

New Semantics for Hybrid Probabilistic Programs

Emad Saad

Department of Computer Science
New Mexico State University
Las Cruces, New Mexico
emsaad@cs.nmsu.edu

Abstract. Hybrid probabilistic programs framework [5] is a variation of probabilistic annotated logic programming approach, which allows the user to explicitly encode the available knowledge about the dependency among the events in the program. In this paper, we extend the language of hybrid probabilistic programs by allowing disjunctive composition functions to be associated with heads of clauses and change its semantics to be more suitable for real-life applications. We show on a probabilistic AI planning example that the new semantics allows us to obtain more intuitive and accurate probabilities. The new semantics of hybrid probabilistic programs subsumes Lakshmanan and Sadri [17] framework of probabilistic logic programming. The fixpoint operator for the new semantics is guaranteed to be always continuous. This is not the case in the probabilistic annotated logic programming in general and the hybrid probabilistic programs framework in particular.

1 Introduction

Reasoning under uncertain knowledge is an important issue in most real-life applications including those in AI domains. In the literature, logic programming has been augmented with various notions of uncertainty [5, 6, 8, 9, 12, 13, 16–20, 22, 23, 25–27, 30, 32], aiming to develop a sophisticated semantics for logic programming with uncertainty. The difference among these frameworks is based mainly on the underlying formalism of uncertainty and the way how certainty values are attached to rules and facts in the logic programs. These formalisms include fuzzy set theory [30, 32], possibilistic logic [6], hybrid (a combination of numerical and non-numerical [16, 18, 19], multi-valued logic [8, 9, 12, 13], and probability theory [5, 17, 25–27].

In probabilistic logic programming framework, Decktyar and Subramanian [5] have proposed the notion of hybrid probabilistic programs (*HPP*). Hybrid probabilistic programs are built upon the idea of annotated logic programs introduced in [31], and extensively studied in [12, 13, 21, 24]. The idea of hybrid probabilistic programs is to enable the user to encode his/her knowledge about the dependency among the probabilistic events being described in the programs. *HPP* is considered a generalization of probabilistic annotated logic programming

approach, which is proposed in [25] and further extended in [26], by allowing different probabilistic strategies instead of a fixed probabilistic strategy as in [25, 26].

The aim of probabilistic logic programming in general and hybrid probabilistic programs framework in particular, is the ability to reason and make decisions under uncertain knowledge, which is represented by the well defined probability theory. Yet, to achieve this goal, more sophisticated semantics for hybrid probabilistic programs needs to be established to be able to perform the reasoning tasks. Consequently, hybrid probabilistic programs framework becomes more suitable for real-life applications. To illustrate the idea consider the following example.

Example 1. Consider the following robot planning task adapted from [15]. Suppose a robot’s grasping operation is not always successful because of the chance of the robot’s gripper to be wet. Suppose that most of the time the robot is able to grasp a block with a probability 0.95 after executing the *pickup* action in the state of the world in which the gripper is dry. Moreover, the robot is able to grasp the block with probability 0.5 in the state of the world in which the gripper is wet. Assume, initially the block is not held and the gripper is dry with probability 0.7 (this means the gripper is wet with probability 0.3) and the goal is to grasp the block. There are two world states in which the robot can hold the block after executing the action *pickup*. Given the initial state in which the gripper is dry and the robot does not hold the block, the first resulting state is the state in which gripper is dry and robot holds the block. The probability of the block is being held in this state is $0.7 \times 0.95 = 0.665$. Also, given the initial state in which the gripper is wet and the robot does not hold the block, the second resulting state is the state in which gripper is wet and robot holds the block. The probability of the block is being held in this state is $0.3 \times 0.5 = 0.15$. Hence, the robot is successfully holds the block after executing the *pickup* action under the pre-mentioned initial conditions with probability $0.665 + 0.15 = 0.815$.

