Carlos Viegas Damásio
Luís Moniz Pereira

Hybrid Probabilistic Logic Programs as Residuated Logic Programs

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 with a logic programming semantics.

Keywords: Logic Programming, Probabilistic Logic Programming, Fuzzy logic

1. Introduction

The literature on logic programming theory is brimming with proposals of languages and semantics for extensions of definite logic programs (e.g. [6, 8, 11, 21]), 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 (whether continuous or not). Often these semantics are many-valued.

In this work 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 for instance [1, 10]) and applied these ideas to logic programming. In fact, a preliminary work in this direction is [21], 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 lattices (c.f. [1, 10]), 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 degree of confidence in the rules.

Besides fuzzy reasoning, probabilistic reasoning forms are essential for knowledge representation in real-world applications (see [5, 6, 13]). However, a major difficulty is that there are several logical ways of determining the probabilities of complex events (conjunctions or disjunctions) from primitive ones (for an overview and justification check [13]). To address this issue, a model theory, fixpoint theory and proof theory for hybrid probabilistic logic programs were recently introduced [5, 6]. The generality of Residuated Logic Programming is illustrated in practice by presenting an embedding of Hybrid Probabilistic Logic Programs [5, 6] into our framework.

Our paper, which is an extended version of [2], proceeds as follows. In the next section we present the residuated logic programs and its model and fixpoint theory. 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 all the proofs for the sake of completeness.

2. Residuated Logic Programs

The theoretical foundations of logic programming were clearly established in [12, 20] for definite logic programs (see also [14]), 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. In this section we generalize the language and semantics of definite logic programs in order to encompass more complex bodies and heads and, evidently, many-valued logics. For simplicity, we consider only the propositional (ground) case.

In a non-classical setting the intended truth-value of a given rule may possibly not be absolute truth. For instance, this is clear from the following *fuzzy logic programming* example of [18] adapted from [21]:

EXAMPLE 1. *Consider the following MYCIN rule [17]:*

IF	*the gram stain of the organism is gram negative, the morphology of the organism is rod, and the aerobicity of the organism is anaerobic.*
THEN	*there is suggestive evidence (0.5) that the identity of the organism is Bacteroides.*

This rule can be encoded, for a particular organism o, as:

$$\begin{array}{rl} \textit{identity(o,bacteroides)} \leftarrow & \textit{gram-stain(o,gram-negative)}, \\ & \textit{morphology(o,rod)}, \\ & \textit{aerobicity(o,anaerobic) with-cf=0.5} \end{array}$$

The assertion with-cf=0.5 *states that the confidence factor of the rule is 0.5.*

From the above example it is clear that it is required a generalization of *Modus Ponens* to reason logically with the confidence factors. In many-valued logics this issue is very well understood, namely in Fuzzy Propositional Logics [1, 10, 16]. Since one of our initial goals was to capture Fuzzy Logic Programming [7, 21], it was natural to adopt as semantical basis the *residuated lattices* (see for instance [1]). This section reviews some of the results presented in [4] and provides the proofs. 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. [10]):

> 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 $y \leftarrow z$ is the maximal x satisfying $x \otimes z \preceq_P y$.

Besides (a_1)–(a_3), it is necessary to assume the existence of a bottom and a top element in the lattice of truth-values, and that $\top$ is an unit element of $\otimes$. It is also usual to assume, additionally, extra conditions on the multiplication operation ($\otimes$), namely associativity, and commutativity (even though this is not strictly necessary, see [15]).

DEFINITION 2 (Residuated Lattice). *Let $\mathfrak{L} =< L, \preceq_L>$ be a lattice. We say that $(\mathfrak{L}, \leftarrow, \otimes)$ is a residuated lattice whenever the following three conditions are met:*

(l_1) *$\mathfrak{L}$ is a bounded lattice: it has a bottom ($\bot$) and a top element ($\top$);*

(l_2) *$(\leftarrow, \otimes)$ is an adjoint pair in $\mathfrak{L}$;*

(l_3) *$(L, \otimes, \top)$ is a commutative monoid.*

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

We have restricted ourselves to a single implication symbol in the definition of a residuated lattice. This definition has been extended in [15], where several implication operators can be put to use. 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).

DEFINITION 3 (Residuated Algebra). *Let $\mathfrak{T} =< \mathcal{T}, \preceq>$ be a complete lattice, and consider an algebra $\mathfrak{R}(\leftarrow_{\mathfrak{R}}, \otimes_{\mathfrak{R}}, \top_{\mathfrak{R}})$ on carrier set $\mathcal{T}$. Algebra $\mathfrak{R}(\leftarrow_{\mathfrak{R}}, \otimes_{\mathfrak{R}}, \top_{\mathfrak{R}})$ is a residuated algebra with respect to $\mathfrak{T}$ if it defines operators $\leftarrow_{\mathfrak{R}}, \otimes_{\mathfrak{R}}$ and $\top_{\mathfrak{R}}$ such that $(\mathfrak{T}, \leftarrow_{\mathfrak{R}}, \otimes_{\mathfrak{R}})$ is a complete residuated lattice. Furthermore, operator $\top_{\mathfrak{R}}$ is a constant mapping to the top element of $\mathfrak{T}$. The algebra may have other operators defined.*

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 [9] for more details.

DEFINITION 4 (Residuated Logic Programs). *Let $\mathfrak{R}(\leftarrow_{\mathfrak{R}}, \otimes_{\mathfrak{R}}, \top_{\mathfrak{R}})$ be a residuated algebra with respect to complete lattice $\mathfrak{T}$. Let Π be a set of proposi-*

tional symbols and the corresponding algebra of formulae[1] $FORM_{\mathfrak{R}}(\Pi)$ *freely generated from* Π. *A residuated logic program is a set of weighted rules of the form* $A \stackrel{\vartheta}{\leftarrow} \Psi$ *such that:*

1. *The rule* $(A \leftarrow \Psi)$ *is a formula of* $FORM_{\mathfrak{R}}(\Pi)$;
2. *The confidence factor* ϑ *is a truth-value in* $\mathcal{T}$;
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.*

In order to make explicit the propositional symbols $B_1, \ldots, B_n$ *occurring in* Ψ, *we may alternatively represent a rule by* $A \stackrel{\vartheta}{\leftarrow} \Psi[B_1, \ldots, B_n]$. *Facts are weighted rules of the form* $A \stackrel{\vartheta}{\leftarrow} \top$.

