Hybrid Probabilistic Logic Programs as Residuated Logic Programs

Carlos Viegas Damásio and Luís Moniz Pereira

Centro de Inteligência Artificial (CENTRIA)
Departamento de Informática, Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa
2825-114 Caparica, Portugal. E-mail: cd|lmp@di.fct.unl.pt

Abstract. In this paper we show the embedding of Hybrid Probabilistic Logic Programs into the rather general framework of Residuated Logic Programs, where the main results of (definite) logic programming are validly extrapolated, namely the extension of the immediate consequences operator of van Emden and Kowalski. The importance of this result is that for the first time a framework encompassing several quite distinct logic programming semantics is described, namely Generalized Annotated Logic Programs, Fuzzy Logic Programming, Hybrid Probabilistic Logic Programs, and Possibilistic Logic Programming. Moreover, the embedding provides a more general semantical structure paving the way for defining paraconsistent probabilistic reasoning logic programming semantics.

1 Introduction

The literature on logic programming theory is brimming with proposals of languages and semantics for extensions of definite logic programs (e.g. [7, 15, 4, 10]), i.e. without non-monotonic or default negation. Usually, the authors characterize their programs with a model theoretic semantics, where a minimum model is guaranteed to exist, and a corresponding monotonic fixpoint operator (continuous or not). In many cases these semantics are many-valued.

In this paper we start by defining a rather general framework of Residuated Logic Programs. We were inspired by the deep theoretical results of many-valued logics and fuzzy logic (see [1, 9] for excellent accounts) and applied these ideas to logic programming. In fact, a preliminary work in this direction is [15], but the authors restrict themselves to a linearly ordered set of truth-values (the real closed interval $[0, 1]$) and to a very limited syntax: the head of rules is a literal and the body is a multiplication (t-norm) of literals. Our main semantical structures are residuated (or residual) lattices (c.f. [1, 9]), where a generalized modus ponens rule is defined. This characterizes the essence of logic programming: from the truth-value of bodies for rules for an atom we can determine the truth-value of that atom, depending on the confidence in the rules.

Besides fuzzy reasoning, probabilistic reasoning forms are essential for knowledge representation in real-world applications. However, a major difficulty is

that there are several logical ways of determining the probabilities of complex events (conjunctions or disjunctions) from primitive ones. To address this issue, a model theory, fixpoint theory and proof theory for hybrid probabilistic logic programs were recently introduced [4, 3]. The generality of Residuated Logic Programming is illustrated in practice by presenting an embedding of Hybrid Probabilistc Logic Programs [4, 3] into our framework.

Our paper proceeds as follows. In the next section we present the residuated logic programs. Afterwards, we overview the hybrid probabilistic logic programming setting and subsequently provide the embedding. We finally draw some conclusions and point out future directions. We included the main proofs for the sake of completeness.

2 Residuated Logic Programs

The theoretical foundations of logic programming were clearly established in [11, 14] for definite logic programs (see also [12]), i.e. programs made up of rules of the form $A_0 \subset A_1 \wedge \ldots \wedge A_n (n \geq 0)$ where each $A_i (0 \leq i \leq n)$ is a propositional symbol (an atom), $\subset$ is classical implication, and $\wedge$ the usual Boolean conjunction[1]. In this section we generalize the language and semantics of definite logic programs in order to encompass more complex bodies and heads and, evidently, multi-valued logics. For simplicity, we consider only the propositional (ground) case.

In general, a logic programming semantics requires a notion of consequence (implication) which satisfies a generalization of *Modus Ponens* to a multi-valued setting. The generalization of *Modus Ponens* to multi-valued logics is very well understood, namely in Fuzzy Propositional Logics [13, 1, 9]. Since one of our initial goals was to capture Fuzzy Logic Programming [6, 15], it was natural to adopt as semantical basis the *residuated lattices* (see [5, 1]). This section summarizes the results fully presented and proved in [2]. We first require some definitions.

Definition 1 (Adjoint pair). *Let $< P, \preceq_P >$ be a partially ordered set and $(\leftarrow, \otimes)$ a pair of binary operations in P such that:*

(a_1) *Operation $\otimes$ is isotonic, i.e. if $x_1, x_2, y \in P$ such that $x_1 \preceq_P x_2$ then $(x_1 \otimes y) \preceq_P (x_2 \otimes y)$ and $(y \otimes x_1) \preceq_P (y \otimes x_2)$;*

(a_2) *Operation $\leftarrow$ is isotonic in the first argument (the consequent) and antitonic in the second argument (the antecedent), i.e. if $x_1, x_2, y \in P$ such that $x_1 \preceq_P x_2$ then $(x_1 \leftarrow y) \preceq_P (x_2 \leftarrow y)$ and $(y \leftarrow x_2) \preceq_P (y \leftarrow x_1)$;*

(a_3) *For any $x, y, z \in P$, we have that $x \preceq_P (y \leftarrow z)$ holds if and only if $(x \otimes z) \preceq_P y$ holds.*

Then we say that $(\leftarrow, \otimes)$ forms an adjoint pair in $< P, \preceq_P >$.

The intuition of the two above properties is immediate, the third one may be more difficult to grasp. In one direction, it is simply asserting that the following *Fuzzy Modus Ponens* rule is valid (cf. [9]):

[1] We remove the parentheses to simplify the reading of the rule.

If x is a lower bound of $\psi \leftarrow \varphi$, and z is a lower bound of φ then a lower bound y of ψ is $x \otimes z$.

The other direction is ensuring that the truth-value of $y \leftarrow x$ is the maximal z satisfying $x \otimes z \preceq_P y$.

Besides (a_1)–(a_3) it is necessary to impose extra conditions on the multiplication operation $(\otimes)$, namely associativity, commutativity and existence of a unit element. It is also indispensable to assume the existence of a bottom element in the lattice of truth-values (the zero element). Formally:

Definition 2 (Residuated Lattice). *Consider the lattice $< L, \preceq_L >$. We say that $(L^{\preceq}, \leftarrow, \otimes)$ is a residuated lattice whenever the following three conditions are met:*

(l_1) *$< L, \preceq_L >$ is a bounded lattice, i.e. it has bottom $(\bot)$ and top $(\top)$ elements;*
(l_2) *$(\leftarrow, \otimes)$ is an adjoint pair in $< L, \preceq_L >$;*
(l_3) *$(L, \otimes, \top)$ is a commutative monoid.*

We say that the residuated lattice is complete whenever $< L, \preceq_L >$ is complete. In this case, condition (l_1) is immediately satisfied.