This planning domain problem can be represented as a logic program in hybrid probabilistic programs framework, which supports propagation of probability, classical negation, conditional probability and Bayesian updates, as follows:

$$\begin{aligned}
holdBlock : [0.95 \times V, 0.95 \times V] &\leftarrow pickup : [1, 1], gripperDry : [V, V] \\
holdBlock : [0.5 \times V, 0.5 \times V] &\leftarrow pickup : [1, 1], not_gripperDry : [V, V] \\
not_gripperDry : [1 - V, 1 - V] &\leftarrow gripperDry : [V, V] \\
pickup : [1, 1] &\leftarrow \\
gripperDry : [0.7, 0.7] &\leftarrow
\end{aligned}$$

Example 1 is pointing out a shortcoming in probabilistic annotated logic programming approach semantics, in general, and in hybrid probabilistic programs framework, in particular. Although, the problem encoding for the gripper planning domain in the hybrid probabilistic programs framework is correct,

the hybrid probabilistic programs semantics failed to assign the correct probabilistic interval to the hybrid basic formula *holdBlock* which is $[0.815, 0.815]$. This can be shown by computing the least fixpoint of the hybrid probabilistic program encoding for the problem. The least fixpoint of the program assigns $[1, 1]$ to *pickup*, $[0.7, 0.7]$ to *gripperDry*, $[0.3, 0.3]$ to *not_gripperDry*, and \emptyset to *holdBlock*. \emptyset is assigned to *holdBlock* because by applying the first rule in the program *holdBlock* is concluded with the probability interval $[0.665, 0.665]$ and by applying the second rule *holdBlock* is concluded with the probability interval $[0.15, 0.15]$. Therefore, according to the hybrid probabilistic programs semantics, *holdBlock* is assigned $\cap\{[0.665, 0.665], [0.15, 0.15]\} = \emptyset$

The reason for this shortcoming is arising, mainly, from the definition of the ordering (set inclusion order) employed in the hybrid probabilistic programs semantics. By recalling the definition of the set inclusion ordering in hybrid probabilistic programs framework [5], given the probabilistic intervals $[a_1, a_2], [b_1, b_2] \subseteq [0, 1]$ for a certain event e , $[a_1, a_2] \subseteq [b_1, b_2]$ iff $[b_1, b_2] \subseteq [a_1, a_2]$. This set inclusion order is known as the knowledge order. This means $[b_1, b_2]$ provides more precise probabilistic knowledge about the probability of e than $[a_1, a_2]$. However, when reasoning is performed to make decisions, the reasoning does not depend on how more knowledgeable we are about the probability of the various events but they depend mainly on how likely are these events to occur. Therefore, the ordering which describes for a certain event e how more likely that event might occur is more convenient for decision making and reasoning tasks. This intuition can be captured by employing the natural ordering \leq_t (truth order) described in [5, 17] in logic programming with probability. The truth order \leq_t asserts that if $[a_1, a_2], [b_1, b_2] \subseteq [0, 1]$ are two probabilistic intervals for the events e_1 and e_2 respectively, then $[a_1, a_2] \leq_t [b_1, b_2]$ iff $a_1 \leq b_1$ and $a_2 \leq b_2$. Hence, the event e_2 is more likely to occur than event e_1 . Therefore, changing the semantics of hybrid probabilistic programs to deal with the truth order is more sophisticated than the existing semantics which deals with set inclusion order.

Using the truth order \leq_t with the hybrid probabilistic programs framework requires changing its current syntax and semantics to adapt with this new ordering. The change includes using disjunctive composition functions similar to that are used in [17] for combing the probabilistic intervals associated to the same hybrid basic formula (an atom, a conjunction of atoms, or a disjunction of atoms) derived from different rules. As a consequence, the problem described in example 1 and the likes are captured by the new hybrid probabilistic programs semantics. In addition, from the new semantics, more intuitive and accurate probabilistic intervals can be derived which reflects the correct solutions of the problems. Moreover, with the new semantics, it can be shown that hybrid probabilistic programs can subsume Lakshmanan and Sadri [17] approach for probabilistic logic programming which was not possible with the semantics of hybrid probabilistic programs in [5]. Additional advantage of the new semantics is that the T_P operator is guaranteed to be always continuous. This is not the case in probabilistic annotated logic programming in general and hybrid prob-

abilistic programs framework in particular, since the T_P operator is not always continuous [25, 26, 5].