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 propositional symbols in the body. The monotonicity of the rule is guaranteed by the isotonicity of formula Ψ: if an argument of Ψ is monotonically increased then the truth-value of Ψ also monotonically increases.

We now proceed by presenting the model and fixpoint theory for residuated logic programs, associating a single (least) model with every program. To simplify the presentation we assume, throughout the rest of this section, that a residuated algebra $\mathfrak{R}(\leftarrow_{\mathfrak{R}}, \otimes_{\mathfrak{R}}, \top_{\mathfrak{R}})$ is given with respect to a complete lattice $\mathfrak{T} = (\mathcal{T}, \preceq)$. We deliberately omit the operators when referring to $\mathfrak{R}(\leftarrow_{\mathfrak{R}}, \otimes_{\mathfrak{R}}, \top_{\mathfrak{R}})$. As usual, we start by defining the notion of interpretation which is simply an assignment of some truth-value to each propositional symbol in the language.

DEFINITION 5 (Interpretation). *An interpretation is a mapping* $I : \Pi \rightarrow \mathcal{T}$. *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 in $\mathfrak{T}$ is extended to the set of interpretations as usual:

[1] We assume that operators $\leftarrow_{\mathfrak{R}}, \otimes_{\mathfrak{R}}$, and $\top_{\mathfrak{R}}$ are represented at the syntactic level by $\leftarrow$, $\otimes$ and $\top$. The multiplication and residuum operators are written in infix notation, as usual.

DEFINITION 6 (Lattice of interpretations). *Consider the set of all interpretations* $\mathcal{I}_{\mathfrak{R}}$ *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* Δ *maps every propositional symbol to the least element of* $\mathfrak{T}$.

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 (Model of a Residuated Logic Program). *Consider an interpretation* $I \in \mathcal{I}_{\mathfrak{R}}$. *A weighted rule* $A \stackrel{\vartheta}{\leftarrow} \Psi$ *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, then we have:

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

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 [20] is extended to the 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:*

$$T_P^{\mathfrak{R}}(I)(A) = lub\left\{\vartheta \otimes_{\mathfrak{R}} \hat{I}(\Psi) \text{ such that } A \stackrel{\vartheta}{\leftarrow} \Psi \in P\right\}$$

The monotonicity of the operator $T_P^{\mathfrak{R}}$ should be obvious, and the proof is presented below:

THEOREM 1 (Monotonicity of the immediate consequences operator). *Let* I_1 *and* I_2 *be two interpretations in* $\mathcal{I}_{\mathfrak{R}}$, *and* P *a residuated logic program. Operator* $T_P^{\mathfrak{R}}$ *is monotonic:*

$$\text{If } I_1 \sqsubseteq I_2 \text{ then } T_P^{\mathfrak{R}}(I_1) \sqsubseteq T_P^{\mathfrak{R}}(I_2)$$

To prove the theorem we require the following direct result:

LEMMA 1. *Let I_1 and I_2 be two interpretations in $\mathcal{I}_{\mathfrak{R}}$, and P a residuated logic program. If $I_1 \sqsubseteq I_2$ then $\hat{I}_1(\Psi) \preceq \hat{I}_2(\Psi)$, where Ψ is the body of a rule in P.*

PROOF OF THEOREM 1. Consider a propositional symbol A and a rule $A \overset{\vartheta}{\leftarrow} \Psi$ for it in P. By the previous lemma we know that $\hat{I}_1(\Psi) \preceq \hat{I}_2(\Psi)$. Since $\otimes_{\mathfrak{R}}$ is isotonic we get $(\vartheta \otimes_{\mathfrak{R}} \hat{I}_1(\Psi)) \preceq (\vartheta \otimes_{\mathfrak{R}} \hat{I}_2(\Psi))$. But then, for every ϑ_1 in $V_1(A) = \left\{\vartheta \otimes_{\mathfrak{R}} \hat{I}_1(\Psi) \text{ such that } A \overset{\vartheta}{\leftarrow} \Psi \in P\right\}$ there is a $\vartheta_2 \succeq \vartheta_1$ in $V_2(A) = \left\{\vartheta \otimes_{\mathfrak{R}} \hat{I}_2(\Psi) \text{ such that } A \overset{\vartheta}{\leftarrow} \Psi \in P\right\}$.

Therefore, $T_P^{\mathfrak{R}}(I_1)(A) = lub\ V_1(A) \preceq lub\ V_2(A) = T_P^{\mathfrak{R}}(I_2)(A)$. Since we have arbitrarily chosen A, we have established that $T_P^{\mathfrak{R}}(I_1) \sqsubseteq T_P^{\mathfrak{R}}(I_2)$, as intended. If there are no rules for A then $T_P^{\mathfrak{R}}(I_1) = T_P^{\mathfrak{R}}(I_2) = \bot$, since the least upper bound of an empty set is the bottom element of the underlying lattice. ∎

The set of models of P is characterized by the post-fixpoints of $T_P^{\mathfrak{R}}$, which is the crux for defining an appropriate semantics for residuated logic programs. All the results follow from monotonicity of the immediate consequences operator and from the next Theorem:

THEOREM 2. *An interpretation I of $\mathcal{I}_{\mathfrak{R}}$ is a model of a residuated logic program P iff $T_P^{\mathfrak{R}}(I) \sqsubseteq I$.*