Our main semantical structure is a residuated algebra, an algebra where a multiplication operation is defined, the corresponding residuum operation (or implication), and a constant representing the top element of the lattice of truth-values (whose set is the carrier of the algebra). They must define a complete residuated lattice, since we intend to deal with infinite programs (theories). Obviously, a residuated algebra may have additional operators. Formally:

Definition 3 (Residuated Algebra). *Consider a algebra $\mathfrak{R}$ defining operators $\leftarrow, \otimes$ and $\top$ on carrier set $\mathcal{T}_{\mathfrak{R}}$ such that $\preceq$ is a partial order on $\mathcal{T}_{\mathfrak{R}}$. We say that $\mathfrak{R}$ is a residuated algebra with respect to $(\leftarrow, \otimes)$ if $(\mathcal{T}_{\mathfrak{R}}^{\preceq}, \leftarrow, \otimes)$ is a complete residuated lattice. Furthermore, operator $\top$ is a constant mapped to the top element of $\mathcal{T}_{\mathfrak{R}}$.*

Our Residuated Logic Programs will be constructed from the abstract syntax induced by a residuated algebra and a set of propositional symbols. The way of relating syntax and semantics in such algebraic setting is well-known and we refer to [8] for more details.

Definition 4 (Residuated Logic Programs). *Let $\mathfrak{R}$ be a residuated algebra with respect to $(\leftarrow, \otimes, \top)$. Let Π be a set of propositional symbols and the corresponding algebra of formulae $\mathfrak{F}$ freely generated from Π. A residuated logic program is a set of weighted rules of the form $\langle (A \leftarrow \Psi), \vartheta \rangle$ such that:*

1. *The rule $(A \leftarrow \Psi)$ is a formula of $\mathfrak{F}$;*
2. *The confidence factor ϑ is a truth-value of $\mathfrak{R}$ belonging to $\mathcal{T}_{\mathfrak{R}}$;*
3. *The head of the rule A is a propositional symbol of Π.*
4. *The body formula Ψ corresponds to an isotonic function with propositional symbols $B_1, \ldots, B_n$ $(n \geq 0)$ as arguments.*

To simplify the notation, we represent the above pair as $A \overset{\vartheta}{\leftarrow} \Psi[B_1, \ldots, B_n]$, *where* $B_1, \ldots, B_n$ *are the propositional variables occurring in* Ψ. *Facts are rules of the form* $A \overset{\top}{\leftarrow} \top$.

A rule of a residuated logic program expresses a (monotonic) computation rule of the truth-value of the head propositional symbol from the truth-values of the symbols in the body. The monotonicity of the rule is guaranteed by isotonicity of formula Ψ: if an argument of Ψ is monotonically increased then the truth-value of Ψ also monotonically increases.

As usual, an interpretation is simply an assignment of truth-values to every propositional symbol in the language. To simplify the presentation we assume, throughout the rest of this section, that a residuated algebra $\mathfrak{R}$ is given with respect to $(\leftarrow, \otimes, \top)$.

Definition 5 (Interpretation). *An interpretation is a mapping* $I : \Pi \rightarrow \mathcal{T}_{\mathfrak{R}}$. *It is well known that an interpretation extends uniquely to a valuation function* $\hat{I}$ *from the set of formulas to the set of truth values. The set of all interpretations with respect to the residuated algebra* $\mathfrak{R}$ *is denoted by* $\mathcal{I}_{\mathfrak{R}}$.

The ordering $\preceq$ of the truth-values $\mathcal{T}_{\mathfrak{R}}$ is extended to the set of interpretations as usual:

Definition 6 (Lattice of interpretations). *Consider the set of all interpretations with respect to the residuated algebra* $\mathfrak{R}$ *and the two interpretations* $I_1, I_2 \in \mathcal{I}_{\mathfrak{R}}$. *Then,* $< \mathcal{I}_{\mathfrak{R}}, \sqsubseteq >$ *is a complete lattice where* $I_1 \sqsubseteq I_2$ *iff* $\forall_{p \in \Pi}\ I_1(p) \preceq I_2(p)$. *The least interpretation* $\triangle$ *maps every propositional symbol to the least element of* $\mathcal{T}_{\mathfrak{R}}$.

A rule of a residuated logic program is satisfied whenever the truth-value of the rule is greater or equal than the confidence factor associated with the rule. Formally:

Definition 7. *Consider an interpretation* $I \in \mathcal{I}_{\mathfrak{R}}$. *A weighted rule* $\langle (A \leftarrow \Psi), \vartheta \rangle$ *is satisfied by* I *iff* $\hat{I}\left((A \leftarrow \Psi)\right) \succeq \vartheta$. *An interpretation* $I \in \mathcal{I}_{\mathfrak{R}}$ *is a model of a residuated logic program* P *iff all weighted rules in* P *are satisfied by* I.

Mark that we used $\hat{I}$ instead of I in the evaluation of the truth-value of a rule, since a complex formula is being evaluated instead of a propositional symbol. If $\leftarrow_{\mathfrak{R}}$ is the function in $\mathfrak{R}$ defining the truth-table for the implication operator, the expression $\hat{I}\left((A \leftarrow \Psi)\right)$ is equal to

$$\hat{I}(A) \leftarrow_{\mathfrak{R}} \hat{I}(\Psi) = I(A) \leftarrow_{\mathfrak{R}} \hat{I}(\Psi)$$

The evaluation of $\hat{I}(\Psi)$ proceeds inductively as usual, till all propositional symbols in Ψ are reached and evaluated in I.

The immediate consequences operator of van Emden and Kowalski [14] is extended to the very general theoretical setting of residuated logic programs as follows:

Definition 8. *Let P be a residuated logic program. The monotonic immediate consequences operator $T_P^{\mathfrak{R}} : \mathcal{I}_{\mathfrak{R}} \to \mathcal{I}_{\mathfrak{R}}$, mapping interpretations to interpretations, is defined by:*

$$T_P^{\mathfrak{R}}(I)(A) = lub\left\{\vartheta \otimes \hat{I}(\Psi) \text{ such that } A \xleftarrow{\vartheta} \Psi[B_1, \ldots, B_n] \in P\right\}$$

As remarked before, the monotonicity of the operator $T_P^{\mathfrak{R}}$ has been shown in [2]. The semantics of a residuated logic program is characterized by the post-fixpoints of $T_P^{\mathfrak{R}}$:

Theorem 1. *An interpretation I of $\mathcal{I}_{\mathfrak{R}}$ is a model of a residuated logic program P iff $T_P^{\mathfrak{R}}(I) \sqsubseteq I$. Moreover, the semantics of P is given by its least model which is exactly the least fixpoint of $T_P^{\mathfrak{R}}$. The least model of P and can be obtained by trasfinitely iterating $T_P^{\mathfrak{R}}$ from the least interpretation Δ.*

The major difference from classical logic programming is that our $T_P^{\mathfrak{R}}$ may not be continuous, and therefore more than ω iterations may be necessary to reach the least fixpoint. This is unavoidable if of one wants to keep generality. All the other important results carry over to our general framework.