The remaining of this paper is organized as follows. Section 2 reviews the basic definitions of hybrid probabilistic programs framework as described in [5]. Section 3 presents the new semantics for hybrid probabilistic programs. In section 4, we show how the new semantics of hybrid probabilistic programs subsumes Lakshmanan and Sadri [17] approach for probabilistic logic programming. Finally, section 5 presents some concluding remarks.

2 Preliminaries

In this section we overview some of the basic definitions related to HPPs [5]. Let $C[0, 1]$ denotes the set of all closed intervals of $[0, 1]$.

Definition 1. *A probabilistic strategy (p-strategy) is a pair of functions $\rho = \langle c, md \rangle$, such that:*

1. *c is a probabilistic composition function $c : C[0, 1] \times C[0, 1] \rightarrow C[0, 1]$ that satisfies the following axioms:*
 - (a) *Commutativity:* $c([a_1, b_1], [a_2, b_2]) = c([a_2, b_2], [a_1, b_1])$
 - (b) *Associativity:* $c(c([a_1, b_1], [a_2, b_2]), [a_3, b_3]) = c([a_1, b_1], c([a_2, b_2], [a_3, b_3]))$
 - (c) *Monotonicity:* $c([a_1, b_1], [a_2, b_2]) \subseteq c([a_3, b_3], [a_2, b_2])$ if $[a_1, b_1] \subseteq [a_3, b_3]$
 - (d) *Separation:* *there exist two functions c_1 and c_2 such that $c([a_1, b_1], [a_2, b_2]) = (c_1([a_1, a_2]), c_2([b_1, b_2]))$*
2. *$md : C[0, 1] \rightarrow C[0, 1]$ is called the maximal interval function.*

Given the probability range of a complex event, the maximal interval function md returns the best estimate of the probability range of a primitive event. The composition function c returns the probability range of a conjunction or disjunction of two or more events. Given that $M = \{[a_1, b_1], \dots, [a_n, b_n]\}$ is a set of probabilistic intervals, we will denote cM to be equivalent to the expression $c([a_1, b_1], c([a_2, b_2], \dots, c([a_{n-1}, b_{n-1}], [a_n, b_n]) \dots)$. According to the type of combination (conjunctive or disjunctive) among events, p-strategies are classified into conjunctive p-strategies and disjunctive p-strategies [5].

Example 2 ([5]). In this example, only composition functions of different types of p-strategies are provided, maximum interval functions are defined uniquely by the type of p-strategy.

1. Independence p-strategy (*in*):
 - Conjunctive (*inc*): $c_{inc}([a_1, b_1], [a_2, b_2]) = [a_1 \cdot a_2, b_1 \cdot b_2]$.
 - Disjunctive (*ind*): $c_{ind}([a_1, b_1], [a_2, b_2]) = [a_1 + a_2 - a_1 \cdot a_2, b_1 + b_2 - b_1 \cdot b_2]$.
2. Ignorance p-strategy (*ig*):
 - Conjunctive (*igc*): $c_{igc}([a_1, b_1], [a_2, b_2]) = [\max(0, a_1 + a_2 - 1), \min(b_1, b_2)]$.
 - Disjunctive (*igd*): $c_{igd}([a_1, b_1], [a_2, b_2]) = [\max(a_1, a_2), \min(1, b_1 + b_2)]$.
3. Positive correlation p-strategy (*pc*):
 - Conjunctive (*pcc*): $c_{pcc}([a_1, b_1], [a_2, b_2]) = [\min(a_1, a_2), \min(b_1, b_2)]$.