PROOF. Given a propositional symbol A let

$$V(A) = \left\{\vartheta \otimes_{\mathfrak{R}} \hat{I}(\Psi) \text{ such that } A \overset{\vartheta}{\leftarrow} \Psi \in P\right\}$$

Then, I is a model of P iff for every rule $A \leftarrow \Psi$

$$\begin{array}{lll} & \vartheta \preceq \hat{I}((A \leftarrow \Psi)) & \\ \text{iff} & \vartheta \preceq (\hat{I}(A) \leftarrow_{\mathfrak{R}} \hat{I}(\Psi)) & \\ \text{iff} & \vartheta \otimes_{\mathfrak{R}} \hat{I}(\Psi) \preceq \hat{I}(A) & \text{(by condition } a_3) \\ \text{iff} & \vartheta \otimes_{\mathfrak{R}} \hat{I}(\Psi) \preceq I(A) & \\ \text{iff} & I(A) \text{ is an upper bound for } V(A) & \\ \text{iff} & \text{lub } V(A) \preceq I(A) & \\ \text{iff} & T_P^{\mathfrak{R}}(I)(A) \preceq I(A). & \end{array}$$

Since the chosen propositional symbol A was arbitrary, we have for every A, $T_P^{\mathfrak{R}}(I)(A) \preceq I(A)$, i.e. $T_P^{\mathfrak{R}}(I) \sqsubseteq I$. Again, the least upper bound of an empty set is the bottom element of the lattice. In this case, the result follows trivially. ∎

We can at last prove the claim made before:

THEOREM 3 (Existence of Least Model). *Let P be a residuated logic program. Then, the least fixpoint of $T_P^{\mathfrak{R}}$ is the least model of P.*

PROOF. By the previous theorem, the set of models $\mathcal{M}$ of P is

$$\mathcal{M} = \left\{ I \text{ such that } T_P^{\mathfrak{R}}(I) \sqsubseteq I \right\}$$

Let $M_P = glb\ \mathcal{M}$. The Knaster-Tarski fixpoint theorem [19] entails $M_P \in \mathcal{M}$ and $T_P^{\mathfrak{R}}(M_P) = M_P$. Again, by the same Theorem, it is known that M_P is the least fixpoint of $T_P^{\mathfrak{R}}$, and since it is the greatest lower bound of $\mathcal{M}$, obviously it is the least model of P. ■

THEOREM 4 (Fixpoint Semantics). *Let P be a residuated logic program, and consider the transfinite sequence of interpretations of $\mathcal{I}_{\mathfrak{R}}$:*

$$\begin{array}{rcl} T_P^{\mathfrak{R}}\uparrow^0 & = & \Delta \\ T_P^{\mathfrak{R}}\uparrow^{n+1} & = & T_P^{\mathfrak{R}}(T_P^{\mathfrak{R}}\uparrow^n),\ \textit{if } n+1 \textit{ is a successor ordinal} \\ T_P^{\mathfrak{R}}\uparrow^\alpha & = & \bigsqcup_{\beta<\alpha} T_P^{\mathfrak{R}}\uparrow^\beta,\ \textit{if } \alpha \textit{ is a limit ordinal} \end{array}$$

Then, there is an ordinal λ such that $T_P^{\mathfrak{R}}\uparrow^{\lambda+1} = T_P^{\mathfrak{R}}\uparrow^\lambda$, and the least model of P is $M_P = T_P^{\mathfrak{R}}\uparrow^\lambda$.

The results of this section are synthesized in the following definition:

DEFINITION 9 (Semantics of Residuated Logic Programs). *Let P be a Residuated Logic Program over residuated algebra $\mathfrak{R}$. The semantics of P is given by M_P, the least model of P which is the least fixpoint of the immediate consequences operator $T_P^{\mathfrak{R}}$.*

The major difference from classical logic programming is that our $T_P^{\mathfrak{R}}$ might not be continuous, and therefore more than ω iterations may be necessary to reach the least fixpoint. For the study of sufficient conditions to guarantee the continuity of the $T_P^{\mathfrak{R}}$ the reader is referred to [15].

3. Hybrid Probabilistic Logic Programs

In this section we provide an overview of the main definitions and results in [5, 6]. 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, those authors introduced the new notion of probabilistic strategies. For an alternative approach, see [13] and references therein.

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. This is justified by the fact that the conjunction and disjunction of events might not determine a single probability value, but may range over an interval of values.

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 properties:

Definition 10. *A probabilistic stategy (p-strategy for short) is a pair of functions $\rho =< c, md >$ such that:*

1. *The function $c : \mathcal{C}[0, 1] \times \mathcal{C}[0, 1] \to \mathcal{C}[0, 1]$ (called probabilistic composition function) satisfies 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. *The mapping $md : \mathcal{C}[0, 1] \to \mathcal{C}[0, 1]$ is called a maximal interval function.*

A probabilistic composition function defines how the probability intervals of two events are combined together, in either disjunctive or conjunctive fashion. The maximal interval function allows to obtain the best estimate of the probability interval of a single event given the compound one. The authors address the special case of strategies that are either conjunctive or disjunctive:

DEFINITION 11. *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	$md([a, b]) = [a, 1]$	$md([a, b]) = [0, b]$

Several natural probabilistic strategies are well-known in the literature: ignorance, independence, positive correlation, negative correlation and mutual exclusion. Ignorance should be applied whenever nothing is known about the interaction of the two events. In some situations, independence is an appropriate assumption if the occurrence of one event does not influence the other, and vice-versa. Positive correlation expresses that the two events overlap as much as possible, whilst negative correlation is the contrary. In mutual exclusion the events do not overlap at all. For a thorough analysis of these strategies see [6, 13].