3 Hybrid Probabilistic Logic Programs

In this section we provide an overview of the main definitions and results in [4, 3]. We do not address any of the aspects of the proof theory present in these works. A major motivation for the Hybrid Probabilistic Logic Programs is the need for combining several probabilistic reasoning forms within a general framework. To capture this generality, the authors introduced the new notion of probabilistic strategies.

A first important remark is that the probabilites of compound events may be closed intervals in $[0, 1]$, and not simply real-valued probability assignments. The set of all closed intervals of [0,1] is denoted by $\mathcal{C}[0, 1]$. Recall that the empty set $\varnothing$ is a closed interval. In $\mathcal{C}[0, 1]$ two partial-orders are defined. Let $[a, b] \in \mathcal{C}[0, 1]$ and $[c, d] \in \mathcal{C}[0, 1]$, then:

- $[a, b] \leq_t [c, d]$ if $a \leq c$ and $b \leq d$, meaning that $[c, d]$ is closer to 1 than $[a, b]$.
- $[a, b] \subseteq [c, d]$ if $c \leq a$ and $b \leq d$, meaning that $[a, b]$ is more precise than $[c, d]$.

The probabilistic strategies must obey the following natural properties:

Definition 9 (Probabilistic strategy). *A p-strategy is a pair of functions $\rho =< c, md >$ such that:*

1. *$c : \mathcal{C}[0, 1] \times \mathcal{C}[0, 1] \to \mathcal{C}[0, 1]$ is called a probabilistic composition function satisfying the following axioms:*
   Commutativity: $c([a_1, b_1], [a_2, b_2]) = c([a_2, b_2], [a_1, b_1])$
   Associativity: $c(c([a_1, b_1], [a_2, b_2]), [a_3, b_3]) = c([a_1, b_1], c([a_2, b_2], [a_3, b_3]))$

Inclusion Monotonicity: *If* $[a_1, b_1] \subseteq [a_3, b_3]$ *then* $c([a_1, b_1], [a_2, b_2]) \subseteq c([a_3, b_3], [a_2, b_2])$

Separation: *There exist two functions* $c^1, c^2 : [0,1] \times [0,1] \to [0,1]$ *such that* $c([a,b],[c,d]) = [c^1(a,c), c^2(b,d)]$.

2. $md : \mathcal{C}[0,1] \to \mathcal{C}[0,1]$ *is called a maximal interval function.*

The strategies are either conjuntive or disjunctive:

Definition 10. *A p-strategy* $< c, md >$ *is called a conjunctive (disjunctive) p-strategy if it satisfies the following axioms:*

	Conjunctive p-strategy	*Disjunctive p-strategy*
Bottomline	$c([a_1,b_1],[a_2,b_2]) \leq_t [min(a_1,a_2), min(b_1,b_2)]$	$[max(a_1,a_2), max(b_1,b_2)] \leq_t c([a_1,b_1],[a_2,b_2])$
Identity	$c([a,b],[1,1]) = [a,b]$	$c([a,b],[0,0]) = [a,b]$
Annihilator	$c([a,b],[0,0]) = [0,0]$	$c([a,b],[1,1]) = [1,1]$
Max. Interval	$md([a,b]) = [a,1])$	$md([a,b]) = [0,b]$

The syntax of hybrid probabilistic logic programs (hp-programs) is built on a first-order language L generated from finitely many constants and predicate symbols. Thus, the Herbrand base B_L of L is finite. Without loss of generality, we restrict the syntax to a propositional language: variables are not admitted in atoms. This simplifies the embedding into residuated logic programs.

In a hp-program one can use arbitrary p-strategies. By definition, for each conjunctive p-strategy the existence of a corresponding disjunctive p-strategy is assumed, and vice-versa. Formally:

Definition 11. *Let* $\mathcal{CONJ}$ *be a finite set of coherent conjunctive p-strategies and* $DISJ$ *be a finite set of coherent disjunctive p-strategies. Let* $\mathcal{L}$ *denote* $\mathcal{CONJ} \cup \mathcal{DISJ}$. *If* $\rho \in \mathcal{CONJ}$ *then connective* $\wedge_\rho$ *is called a* ρ*-annotated conjunction. If* $\rho \in \mathcal{DISJ}$ *then* $\vee_\rho$ *is called a* ρ*-annotated disjunction.*

The elementary syntactic elements of hp-programs are basic formulas:

Definition 12. *Let* ρ *be a conjunctive p-strategy,* ρ' *be a disjunctive p-strategy and* $A_1, \ldots, A_k$ *be atoms. Then* $A_1 \wedge_\rho A_2 \wedge_\rho \ldots \wedge_\rho A_k$ *and* $A_1 \vee_{\rho'} A_2 \vee_{\rho'} \ldots \vee_{\rho'} A_k$ *are hybrid basic formulas. Let* $bf_\rho(B_L)$ *denote the set of all ground hybrid basic formulas for a connective. The set of ground hybrid basic formulas is* $bf_\mathcal{L} = \cup_{\rho \in \mathcal{L}} bf_\rho(B_L)$.

Basic formulas are annotated with probability intervals. Here we differ from [4] where basic formulas can be additionally annotated with variables and functions.

Definition 13. *A hybrid probabilistic annotated basic formula is an expression of the form* $B : \mu$ *where* B *is a hybrid basic formula and* $\mu \in \mathcal{C}[0,1]$.

Finally, we can present the syntax of hybrid rules and hp-programs:

Definition 14. *A hybrid probabilistic program over the set $\mathcal{L}$ of p-strategies is a finite set of hp-clauses of the form $B_0 : \mu_0 \leftarrow B_1 : \mu_1 \wedge \ldots \wedge B_k : \mu_k$ where each $B_i : \mu_i$ is a hp-annotated basic formula over $\mathcal{L}$.*

Intuitively, an hp-clause means that "if the probability of B_1 falls in the interval μ_1 and ... and the probability of B_k falls within the interval μ_k, then the probability of B_0 lies in the interval μ_0". Mark that the conjunction symbol $\wedge$ in the antecedent of hp-clauses should be interpreted as logical conjunction and should not be confused with a conjunctive p-strategy.