- Disjunctive (pcd): $c_{pcd}([a_1, b_1], [a_2, b_2]) = [\max(a_1, a_2), \max(b_1, b_2)]$.
- 4. Mutual exclusion p-strategy (me):
 - Disjunctive (med): $c_{med}([a_1, b_1], [a_2, b_2]) = [(a_1 + a_2), (b_1 + b_2)]$ such that $a_1 + a_2 \leq b_1 + b_2 \leq 1$.

Let L be an arbitrary but fixed first-order language with finitely many predicate symbols, constants, and infinitely many variables. Function symbols are not allowed. In addition, let S be an arbitrary but fixed set of p-strategies. The notions of terms, atoms, and literals are defined as usual. The Herbrand base of L is denoted by B_L . An annotation is an expression of the form $[\alpha_1, \alpha_2]$ where α_1 and α_2 are annotation items. An annotation item is a constant in $[0, 1]$, a variable ranging over $[0, 1]$ (called annotation variable), or a computable total function, called annotation function, where an n-ary annotation function f is a mapping $f : ([0, 1])^n \rightarrow [0, 1]$. When α_1 and α_2 are constants, then $[\alpha_1, \alpha_2]$ is called constant annotation. If α_1 and α_2 are annotation variables then $[\alpha_1, \alpha_2]$ is called variable annotation.

Definition 2. Let ρ be a probabilistic strategy and A_1, \dots, A_n be atoms. Then $A_1 \wedge_\rho \dots \wedge_\rho A_n$ and $A_1 \vee_\rho \dots \vee_\rho A_n$ are called hybrid basic formulas. $bf_S(B_L)$ is the set of all ground hybrid basic formulas formed using distinct atoms from B_L and p-strategies from S .

Definition 3. A hybrid probabilistic clause (hp -clause) is an expression of the form $F : \mu \leftarrow F_1 : \mu_1, \dots, F_n : \mu_n$ where F, F_1, \dots, F_n are hybrid basic formulas, and μ, μ_1, \dots, μ_n are annotations, such that every annotation variable (if any) occurring in μ also occurs in at least one of μ_1, \dots, μ_n . $F : \mu$ is called the head of the hp -clause and $(F_1 : \mu_1, \dots, F_n : \mu_n)$ is its body.

Definition 4. A hybrid probabilistic program over S (hp -program) is a finite set of hp -clauses involving only p-strategies from S .

Definition 5. Given $F = F_1 *_\rho \dots *_\rho F_n$, $G = G_1 *_\rho \dots *_\rho G_k$, and $H = H_1 *_\rho \dots *_\rho H_m$, where $* \in \{\wedge, \vee\}$. Then $G \oplus_\rho H = F$ iff $k > 0$ and $m > 0$, $\{G_1, \dots, G_k\} \cup \{H_1, \dots, H_m\} = \{F_1, \dots, F_n\}$, and $\{G_1, \dots, G_k\} \cap \{H_1, \dots, H_m\} = \emptyset$.

Definition 6. A hybrid formula function h is a mapping $h : bf_S(B_L) \rightarrow C[0, 1]$ iff the following conditions are satisfied:

1. *Commutativity.* $h(F) = h(G_1 *_\rho G_2)$ if $F = G_1 \oplus_\rho G_2$.
2. *Composition.* $h(F) \subseteq c_\rho(h(G_1), h(G_2))$ if $F = G_1 \oplus_\rho G_2$.
3. *Decomposition.* For any hybrid basic formula F , $h(F) \subseteq md_\rho(h(F *_\rho G))$ for all $\rho \in S$ and $G \in bf_S(B_L)$.

3 New Semantics for Hybrid Probabilistic Programs

In the new semantics, the composition functions in disjunctive p-strategies are used to combine the probabilistic intervals of the same hybrid basic formula

derived from different hp-clauses. For example, if a hybrid probabilistic program consists of the hp-clauses

$$\begin{aligned} a : [0.5, 0.6] &\leftarrow b : [0.7, 0.7] \\ a : [0.4, 0.7] &\leftarrow c : [0.5, 0.8] \\ b : [0.7, 0.7] &\leftarrow \\ c : [0.5, 0.8] &\leftarrow \end{aligned}$$

and it is known that deriving a from the first hp-clause with probability $[0.5, 0.6]$ is positively correlated to deriving a with probability $[0.4, 0.7]$ from the second hp-clause, then a can be concluded with probability $[0.5, 0.7]$ using the composition function of the disjunctive positive correlation p-strategy. A similar idea is introduced in [17].