Every conjunctive probabilistic strategy has a dual disjunctive strategy, and vice-versa. Thus, for each probabilistic strategy one assumes the existence of a conjunctive p-strategy, denoted by $\wedge_p$, and the corresponding disjunctive p-strategy, denoted by $\vee_p$. Formally:

DEFINITION 12. *Let $\mathcal{CONJ}$ be a finite set of conjunctive p-strategies and $\mathcal{DISJ}$ be a finite set of disjunctive p-strategies. Let $\mathcal{S}$ 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 hybrid probabilistic logic programs (hp-programs) will define rules assigning probability intervals to compound events, given a set of conditions on other (or same...) compound events. The syntax of hybrid probabilistic logic programs is built on a first-order language L generated from finitely many constants, variables and predicate symbols (with no function symbols). Let B_L denote the set of all ground atoms of L. 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. The elementary syntactic elements of hp-programs are basic formulas:

DEFINITION 13. *Let ρ be a conjunctive p-strategy, ρ' be a disjunctive p-strategy and $A_1, \ldots, A_k$ be distinct atoms. Then the following are hybrid basic formulas:*

$$A_1 \wedge_\rho A_2 \wedge_\rho \ldots \wedge_\rho A_k \text{ and } A_1 \vee_{\rho'} A_2 \vee_{\rho'} \ldots \vee_{\rho'} A_k$$

Let $bf_\rho(B_L)$ denote the set of all ground hybrid basic formulas for the $\vee_\rho$ and $\wedge_\rho$ connectives. The set of ground hybrid basic formulas is $bf_{\mathcal{S}}(B_L) = \bigcup_{\rho \in \mathcal{S}} bf_\rho(B_L)$.

Notice that it is neither allowed the mixture of conjunctive with disjunctive probabilistic strategies nor the use of different strategies in the same hybrid basic formula. Basic formulas are annotated with probability intervals. Here we differ from [6] where basic formulas can be additionally annotated with variables and functions.

DEFINITION 14. *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 15. *A hybrid probabilistic program over the set $\mathcal{S}$ of p-strategies is a finite set of formulas (called 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 $\mathcal{S}$.

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 16. *Let $F = F_1 *_\rho \ldots *_\rho F_n$, $G = G_1 *_\rho \ldots *_\rho G_k$, and $H = H_1 *_\rho \ldots *_\rho H_m$ 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 hybrid basic formulas is as follows:

DEFINITION 17. *A hybrid formula function* $h : bf_{\mathcal{S}}(B_L) \to \mathcal{C}[0,1]$ *is a function 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* $\rho \in \mathcal{S}$ *and* $G \in bf_{\mathcal{S}}(B_L)$.

The set of all hybrid formula functions generated by some arbitrary but fixed set of p-strategies is denoted by $\mathcal{HFF}$. Let h_1 and h_2 be two hybrid formula functions. We say that $h_1 \leq h_2$ iff $(\forall F \in bf_{\mathcal{S}}(B_L))\ h_1(F) \supseteq h_2(F)$. In particular, this means that there is a minimum element of $\mathcal{HFF}$ mapping every hybrid basic formula to the interval $[0, 1]$.

The immediate consequences operator for hp-programs resorts to the following auxiliary operator, which determines the probability intervals from the applicable rules in the hp-program. Again, we consider the ground case only:

DEFINITION 18. *Let* P *be a ground hp-program. The operator* $S_P : \mathcal{HFF} \to \mathcal{HFF}$ *is defined as follows, where* F *is a basic formula.*

$$\begin{aligned} S_P(h)(F) \quad = \bigcap\{ \quad & \mu \mid F : \mu \leftarrow F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n \textit{ some} \\ & \textit{hp-clause in } P \textit{ and } (\forall j \leq n)\ h(F_j) \subseteq \mu_j\}. \end{aligned}$$

If there is no hp-clause in the above conditions then $S_P(h)(F) = [0, 1]$.

The S_P operator does not take into account the influence of compound events into basic (elemental) ones. Knowing the probability of a compound event may sharpen the probability intervals of its sub-events. This is achieved with the following operator:

DEFINITION 19. *Let* P *be a hp-program. The immediate consequences operator* $T_P : \mathcal{HFF} \to \mathcal{HFF}$ *is defined as follows:*

1. *Let* F *be an atom.*

(a) *if* $S_P(h)(F) = \varnothing$ *then* $T_P(h)(F) = \varnothing$.

(b) *if* $S_P(h)(F) \neq \varnothing$ *then define*

$$\begin{array}{rl} M = \{<\mu, \rho>| & (F \oplus_\rho G) : \mu \leftarrow F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n \ \textit{with} \\ & \rho \in \mathcal{S} \ \textit{and} \ (\forall j \leq n) \ h(F_j) \subseteq \mu_j\} \end{array}$$

$$\textit{Let } T_P(h)(F) = (\bigcap \{md_\rho(\mu) \mid < \mu, \rho > \in M\}) \bigcap S_P(h)(F)$$

2. *If* F *is not an atom, i.e. is of the form* $F_1 *_\rho \ldots *_\rho F_n$, *then*

$$\begin{array}{rl} T_P(h)(F) = & S_P(h)(F) \\ & \bigcap (\cap \{c_\rho(T_P(h)(G), T_P(h)(H)) \ \mid G \oplus_\rho H = F\}) \\ & \bigcap (\{md_\rho(\mu)| \mid < \mu, \rho > \in M\}) \end{array}$$

with $M = \{< \mu, \rho >| \ D_1 *_\rho \ldots *_\rho D_k : \mu \leftarrow E_1 : \mu_1 \wedge \ldots \wedge E_m : \mu_m$ *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 [6]. Mark that the interval intersection operator $\cap$ in operators S_P and T_P corresponds to the join operation in lattice $\mathcal{C}[0,1]$ ordered by containment relation $\supseteq$.

We end this section with a small example from [6], adapted to the ground case.