The semantics of hp-programs is given by a fixpoint operator. Atomic functions are akin to our notion of interpretation and are functions $f : B_L \to \mathcal{C}[0,1]$. They may be extended to hybrid basic formulas. For this the notion of splitting a formula into two disjoint parts is necessary::

Definition 15. *Let $F = F_1 *_\rho \ldots *_\rho F_n$, $G = G_1 *_\rho \ldots *_\rho G_k$, $H = H_1 *_\rho \ldots *_\rho H_m$ where $* \in \{\wedge, \vee\}$. We write $G \oplus_\rho H = F$ iff*

1. $\{G_1, \ldots, G_k\} \cup \{H_1, \ldots, H_m\} = \{F_1, \ldots, F_n\}$,
2. $\{G_1, \ldots, G_k\} \cap \{H_1, \ldots, H_m\} = \varnothing$,
3. $k > 0$ *and* $m > 0$.

The extension to atomic formulas is as follows:

Definition 16. *A hybrid formula function is a function $h : bf_{\mathcal{L}}(B_L) \to \mathcal{C}[0,1]$ which satisfies the following properties:*

1. **Commutativity**. *If $F = G_1 \oplus_\rho G_2$ then $h(F) = h(G_1 *_\rho G_2)$.*
2. **Composition**. *If $F = G_1 \oplus_\rho G_2$ then $h(F) \subseteq c_\rho(h(G_1), h(G_2))$.*
3. **Decomposition**. *For any basic formula F, $h(F) \subseteq md_\rho(h(F *_\rho G)$ for all $\rho \in \mathcal{L}$ and $G \in bf_{\mathcal{L}}(B_L)$.*

Let h_1 and h_2 be two hybrid formula functions. We say that $h_1 \leq h_2$ iff $(\forall F \in bf_{\mathcal{L}}(B_L))$ $h_1(F) \supseteq h_2(F)$. In particular, this means that there is a minimum element of $\mathcal{HFF}$ mapping every hybrid basic formula to $[0,1]$.

The immediate consequences operator for hp-programs resorts to the following auxiliary operator. Again, we consider the ground case only:

Definition 17. *Let P be a hp-program. Operator $S_P : \mathcal{HFF} \to \mathcal{HFF}$ is defined as follows, where F is a basic formula. $S_P(h)(F) = \cap M$ where $M = \{\mu | F : \mu \leftarrow F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n$ is an instance of some hp-clause in P and $(\forall j \leq n)\ h(F_j) \subseteq \mu_j\}$. Obviously, if $M = \varnothing$ then $S_P(h)(F) = [0,1]$.*

Definition 18. *Let P be a hp-program. Operator $T_P : \mathcal{HFF} \to \mathcal{HFF}$ is defined as follows:*

1. *Let F be an atomic formula.*
   (a) if $S_P(h)(F) = \varnothing$ then $T_P(h)(F) = \varnothing$.

(b) *if* $S_P(h)(F) \neq \varnothing$ *then let* $M = \{< \mu, \rho > |(F \oplus_\rho G) : \mu \leftarrow F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n$ *where* $* \in \{\vee, \wedge\}$, $\rho \in \mathcal{L}$ *and* $(\forall j \leq n)\ h(F_j) \subseteq \mu_j\}$. *We define*

$$T_P(h)(F) = (\cap \{md_\rho(\mu) | < \mu, \rho > \in M\}) \cap S_P(h)(F)$$

2. *If* F *is not atomic, then*

$$T_P(h)(F) = S_P(h)(F) \cap (\cap \{c_\rho(T_P(h)(G), T_P(h)(H)) \mid G \oplus H = F\}) \cap (\{md_\rho(\mu) | < \mu, \rho > \in M\})$$

where $M = \{< \mu, \rho > |\ D_1 *_\rho \ldots *_\rho D_k : \mu \leftarrow E_1 : \mu_1 \wedge \ldots \wedge E_m : \mu_m$ *such that* $(\forall j \leq n)\ h(E_j) \subseteq \mu_j$ *and* $\exists_H F \oplus_\rho H = \{D_1, \ldots, D_k\}\}$

A full explanation and intuition of the above operators can be found in [4]. Mark that the interval intersection operator $\cap$ in operators S_P and T_P corresponds to the join operation in lattice $C[0,1]$ ordered by containment relation $\supseteq$. For the continuation of our work it is enough to recall that the T_P operator is monotonic (on the containment relation) and that it has a least fixpoint. Furthermore, the least model of a hp-program is given by the least fixpoint of T_P. We will base our results in these properties of the T_P operator. We end this section with a small example from [4], adapted to the ground case.

Example 1. Assume that if the CEO of a company sells the stock, retires with the probability over 85% and we are ignorant about the relationship between the two events, then the probability that the stock of the company drops is 40-90%. However, if the CEO retires and sells the stock, but we know that the former entails the latter, then the probability that the stock of the company will drop is only 5-20%. This situation is formalized with the following two rules:

price-drop:[0.4,0.9] $\leftarrow$ (ch-sells-stock $\wedge_{igc}$ ch-retires):[0.85,1]
price-drop:[0.05,0.2] $\leftarrow$ (ch-sells-stock $\wedge_{pcc}$ ch-retires):[1,1]

Where $\wedge_{igc}$ is a conjunctive ignorance p-strategy with $c_{igc}([a_1, b_1], [a_2, b_2]) = [max(0, a_1 + a_2 - 1), min(b_1, b_2)]$, and $\wedge_{pcc}$ is the positive correlation conjunctive p-strategy such that $c_{pcc}([a_1, b_1], [a_2, b_2]) = [min(a_1, a_2), min(b_1, b_2)]$.

Now assume we have the two facts *ch-sells-stock:[1,1]* and *ch-retires:[0.9,1]*. In this case, we obtain in the model of P that the probability of *price-drop* is in [0.4,0.9] since the first rule will fire and the second won't. If instead of the above two facts we have *(ch-sells-stock* $\wedge_{igc}$ *ch-retires):[1,1]* then in the least fixpoint of T_P *price-drop* will be assigned $\varnothing$.

