog implementation can be downloaded from http://www.clip.dia.fi.upm.es/Software/Ciao.

References

[CH00] D. Cabeza and M. Hermenegildo. A New Module System for Prolog. In *CL2000*, number 1861 in LNAI, pages 131–148. Springer-Verlag, July 2000.

[ET99] S. Cubillo E. Trillas, A. Pradera. *A mathematical model for fuzzy connectives and its application to operators behavioural study*, volume 516, chapter 4, pages 307–318. Kluwer Academic Publishers (Series: The Kluwer International Series in Engineering and Computer Sciences), 1999.

[GMV04] S. Guadarrama, S. Muoz, and C. Vaucheret. Fuzzy prolog: A new approach using soft constraints propagation. *Fuzzy Sets and Systems, FSS*, 144(1):127–150, 2004. ISSN 0165-0114.

[HBC+99] M. Hermenegildo, F. Bueno, D. Cabeza, M. Carro, M. García de la Banda, P. López-García, and G. Puebla. The CIAO Multi-Dialect Compiler and System: An Experimentation Workbench for Future (C)LP Systems. In *Parallelism and Implementation of Logic and Constraint Logic Programming*, pages 65–85. Nova Science, Commack, NY, USA, April 1999.

[JL87] J. Jaffar and J. L. Lassez. Constraint Logic Programming. In *ACM Symp. Principles of Programming Languages*, pages 111–119. ACM, 1987.

[JMSY92] J. Jaffar, S. Michaylov, P. J. Stuckey, and R. H. C. Yap. The clp(r) language and system. *ACM Transactions on Programming Languages and Systems*, 14(3):339–395, 1992.

[KMP] E.P. Klement, R. Mesiar, and E. Pap. Triangular norms. Kluwer Academic Publishers.

[NW00] H. T. Nguyen and E. A. Walker. *A first Course in Fuzzy Logic.* Chapman & Hall/Crc, 2000.

[Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[VGM02] C. Vaucheret, S. Guadarrama, and S. Muoz. Fuzzy prolog: A simple general implementation using *clp(r)*. In M. Baaz and A. Voronkov, editors, *Logic for Programming, Artificial Intelligence, and Reasoning, LPAR 2002*, number 2514 in LNAI, pages 450–463, Tbilisi, Georgia, October 2002. Springer.