EXAMPLE 2. *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] $\leftarrow$ *(sells-stock* $\wedge_{igc}$ *retires):[0.85,1]*
price-drop:[0.05,0.2] $\leftarrow$ *(sells-stock* $\wedge_{pcc}$ *retires):[1,1]*

Where $\wedge_{igc}$ *is a conjunctive ignorance probabilistic 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 where*

$$c_{pcc}([a_1, b_1], [a_2, b_2]) = [min(a_1, a_2), min(b_1, b_2)]$$

Assume that the two facts sells-stock:[1,1] and retires:[0.9,1] hold. 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. The fixpoint construction is detailed below:

	price-drop	*sells-stock*	*retires*	*sells-stock* $\wedge_{igc}$ *retires*	*sells-stock* $\wedge_{pcc}$ *retires*
h_0	$[0,1]$	$[0,1]$	$[0,1]$	$[0,1]$	$[0,1]$
$S_P(h_0)$	$[0,1]$	$[1,1]$	$[0.9,1]$	$[0,1]$	$[0,1]$
$h_1 = T_P(h_0)$	$[0,1]$	$[1,1]$	$[0.9,1]$	$[0.9,1]$	$[0.9,1]$
$S_P(h_1)$	$[0.4,0.9]$	$[1,1]$	$[0.9,1]$	$[0,1]$	$[0,1]$
$h_2 = T_P(h_1)$	$[0.4,0.9]$	$[1,1]$	$[0.9,1]$	$[0.9,1]$	$[0.9,1]$

If instead of the above two facts we have simply

$$(\textit{sells-stock} \wedge_{igc} \textit{retires}) : [1,1]$$

then in the least fixpoint of T_P price-drop will be assigned $\varnothing$, which illustrates the use of the maximal decomposition functions:

	price-drop	*sells-stock*	*retires*	*sells-stock* $\wedge_{igc}$ *retires*	*sells-stock* $\wedge_{pcc}$ *retires*
h_0	$[0,1]$	$[0,1]$	$[0,1]$	$[0,1]$	$[0,1]$
$S_P(h_0)$	$[0,1]$	$[0,1]$	$[0,1]$	$[1,1]$	$[0,1]$
$h_1 = T_P(h_0)$	$[0,1]$	$[1,1]$	$[1,1]$	$[1,1]$	$[1,1]$
$S_P(h_1)$	$\varnothing$	$[0,1]$	$[0,1]$	$[1,1]$	$[0,1]$
$h_2 = T_P(h_1)$	$\varnothing$	$[1,1]$	$[1,1]$	$[1,1]$	$[1,1]$

4. Embedding of Hybrid Probabilistic Logic Programs into Residuated Logic Programs

In this section we present the embedding 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 20. *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 $\mathcal{C}[0,1]$. 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 $\mathcal{C}[0,1]$. The relation $\leq$ on $\mathcal{INT}$ forms a partial order, and extends the containment relation of $\mathcal{C}[0,1]$ 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 $\bar{\cdot}$ from $\mathcal{INT}$ to $\mathcal{C}[0,1]$ 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 $\mathcal{C}[0,1]$. Then,*

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

EXAMPLE 3. *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 21. *Let* $[a_1, b_1]$ *and* $[a_2, b_2]$ *belong to* $\mathcal{INT}$. *Then:*

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

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 $[a_1, b_1] \longleftarrow [a_2, b_2]$ 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 5. *The operations* $(\longleftarrow, \sqcap)$ *form an adjoint pair in the partially ordered set* $< \mathcal{INT}, \leq >$.

PROOF. The validity of the first condition is immediate from the properties of the meet operation in lattices. Next, we have to show that the residuum operation is isotonic in the first argument and antitonic in the antecedent. The proofs proceed by case analysis on the possible result of the operation.

We start by showing isotonicity in the first argument. Consider the results of $[a_1, b_1] \longleftarrow [a, b]$ and $[a_2, b_2] \longleftarrow [a, b]$ with $[a_1, b_1] \leq [a_2, b_2]$. One can conclude immediately that $a_1 \leq a_2$ and $b_2 \leq b_1$. There are four possible outcomes for the first operation:

- $[a_1, b_1]$: Therefore $a > a_1$ and $b < b_1$. It is obvious that $[a_2, b_2] \longleftarrow [a, b]$ can have as result any of $[a_2, b_2]$, $[1, b_2]$, $[a_2, 0]$ and $[1, 0]$. Clearly, $[a_1, b_1]$ is less or equal than any of these results. For instance, $[a_1, b_1] \leq [1, b_2]$ since $a_1 \leq 1$ and $b_2 \leq b_1$, by hypothesis.

- $[1, b_1]$: This means that $a \leq a_1$ and $b < b_1$, thus $a \leq a_2$. Therefore $[a_2, b_2] \longleftarrow [a, b]$ is either $[1, b_2]$ or $[1, 0]$. We conclude immediately that $[1, b_1] \leq [1, b_2]$ and $[1, b_1] \leq [1, 0]$.

- The proofs for the remaining cases, $[a_1, 0]$ and $[1, 0]$, are similar.

The proof of antitonicity in the antecedent is a little more complex, but still straightforward. We want to compare the result of $[a, b] \longleftarrow [a_1, b_1]$ with $[a, b] \longleftarrow [a_2, b_2]$, where $[a_1, b_1] \leq [a_2, b_2]$. Again, $a_1 \leq a_2$ and $b_2 \leq b_1$. If the result of $[a, b] \longleftarrow [a_1, b_1]$ is:

- $[1, 0]$: Trivial.

- $[a, 0]$: Thus $a_1 > a$ and $b_1 \geq b$. Therefore $a_2 > a$. Now, we have $b \leq b_2 \leq b_1$ or $b_2 < b \leq b_1$. We have $[a, b] \longleftarrow [a_2, b_2]$ equal to $[a, 0]$ and $[a, b]$, respectively in the first and second cases. Clearly, $[a, 0] \leq [a, 0]$ and $[a, b] \leq [a, 0]$, as desired.

- The proofs for the remaining cases, $[1, b]$ and $[a, b]$, are similar.