4 Embedding of Hybrid Probabilistic Logic Programs into Residuated Logic Programs

In this section we present the embedment result. This will require some effort. First, we need to define our underlying residuated lattice. We will not restrict ourselves to closed intervals of $[0,1]$. We require additional truth-values:

Definition 19. *Let $\mathcal{INT}$ be the set of pairs formed from values in $[0,1]$. We represent a value $< a, b > \in \mathcal{INT}$ by $[a,b]$. We say that $[a_1,b_1] \leq [a_2,b_2]$ iff $a_1 \leq a_2$ and $b_2 \leq b_1$.*

A pair $[a,b]$ in $\mathcal{INT}$ (with $a \leq b$) represents a non-empty closed interval of $C[0,1]$. The intuition for the remaining "intervals" of the form $[c,d]$ with $c > d$ will be provided later on, but we can advance now that they represent a form of inconsistent probability intervals. They correspond to $\varnothing$ in $C[0,1]$. The relation $\leq$ on $\mathcal{INT}$ forms a partial order, and extends the containment relation of $C[0,1]$ to $\mathcal{INT}$. In particular, $[0,1]$ and $[1,0]$ are, respectively, the least and greatest elements of $\mathcal{INT}$. These remarks are justified by the following two results:

Proposition 1. *The set $\mathcal{INT}$ with the partial order forms a complete lattice with the following meet and join operators:*

$$[a_1,b_1] \sqcap [a_2,b_2] = [min(a_1,a_2), max(b_1,b_2)]$$
$$[a_1,b_1] \sqcup [a_2,b_2] = [max(a_1,a_2), min(b_1,b_2)]$$

In general, consider the family $\{[a_i,b_i]\}_{i \in I}$ then

$$\sqcap_{i \in I}[a_i,b_i] = [inf\{a_i \mid i \in I\}, sup\{b_i \mid i \in I\}]$$
$$\sqcup_{i \in I}[a_i,b_i] = [sup\{a_i \mid i \in I\}, inf\{b_i \mid i \in I\}]$$

Proposition 2. *Consider the mapping $\overline{\cdot}$ from $\mathcal{INT}$ to $C[0,1]$ such that $\overline{[a,b]} = [a,b]$ if $a \leq b$, otherwise it is $\varnothing$. Let $[a_1,b_1]$ and $[a_2,b_2]$ belong to $C[0,1]$. Then,*

$$[a_1,b_1] \cap [a_2,b_2] = \overline{[a_1,b_1] \sqcup [a_2,b_2]}$$

Example 2. Consider the intervals $[0.5,0.7]$ and $[0.6,0.9]$. Their intersection is $[0.6,0.7]$ which is identical to their join in lattice $\mathcal{INT}$. Now, the intervals $[0.5,0.7]$ and $[0.8,0.9]$ have empty intersection. However their join is $[0.8,0.7]$. This will mean that there is some inconsistency in the assignment of probability intervals. In fact, we know that there is a gap from $[0.7,0.8]$. Thus, $\overline{[0.8,0.7]}$ is $\varnothing$.

The interpretation is a little more complex when more than two intervals are involved in the join operation. The intersection of $[0.1,0.2]$, $[0.4,0.6]$ and $[0.7,0.9]$ is empty again. Their join is $[0.7,0.2]$, meaning that the leftmost interval ends at 0.2 while the rightmost begins at 0.7. Again, $\overline{[0.7,0.2]} = \varnothing$.

We have seen that the meet and join operations perform the union and intersection of "intervals" of $\mathcal{INT}$, respectively. Our objective is to construct a residuated lattice from $\mathcal{INT}$ and the meet operation, which will be the multiplication operation. The adjoint residuum operation (implication) is defined as follows:

Definition 20. *Let $[a_1,b_1]$ and $[a_2,b_2]$ belong to $\mathcal{INT}$. Then:*

$$[a_1,b_1] \hookleftarrow [a_2,b_2] = \begin{cases} [1,0] \text{ if } a_2 \leq a_1 \text{ and } b_2 \geq b_1 \\ [1,b_1] \text{ if } a_2 \leq a_1 \text{ and } b_2 < b_1 \\ [a_1,0] \text{ if } a_2 > a_1 \text{ and } b_2 \geq b_1 \\ [a_1,b_1] \text{ if } a_2 > a_1 \text{ and } b_2 < b_1 \end{cases}$$

The result of the residuum operation is not obvious but still intuitive. In fact, we are testing whether $[a_2, b_2]$ contains $[a_1, b_1]$ (i.e. if $[a_1, b_1] \geq [a_2, b_2]$) and how $[a_2, b_2]$ should be extended in order to satisfy the inclusion. If the first (second) component of $[a_1, b_1] \hookleftarrow [a_2, b_2]$ is 1 (respectively 0) we do not have to do anything to $[a_2, b_2]$. Otherwise, a_2 (resp. b_2) should be reduced (increased) to a_1 (b_1). Notice again that $[1, 0]$ is our top element in lattice $\mathcal{INT}$.

Theorem 2. *The operations* $(\hookleftarrow, \sqcap)$ *form an adjoint pair in the partially ordered set* $< \mathcal{INT}, \leq >$.

Clearly, the structure $< \mathcal{INT}, \hookleftarrow, \sqcap >$ is a complete residuated lattice, with top element $[1, 0]$. A corresponding residuated algebra is easily constructed. We proceed by presenting a result which will enable the embedding of hybrid probabilistic logic programs into residuated logic programs:

Theorem 3. *Consider the operator* T'_P *which is identical to* T_P *except for when its argument formula* F *is not atomic; then:*

$$T'_P(h)(F) = S_P(h)(F) \cap (\cap \{c_\rho(h(G), h(H)) \mid G \oplus H = F\}) \cap (\{md_\rho(\mu)| < \mu, \rho > \in M\})$$

with M *defined as before. Then* h *is a fixpoint of* T_P *iff* h *is a fixpoint of* T'_P.

Proof: The only difference between the operators is that we have replaced $c_\rho(T_P(h)(G), T_P(h)(H))$ in T_P by $c_\rho(h(G), h(H))$ in T'_P. Clearly, if h is a fixpoint of T_P then it is also a fixpoint of T'_P, since $h = T_P(h)$ we can substitute h by $T_P(h)$ in the definition of T'_P getting T_P. For the other direction, we prove the result by induction on the number of atoms in F. If F is atomic then $T_P(h) = T'_P(h)$, by definition. Otherwise, F is not an atomic formula. Since h is a fixpoint of T'_P we have:

$$\begin{aligned} T'_P(h)(F) &= S_P(h)(F) \cap (\cap \{c_\rho(h(G), h(H)) \mid G \oplus H = F\}) \cap \\ &\quad (\{md_\rho(\mu)| < \mu, \rho > \in M\}) \\ &= S_P(h)(F) \cap (\cap \{c_\rho(T'_P(h)(G), T'_P(h)(H)) \mid G \oplus H = F\}) \cap \\ &\quad (\{md_\rho(\mu)| < \mu, \rho > \in M\}) \end{aligned}$$

But clearly G and H have a smaller number of atoms. So, from the induction hypothesis we know that $T'_P(h)(G) = T_P(h)(G)$ and $T'_P(h)(H) = T_P(h)(H)$. Substituting these equalities into the above equation we get $T'_P(h)(F) = S_P(h)(F) \cap (\cap \{c_\rho(T_P(h)(G), T_P(h)(H)) \mid G \oplus H = F\}) \cap (\{md_\rho(\mu)| < \mu, \rho > \in M\})$ which is $T_P(h)(F)$. □

Before we present the embedding, we need some auxiliary functions in $\mathcal{INT}$:

Definition 21. *The double bar function* $\overline{\overline{\cdot}}$ *from* $\mathcal{INT}$ *to* $\mathcal{INT}$ *and the functions* $s_\mu : \mathcal{INT} \to \mathcal{INT}$ *where* μ *in* $\mathcal{INT}$ *are defined as follows:*

$$\overline{\overline{[a, b]}} = \begin{cases} [1, 0], \textit{ if } a > b \\ [a, b], \textit{ otherwise.} \end{cases} \qquad s_\mu(\vartheta) = \begin{cases} [1, 0], \textit{ if } \mu \leq \vartheta \\ [0, 1], \textit{ otherwise.} \end{cases}$$

The above functions are clearly monotonic. Furthermore, the s_μ functions are "two-valued" and will be used to perform the comparisons in the rule bodies of a probabilistic logic program. Now, the embedding is immediate:

Definition 22. *Consider the hp-program P on the set of p-strategies $\mathcal{L}$. First, we construct the residual algebra $\mathfrak{I}$ from the carrier set $\mathcal{INT}$, and operations $\leftharpoondown$, $\sqcap$, $c_\rho(\rho \in \mathcal{L})$, $s_\mu(\mu \in \mathcal{INT})$, the double bar function, and the top constant $[1,0]$. Next, we build the residuated logic program P_{hp} from P as follows, where every ground hybrid basic formula in $bf_\mathcal{L}$ is viewed as a new propositional symbol[2] in the language of P_{hp}.*

1. *For each rule in P of the form $F : \mu \leftarrow F_1 : \mu_1 \wedge \ldots \wedge F_k : \mu_k$ we add to P_{hp} the rule[3] $F \overset{\mu}{\leftharpoondown} s_{\mu_1}\left(\overline{\overline{F_1}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{F_k}}\right)$.*

2. *For every, F, G, and H in $bf_\mathcal{L}$ such that $H = F \oplus_\rho G$, and ρ is a conjunctive p-strategy, then for every rule $H : [a,b] \leftarrow E_1 : \mu_1 \wedge \ldots \wedge E_m : \mu_m$ in P we add to P_{hp} the rule $F \overset{[a,1]}{\leftharpoondown} s_{\mu_1}\left(\overline{\overline{E_1}}\right) \sqcap \ldots \sqcap s_{\mu_m}\left(\overline{\overline{E_m}}\right)$.*

3. *For every, F, G, and H in $bf_\mathcal{L}$ such that $H = F \oplus_\rho G$, and ρ is a disjunctive p-strategy, then for every rule $H : [a,b] \leftarrow E_1 : \mu_1 \wedge \ldots \wedge E_m : \mu_m$ in P we add to P_{hp} the rule $F \overset{[0,b]}{\leftharpoondown} s_{\mu_1}\left(\overline{\overline{E_1}}\right) \sqcap \ldots \sqcap s_{\mu_m}\left(\overline{\overline{E_m}}\right)$.*

4. *Finally, for every F, G, and H in $bf_\mathcal{L}$ such that $F = G \oplus_\rho H$ then include in P_{hp} the rule $F \overset{[1,0]}{\leftharpoondown} c_\rho\left(\overline{\overline{G}}, \overline{\overline{H}}\right)$.*

Some remarks are necessary to fully clarify the above translation. First, the c_ρ functions were previously defined on domain $\mathcal{C}[0,1]$. It is required to extend them to $\mathcal{INT}$. For elements of $\mathcal{INT}$ isomorphic to elements of $\mathcal{C}[0,1]$ the functions should coincide. For values in $\mathcal{INT}$ not in $\mathcal{C}[0,1]$ the functions c_ρ can take arbitrary values, since in the embedding the arguments of these functions always take values from $\mathcal{C}[0,1]$.

Also, the above translation produces a residuated logic program. The rules belong to the algebra of formulae freely generated from the set of propositional symbols and operators in the corresponding residual algebra. Thus, when evaluating $F \leftharpoondown s_{\mu_1}\left(\overline{\overline{F_1}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{F_k}}\right)$ with respect to interpretation I we really mean $I(F) \leftharpoondown s_{\mu_1}\left(\overline{\overline{I(F_1)}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{I(F_k)}}\right)$, as usual. It should be clear that every body formula is isotonic on its arguments: for the first three types of rules the body is the composition of isotonic functions and therefore the resulting function is also isotonic. The probabilistic composition functions are isotonic by definition (check Definition 9).

[2] Without loss of generality, we assume that the ocurrences of atoms in each hybrid basic formula are ordered according to some total order in the set of all atoms.

[3] We assign to the body of translated facts the top constant $[1,0]$.

The rules introduced in the fourth step are exponential in the number of atoms (width) in F. This is expected since it is known that the computation of the least fixpoint of an HPP is exponential in the width of the largest formula of interest, as shown in [3]. The complexity of the entailment and consistency problems for HPPs are more subtle and the reader is referred again to [3] for these profound results.