There remains to prove the third condition of the adjoint pair definition:

$$([a_1, b_1] \sqcap [a_2, b_2]) \leq [a_3, b_3] \text{ iff } [a_1, b_1] \leq ([a_3, b_3] \longleftarrow [a_2, b_2])$$

which is equivalent to

$$[min(a_1, a_2), max(b_1, b_2)] \leq [a_3, b_3] \text{ iff } [a_1, b_1] \leq ([a_3, b_3] \longleftarrow [a_2, b_2])$$

Therefore, we have to show that $min(a_1, a_2) \leq a_3$ and $b_3 \leq max(b_1, b_2)$ hold if and only if $[a_1, b_1] \leq ([a_3, b_3] \longleftarrow [a_2, b_2])$. Again, we proceed by case analysis. Assume that $[a_3, b_3] \longleftarrow [a_2, b_2]$ is equal to:

- $[1, 0]$: Then, $a_2 \leq a_3$ and $b_2 \geq b_3$. The equivalence to show is:

  $$min(a_1, a_2) \leq a_3 \text{ and } b_3 \leq max(b_1, b_2) \text{ iff } [a_1, b_1] \leq [1, 0]$$

  The righthand side is always true. Since we know that $a_2 \leq a_3$, clearly $min(a_1, a_2) \leq a_3$, and $b_3 \leq max(b_1, b_2)$ because $b_3 \leq b_2$.

- $[a_3, b_3]$: Therefore we must have $a_2 > a_3$ and $b_2 < b_3$. Moreover, $min(a_1, a_2) \leq a_3$ iff $a_1 \leq a_3$, and $b_3 \leq max(b_1, b_2)$ iff $b_3 \leq b_1$. Thus,

  $$\begin{array}{rl} & min(a_1, a_2) \leq a_3 \text{ and } b_3 \leq max(b_1, b_2) \text{ iff } a_1 \leq a_3 \text{ and } b_3 \leq b_1 \\ \equiv & min(a_1, a_2) \leq a_3 \text{ and } b_3 \leq max(b_1, b_2) \text{ iff } [a_1, b_1] \leq [a_3, b_3] \end{array}$$

  as intended.

The proofs for the remaining cases, $[1, b_3]$ and $[a_3, 0]$, are similar. ∎

Clearly, the structure $< \mathcal{INT}, \leftarrow, \sqcap >$ is a complete residuated lattice, with top element $[1, 0]$. A corresponding residuated algebra is easily constructed. The attentive reader noticed certainly that the T_P operator of Dekhtyar and Subrahmanian defines the value of non atomic hybrid basic formulas in terms of the combination of applications of the composition functions to sub-formulas, and not directly from the hybrid formula function argument. This is a major shortcoming for defining an embedding to residuated logic programs since the values of rule bodies are always determined with respect to a given interpretation. The next theorem helps circumvent the problem:

THEOREM 6. *Consider the operator T'_P which is identical to T_P except for when its argument formula F is not an atom; then:*

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

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 an atom then $T_P(h) = T'_P(h)$, by definition. Otherwise, F is not an atom. Since h is a fixpoint of T'_P we have:

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

Clearly G and H have a smaller number of atoms. By the induction hypothesis $T'_P(h)(G) = T_P(h)(G)$ and $T'_P(h)(H) = T_P(h)(H)$. Substituting these equalities into the above equation we get

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

which is the definition of $T_P(h)(F)$. ∎

A property of the hp-programs is that the probability intervals of annotated basic formulas in the body are only used for testing containment of the interval assigned to the associated basic formula. We will need an auxiliary function to perform such comparisons in $\mathcal{INT}$. Also, in order to respect the semantics of hp-programs we have to map every inconsistent probability interval to $[1,0]$ before performing the test. These functions are defined below:

DEFINITION 22. *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:*

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

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 23. *Consider the hp-program* P *on the set of p-strategies* $\mathcal{S}$. *First, we construct the residuated algebra* $\mathfrak{I}$ *from the carrier set* $\mathcal{INT}$, *and operations*[2] $\longleftarrow$, $\sqcap$, $c_\rho(\rho \in \mathcal{S})$, $s_\mu(\mu \in \mathcal{INT})$, *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 hybrid basic formula in* $bf_{\mathcal{S}}(B_L)$ *is viewed as a new propositional symbol*[3] *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*[4] $F \overset{\mu}{\longleftarrow} s_{\mu_1}\left(\overline{\overline{F_1}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{F_k}}\right)$.

2. *For every,* F, G, *and* H *in* $bf_{\mathcal{S}}(B_L)$ *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* $F \overset{[a,1]}{\longleftarrow} s_{\mu_1}\left(\overline{\overline{E_1}}\right) \sqcap \ldots \sqcap s_{\mu_m}\left(\overline{\overline{E_m}}\right)$.

[2] To avoid unnecessary notational overhead we do not subscript the operations with $\mathfrak{I}$.

[3] 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.

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

3. *For every, F, G, and H in $bf_{\mathcal{S}}(B_L)$ 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 $F \overset{[0,b]}{\longleftarrow} s_{\mu_1}\left(\overline{\overline{E_1}}\right) \sqcap \ldots \sqcap s_{\mu_m}\left(\overline{\overline{E_m}}\right)$.

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

Some remarks are necessary to fully clarify the above translation. First, the c_ρ functions were previously defined on the domain $\mathcal{C}[0,1]$. It is required extending 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 but keeping isotonicity, 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 residuated algebra. Thus, when evaluating $F \longleftarrow s_{\mu_1}\left(\overline{\overline{F_1}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{F_k}}\right)$ with respect to interpretation I we really mean as usual

$$I(F) \longleftarrow s_{\mu_1}\left(\overline{\overline{I(F_1)}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{I(F_k)}}\right)$$

It should be clear that every body formula is isotonic on its arguments: for the first three types of rule the body is the composition of isotonic functions and therefore the resulting function is also isotonic. The probabilistic composition functions are isotonic by definition (see Definition 10).