Theorem 4. *Let P be a hybrid probabilistic logic program and P_{hp} the corresponding residuated logic program over $\mathfrak{I}$. Let h be the least fixpoint of $T^{\mathfrak{I}}_{P_{hp}}$ and h' be the least fixpoint of T'_P. Then, for every F in $bf_{\mathcal{L}}$, we have $h'(F) = \overline{h(F)}$.*

Proof: We will prove that for every F in $bf_{\mathcal{L}}$ we have $T'_P \uparrow^{\alpha} (F) = \overline{T^{\mathfrak{I}}_{P_{hp}} \uparrow^{\alpha} (F)}$. To simplify notation we drop the subscripts in the operators. The proof is by transfinite induction on α:

$\alpha = 0$**:** Trivial since every hybrid basic formula is mapped to $[0,1]$ in both operators.

Sucessor ordinal $\alpha = \beta + 1$**:** Let $h' = T' \uparrow^{\beta}$ and $h = T^{\mathfrak{I}} \uparrow^{\beta}$. By induction hypothesis we know that for every F in $bf_{\mathcal{L}}$ we have $h'(F) = \overline{h(F)}$. The essential point is that $h'(F) \subseteq \mu$ iff $s_\mu\left(\overline{\overline{h(F)}}\right) = [1,0]$. Therefore, we have the body of a rule in P satisfied by h' iff the body of the corresponding rule in P_{hp} evaluates to $[1,0]$. Otherwise, the body of the rule in P_{hp} has truth-value $[0,1]$.

Rules of the first kind in the embedding implement the S_P operator because

$$\begin{aligned} T^{\mathfrak{I}}(h)(F) &= \bigsqcup \Big\{ \mu \sqcap \hat{h}\left(s_{\mu_1}\left(\overline{\overline{F_1}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{F_k}}\right)\right) \\ &\qquad \text{such that } F \overset{\mu}{\longleftarrow} s_{\mu_1}\left(\overline{\overline{F_1}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{F_k}}\right) \in P_{hp} \Big\} \\ &= \bigsqcup \Big\{ \mu \sqcap [1,0] \text{ such that } F \overset{\mu}{\longleftarrow} s_{\mu_1}\left(\overline{\overline{F_1}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{F_k}}\right) \in P_{hp} \\ &\qquad \text{and } \hat{h}\left(s_{\mu_1}\left(\overline{\overline{F_1}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{F_k}}\right)\right) = [1,0] \Big\} \\ &= \bigsqcup \{ \mu \text{ where } F : \mu \leftarrow F_1 : \mu_1 \wedge \ldots \wedge F_k : \mu_k \text{ is satisfied by } h' \} \end{aligned}$$

Rules of the second and third kind extract the maximal interval associated with F with respect to connective ρ. By definition, we know that the maximal interval $md_\rho([a,b])$ is $[a,1]$ for a conjunctive p-strategy ρ, or $[0,b]$ if ρ is a disjunctive p-strategy. Therefore the rules of the second and third kind implement $\bigsqcup \{ md_\rho(\mu) | < \mu, \rho > \in M \}$ for both cases 1b) and 2 of Definition 18. Finally, the remaining rules compute $\bigsqcup \{ c_\rho(h'(G), h'(H)) \mid G \oplus H = F \}$. The result immediately follows from Proposition 2.

Limit ordinal α **other than** 0**:** We have to show that

$$\bigcap_{\beta<\alpha} T' \uparrow^{\beta} (F) = \overline{\bigsqcup_{\beta<\alpha} T^{\mathfrak{I}} \uparrow^{\beta} (F)}$$

Suppose for every $\beta < \alpha$ we have $T' \uparrow^{\beta} (F) \neq \varnothing$. This means that $T' \uparrow^{\beta} (F) = T^{\mathfrak{I}} \uparrow^{\beta} (F)$. The result then follows again from Proposition 2.

If for some β it is the case that $T' \uparrow^{\beta} (F) = \varnothing$ this means $T' \uparrow^{\alpha} (F) = \varnothing$. By induction hypothesis, $T^{\mathfrak{I}} \uparrow^{\beta} (F) = [a_\beta, b_\beta]$ with $a_\beta > b_\beta$. Let $T^{\mathfrak{I}} \uparrow^{\alpha} (F) = [a_\alpha, b_\alpha]$. We conclude $[a_\beta, b_\beta] \leq [a_\alpha, b_\alpha]$ by monotonicity of $T^{\mathfrak{I}}$, i.e. $a_\alpha \geq a_\beta$ and $b_\beta \geq b_\alpha$. Obviously, $a_\alpha > b_\alpha$ and the theorem holds.

□

By Theorem 3 we conclude immediately that $\overline{lfpT^{\mathfrak{I}}}$ is the least fixpoint of Dekhtyar and Subrahmanian's T_P operator, and the embedding is proved. The convergence of the process is guaranteed both by the properties of the T_P operator and the fixpoint results for residuated logic programs. We now return to Example 1 to illustrate the embedding. For simplicity, we ignore the rules generated in the fourth step for annotated disjunctions since they will not be required.

Example 3. The first two rules will be encoded as follows:

$$\text{price-drop} \overset{[0.4,0.9]}{\longleftarrow} s_{[0.85,1]} \left(\overline{\overline{\text{ch-sells-stock}} \wedge_{igc} \overline{\text{ch-retires}}}\right)$$
$$\text{price-drop} \overset{[0.05,0.2]}{\longleftarrow} s_{[1,1]} \left(\overline{\overline{\text{ch-sells-stock}} \wedge_{pcc} \overline{\text{ch-retires}}}\right)$$

Additionally, the following two rules will be introduced by the fourth step in the transformation:

$$\text{ch-sells-stock} \wedge_{igc} \text{ch-retires} \overset{[1,0]}{\longleftarrow} c_{igc} \left(\overline{\overline{\text{ch-sells-stock}}}, \overline{\overline{\text{ch-retires}}}\right)$$
$$\text{ch-sells-stock} \wedge_{pcc} \text{ch-retires} \overset{[1,0]}{\longleftarrow} c_{pcc} \left(\overline{\overline{\text{ch-sells-stock}}}, \overline{\overline{\text{ch-retires}}}\right)$$

In the first situation, the two facts will be translated to

$$\text{ch-sells-stock} \overset{[1,1]}{\longleftarrow} [1, 0]$$
$$\text{ch-retires} \overset{[0.9,1]}{\longleftarrow} [1, 0]$$

In the least fixpoint of $T^{\mathfrak{I}}$ the literals *ch-sells-stock* and *ch-retires* have truth-value $[1, 1]$ and $[0.9, 1]$, respectively. From this we obtain for the literals representing hybrid basic formulas *ch-sells-stock* $\wedge_{igc}$ *ch-retires* and *ch-sells-stock* $\wedge_{pcc}$ *ch-retires* the same truth-value of $[0.9, 1]$. Finally, we obtain the interval $[0.4, 0.9]$ by application of the first rule.