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 a hybrid probabilistic logic program is exponential in the width of the largest formula of interest, as shown in [5]. If one is only interested in basic formulas with a limited maximum width we can drop all other rules introduced in step 4 of Definition 23. It is argued in [5] that a width larger than a small constant (e.g. 4) will be of no practical use. In this case our transformation is polynomial in the size of the original program. The complexity of the entailment and consistency problems for hybrid probabilistic programs are more subtle and the reader is referred again to [5] for these profound results.

THEOREM 7. *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 $bf_{\mathcal{S}}(B_L)$, we have $h'(F) = \overline{h(F)}$.*

PROOF. We will prove that for every F in $bf_{\mathcal{S}}(B_L)$ 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 $bf_{\mathcal{S}}(B_L)$ 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 $T^{\mathfrak{I}}(h)(F)$ is equal to

$$\begin{aligned}
& \bigsqcup \Big\{\mu \sqcap \hat{h}\left(s_{\mu_1}\left(\overline{\overline{F_1}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{F_k}}\right)\right) \\
& \qquad \text{such that } F \stackrel{\mu}{\longleftarrow} s_{\mu_1}\left(\overline{\overline{F_1}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{F_k}}\right) \in P_{hp}\Big\} \\
= \;& \bigsqcup \Big\{\mu \sqcap [1,0] \text{ such that } F \stackrel{\mu}{\longleftarrow} s_{\mu_1}\left(\overline{\overline{F_1}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{F_k}}\right) \in P_{hp} \\
& \qquad \text{and } \hat{h}\left(s_{\mu_1}\left(\overline{\overline{F_1}}\right) \sqcap \ldots \sqcap s_{\mu_k}\left(\overline{\overline{F_k}}\right)\right) = [1,0]\Big\} \\
= \;& \bigsqcup \{\mu \text{ where } F : \mu \leftarrow F_1 : \mu_1 \wedge \ldots \wedge F_k : \mu_k \text{ is satisfied by } h'\}
\end{aligned}$$

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

$$\bigsqcup \{md_{\rho}(\mu) \mid < \mu, \rho > \in M\}$$

for both cases 1b) and 2 of Definition 19. Finally, the remaining rules compute $\bigsqcup \{c_{\rho}(h'(G), h'(H)) \mid G \oplus_{\rho} H = F\}$. The result immediately follows from Proposition 2.

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

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

Suppose for every $\beta < \alpha$ we have $T' \uparrow^{\beta} (F) \neq \varnothing$. This means that $T' \uparrow^{\beta} (F) = T^{\Im} \uparrow^{\beta} (F)$. 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, $T^{\Im} \uparrow^{\beta} (F) = [a_\beta, b_\beta]$ with $a_\beta > b_\beta$. Let $T^{\Im} \uparrow^{\alpha} (F) = [a_\alpha, b_\alpha]$. We conclude $[a_\beta, b_\beta] \leq [a_\alpha, b_\alpha]$ by monotonicity of $T^{\Im}$, i.e. $a_\alpha \geq a_\beta$ and $b_\beta \geq b_\alpha$. Obviously, $a_\alpha > b_\alpha$ and the theorem holds.

■

By Theorem 6 we conclude immediately that $\overline{lfpT^{\Im}}$ 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. A simple inspection of the proofs presented in the reference work on Hybrid Probabilistic Programs [6] shows that the T_P operator is monotonic (on the containment relation) independently of the mapping $h : bf_{\mathcal{S}}(B_L) \rightarrow \mathcal{C}[0,1]$ obeying or not the conditions of Definition 17. Furthermore, it has a least fixpoint which corresponds to the least model of a hp-program. The existence of such least fixpoint in [6] assumes that the probabilistic strategies are continuous. We did not require such assumption here because we have based all our arguments on standard fixpoint results and relations, which do not depend on continuity assumptions. As remarked before, the T_P operator is monotonic on the complete lattice of the functions mapping hybrid basic formulas to probability intervals in $\mathcal{C}[0,1]$. The original transfinite inductive construction of the T_P operator over the complete partial-ordered set of hybrid formula functions in [6] corresponds exactly to the construction of the least fixpoint of T_P over the complete lattice of arbitrary interpretations of hybrid formulas over $\mathcal{C}[0,1]$, which ensures the validity of our results given the result of Theorem 6.

We now return to Example 2 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 4. *The first two rules will be encoded as follows:*

$$
\begin{array}{l}
\textit{price-drop} \stackrel{[0.4,0.9]}{\longleftarrow} s_{[0.85,1]}\left(\overline{\overline{\textit{sells-stock} \wedge_{igc} \textit{retires}}}\right) \\
\textit{price-drop} \stackrel{[0.05,0.2]}{\longleftarrow} s_{[1,1]}\left(\overline{\overline{\textit{sells-stock} \ \wedge_{pcc} \textit{retires}}}\right)
\end{array}
$$

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

$$
\begin{array}{l}
\textit{sells-stock} \wedge_{igc} \textit{retires} \stackrel{[1,0]}{\longleftarrow} c_{igc}\left(\overline{\overline{\textit{sells-stock}}}, \overline{\overline{\textit{retires}}}\right) \\
\textit{sells-stock} \wedge_{pcc} \textit{retires} \stackrel{[1,0]}{\longleftarrow} c_{pcc}\left(\overline{\overline{\textit{sells-stock}}}, \overline{\overline{\textit{retires}}}\right)
\end{array}
$$

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

$$
\begin{array}{l}
\textit{sells-stock} \stackrel{[1,1]}{\longleftarrow} [1,0] \\
\textit{retires} \stackrel{[0.9,1]}{\longleftarrow} [1,0]
\end{array}
$$