The fact (ch-sells-stock $\wedge_{igc}$ ch-retires):[1,1] will be encoded instead as follows, by application of the first and second rules:

$$\text{ch-sells-stock} \wedge_{igc} \text{ch-retires} \overset{[1,1]}{\longleftarrow} [1, 0]$$
$$\text{ch-sells-stock} \overset{[1,1]}{\longleftarrow} [1, 0]$$
$$\text{ch-retires} \overset{[1,1]}{\longleftarrow} [1, 0]$$

From the above facts we conclude that *ch-sells-stock* $\wedge_{pcc}$ *ch-retires* gets truth-value $[1, 1]$, and by application of the rules for *price-drop* we obtain for this literal the assignment $[0.4, 0.2]$, and as expected $\overline{[0.4, 0.2]} = \varnothing$.

We conclude by remarking that the substitution of $F : \mu$ by $s_\mu\left(\overline{\overline{F}}\right)$ instead of by $s_\mu(F)$ in the transformed program is of the essence. Otherwise, we could get different semantics when some literal is mapped to $\varnothing$. However, it is not clear what is the better semantics in that case, and further work is necessary. We illustrate the distinction in the next example:

Example 4. Consider the hp-program:

$$a : [0.5, 0.7] \leftarrow \qquad a : [0.8, 0.9] \leftarrow \qquad b : [1, 1] \leftarrow a : [0.9, 0.95]$$

According to the transformation of Definition 22 we have:

$$a \overset{[0.5,0.7]}{\leftarrowtail} [1, 0] \qquad a \overset{[0.8,0.9]}{\leftarrowtail} [1, 0] \qquad b \overset{[1,1]}{\leftarrowtail} s_{[0.9,0.95]}\left(\overline{\overline{a}}\right)$$

In the model of the program a is mapped to $[0.8, 0.7]$ and b to $[1, 1]$. Now, if we translate the rule for b as $b \overset{[1,1]}{\leftarrowtail} s_{[0.9,0.95]}(a)$, literal a is still mapped to $[0.8, 0.7]$. However, the body of the rule for b has truth-value $[0, 1]$, and b also has this value, since $[1, 1] \sqcap [0, 1] = [0, 1]$.

5 Conclusions and Further Work

The major contribution of this paper is the generality of our setting, both at the language and the semantic level. We presented an algebraic characterization of Residuated Logic Programs. Program rules have arbitrary monotonic body functions and our semantical structures are residuated lattices, where a generalized form of *Modus Ponens Rule* is valid. After having defined an implication (or residuum operator) and the associated multiplication (t-norm in the fuzzy logic setting) we obtain a logic programming semantics with corresponding model and fixpoint theory.

The embedding of hybrid probabilistic logic programs into residuated logic programs relies on a generalization of the complete lattice of closed intervals in $[0, 1]$. The extra truth-values capture invalid probability interval assignments, not used in [4]. The program transformation capturing the hp-semantics is a direct translation of the fixpoint conditions on a logic program. This aspect illustrates the generality and potential of our approach. Besides hp-programs we have shown that Generalized Annotated Logic Programs, Fuzzy Logic Programming, and Possibilistic Logic Programming are all captured by Residuated Logic Programs. These results could not be included for lack of space.

Our work paves the way to combine and integrate several forms of reasoning into a single framework, namely fuzzy, probabilistic, uncertain, and paraconsistent. We have also defined another class of logic programs, extending the Residuated one, where rule bodies can be anti-monotonic functions, with Well-Founded and Stable Model like semantics. This brings together non-monotonic and incomplete forms of reasoning to those listed before. It will be the subject of a forthcoming paper.

Acknowledgements

Work partially supported by PRAXIS project MENTAL. We thank V. S. Subrahmanian, J. Dix, T. Eiter, J. Alferes and L. Caires for helpful discussions. We also thank the anonoymous referees for their valuable comments.

References

1. L. Bolc and P. Borowik. *Many-Valued Logics. Theoretical Foundations.* Springer–Verlag, 1992.
2. C. V. Damásio and L. M. Pereira. Residuated logic programs. Technical report, Universidade Nova de Lisboa, 2000. Available at http://centria.di.fct.unl.pt/∼cd.
3. A. Dekhtyar, M. Dekhtyar, and V. S. Subrahmanian. Hybrid probabilistic programs: Algorithms and complexity. In *Proc. of 1999 Conference on Uncertainty in AI*, 1999.
4. A. Dekhtyar and V. S. Subrahmanian. Hybrid probabilistic programs. *J. of Logic Programming*, 43:187–250, 2000.
5. R. P. Dilworth and M. Ward. Residuated lattices. *Trans. A.M.S.*, 45:335–354, 1939.
6. D. Dubois, J. Lang, and H. Prade. Fuzzy sets in approximate reasoning, part 2: Logical approaches. *Fuzzy Sets and Systems*, 40:203–244, 1991.
7. D. Dubois, J. Lang, and H. Prade. Towards possibilistic logic programming. In *International Conference on Logic Programming 1991*, pages 581–598. MIT Press, 1991.
8. J. H. Gallier. *Logic for Computer Science.* John Wiley & Sons, 1987.
9. P. Hájek. *Metamathematics of Fuzzy Logic.* Trends in Logic. Studia Logica Library. Kluwer Academic Publishers, 1998.
10. M. Kifer and V. S. Subrahmanian. Theory of generalized annotated logic programming and its applications. *J. of Logic Programming*, 12:335–367, 1992.
11. R. Kowalski. Predicate logic as a programming language. In *Proceedings of IFIP'74*, pages 569–574. North Holland Publishing Company, 1974.
12. J. W. Lloyd. *Foundations of Logic Programming.* Springer–Verlag, 1987. 2nd. edition.
13. J. Pavelka. On fuzzy logic I, II, III. *Zeitschr. f. Math. Logik und Grundl. der Math.*, 25, 1979.
14. M. van Emden and R. Kowalski. The semantics of predicate logic as a programming language. *Journal of ACM*, 4(23):733–742, 1976.
15. P. Vojtás and L. Paulík. Soundness and completeness of non-classical extended SLD-resolution. In *Proc. of the Ws. on Extensions of Logic Programming (ELP'96)*, pages 289–301. LNCS 1050, Springer–Verlag., 1996.