In the least fixpoint of $T^{\Im}$ *the literals sells-stock and retires have truth-value* $[1,1]$ *and* $[0.9,1]$, *respectively. We get for the literals representing hybrid basic formulas, sells-stock* $\wedge_{igc}$ *retires and sells-stock* $\wedge_{pcc}$ *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 (sells-stock $\wedge_{igc}$ *retires):[1,1] will be encoded instead as follows, by application of the first and second rules:*

$$
\begin{array}{l}
\textit{sells-stock} \wedge_{igc} \textit{retires} \stackrel{[1,1]}{\longleftarrow} [1,0] \\
\textit{sells-stock} \stackrel{[1,1]}{\longleftarrow} [1,0] \\
\textit{retires} \stackrel{[1,1]}{\longleftarrow} [1,0]
\end{array}
$$

From the above facts we conclude that sells-stock $\wedge_{pcc}$ *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 since we are in a paraconsistent case, and further work is necessary, namely how to extend the probabilistic composition functions to this more general setting. We illustrate the distinction in the next example:

EXAMPLE 5. *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 23 we have:

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

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* $b \overset{[1,1]}{\longleftarrow} s_{[0.9,0.95]}(a)$, *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. This is an instance of the larger class of Monotonic Logic Programs defined in [3].

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 [6]. 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 the potential of our approach.

An alternative approach to probabilistic deductive databases was recently presented in [13]. It differs from Dekhtyar and Subrahmanian's approach in not allowing compound events in the bodies and rules of programs. Thus, the combination and interaction of several strategies is not automatically handled. However, it allows the definition of probabilistic confidence levels expressing both the belief and doubt in the probability of an event. These confidence levels are associated with facts and rules, and form a trilattice. Each rule has a mode for combining the confidence levels in the body (similar to conjunctive p-strategies) and another one for joining the conclusions of rules (the disjunctive strategy). A fixpoint operator is also defined. The translation to residuated logic programming is almost trivial. We will present this translation elsewhere.

Besides probabilistic approaches to logic programming, we have shown that Generalized Annotated Logic Programs, Fuzzy Logic Programming, and Possibilistic Logic Programming are all captured by Residuated Logic Programs [4].

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 Monotonic one, where rule bodies can be anti-monotonic functions, with Well-Founded and Stable Model like semantics [3]. This brings together non-monotonic forms of reasoning to those listed before.

Acknowledgements

Work partially supported by POCTI project FLUX. We thank V. S. Subrahmanian, J. Dix, T. Eiter, J. Alferes, L. Caires and M. Ojeda-Aciego for helpful discussions. We also thank the anonymous referees paper for their valuable and insightful comments.

References

[1] L. Bolc and P. Borowik. *Many-Valued Logics. Theoretical Foundations.* Springer–Verlag, 1992.

[2] C. V. Damásio and L. M. Pereira. Hybrid probabilistic logic programs as residuated logic programs. In M. Ojeda-Aciego, I. P. de Guzmán, G. Brewka, and L. M. Pereira, editors, *Logics in AI, Proceedings of JELIA'00*, pages 57–72, Málaga, Spain, September 2000. LNAI 1919, Springer–Verlag.

[3] C. V. Damásio and L. M. Pereira. Antitonic logic programs. In T. Eiter and M. Truszczynski, editors, *Procs. of the 6th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'01)*, LNCS/LNAI. Springer, 2001.

[4] C. V. Damásio and L. M. Pereira. Monotonic and residuated logic programs. In S. Benferhat and P. Besnard, editors, *6th European Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'01)*, LNCS/LNAI. Springer, 2001.

[5] A. Dekhtyar, M. Dekhtyar, and V. S. Subrahmanian. Hybrid probabilistic programs: Algorithms and complexity. In *Proc. of 1999 Conf. on Uncertainty in AI*, 1999.

[6] A. Dekhtyar and V. S. Subrahmanian. Hybrid probabilistic programs. *J. of Logic Programming*, 43:187–250, 2000.

[7] D. Dubois, J. Lang, and H. Prade. Fuzzy sets in approximate reasoning, part 2: Logical approaches. *Fuzzy Sets and Systems*, 40:203–244, 1991.

[8] D. Dubois, J. Lang, and H. Prade. Towards possibilistic logic programming. In *International Conference on Logic Programming 1991*, pages 581–598. MIT Press, 1991.

[9] J. H. Gallier. *Logic for Computer Science*. John Wiley & Sons, 1987.

[10] P. Hájek. *Metamathematics of Fuzzy Logic*. Trends in Logic. Studia Logica Library. Kluwer Academic Publishers, 1998.

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

[12] R. Kowalski. Predicate logic as a programming language. In *Proceedings of IFIP'74*, pages 569–574. North Holland Publishing Company, 1974.

[13] L. V. S. Lakshmanan and F. Sadri. On a theory of probabilistic deductive databases. *Theory and Practice of Logic Programming*, 1(1):5–42, 2001.

[14] J. W. Lloyd. *Foundations of Logic Programming*. Springer–Verlag, 1987.

[15] J. Medina, M. Ojeda-Aciego, and P. Vojtas. Multi-adjoint logic programming with continuous semantics. In T. Eiter and M. Truszczynski, editors, *Procs. of the 6th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR'01)*, LNCS/LNAI. Springer, 2001.

[16] J. Pavelka. On fuzzy logic I, II, III. *Zeitschr. f. Math. Logik und Grundl. der Math.*, 25, 1979.

[17] E. H. Shortlife and B. G. Buchanan. A model of inexact reasoning in medicine. *Math. Biosci*, 23:351–379, 1975.

[18] L. Sterling and E. Shapiro. *The Art of Prolog*. MIT Press, 1986.

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

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

[21] 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.

Carlos Viegas Damásio
Centro de Inteligência Artificial (CENTRIA)
Departamento de Informática, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa
2829-516 Caparica
Portugal
cd@di.fct.unl.pt

Luís Moniz Pereira
Centro de Inteligência Artificial (CENTRIA)
Departamento de Informática, Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa
2829-516 Caparica
Portugal
lmp@di.fct.unl.pt