3.1 Syntax

In this subsection, we provide a new syntax for hybrid probabilistic programs by modifying the notions of hp-clause and hp-program in [5] by allowing the user to encode his knowledge about how to disjoin the probabilistic intervals for the same hybrid basic formula derived from different clauses.

Definition 7. A hybrid probabilistic rule (*h-rule*) is an expression of the form $r \equiv (F : \mu \leftarrow F_1 : \mu_1, \dots, F_n : \mu_n; \rho_F)$ where F, F_1, \dots, F_n are hybrid basic formulas, μ, μ_1, \dots, μ_n are annotations, and $\rho_F \in S$. $F : \mu$ is the head of the h-rule, $(F_1 : \mu_1, \dots, F_n : \mu_n)$ is its body, and ρ_F is a disjunctive p-strategy associated to F in r indicating how to disjoin the probabilistic intervals associated to F from different derivations of heads of h-rules involving F .

Definition 8. A hybrid probabilistic program over S (*h-program*) is a finite set of h-rules involving only connectives from S such that whenever a hybrid basic formula F is in heads of more than one h-rule, the disjunctive p-strategies associated to F in these h-rules are identical.

Definition 9. Let P be an h-program. P is said to be a ground h-program iff all h-rules in P do not include variables or annotation variables.

In what follows, the Herbrand base of a ground h-program is denoted by B_P .

3.2 Declarative Semantics

The definition of satisfaction are based on the notion of hybrid formula functions which has been defined earlier in the previous section.

Definition 10. Let h be a hybrid formula function, $F \in bf_S(B_L)$, and $\mu \in C[0, 1]$. Then

1. h satisfies $F : \mu$ iff $\mu \leq_t h(F)$.
2. h satisfies $F_1 : \mu_1, \dots, F_n : \mu_n$ iff for all $1 \leq i \leq n$, h satisfies $F_i : \mu_i$.
3. h satisfies $(F : \mu \leftarrow F_1 : \mu_1, \dots, F_n : \mu_n; \rho_F)$ iff h satisfies $F : \mu$ or h does not satisfy $(F_1 : \mu_1, \dots, F_n : \mu_n)$.

Since probabilistic intervals of the same hybrid basic formula F derived from different h-rules are combined together to strengthen the overall probabilistic interval of F , more conditions need to be imposed on the the satisfaction of h-programs. The following definitions are needed in defining the satisfaction of h-programs.

Definition 11. Let P be a ground h-program and h be a hybrid formula function. Then for each $F \in \text{bf}_S(B_P)$:

1. If F is atomic then

$$M_1^h = \{ \langle \mu, \rho \rangle \mid ((F *_\rho G) : \mu \leftarrow F_1 : \mu_1, \dots, F_n : \mu_n; \rho_{FG}) \in P, * \in \{ \vee, \wedge \}, \rho, \rho_{FG} \in S, \text{ and for all } i, 1 \leq i \leq n, \mu_i \leq_t h(F_i) \}.$$
2. $F = F_1 *_\rho \dots *_\rho F_n$ is not atomic then

$$M_2^h = \{ \langle \mu, \rho \rangle \mid ((D_1 *_\rho \dots *_\rho D_k) : \mu \leftarrow E_1 : \mu_1, \dots, E_m : \mu_m; \rho_D) \in P \text{ and for all } i, 1 \leq i \leq n, \mu_i \leq_t h(E_i); \{F_1, \dots, F_n\} \subset \{D_1, \dots, D_k\}, n < k \}.$$

Definition 12. Let P be a ground h-program, h be a hybrid formula function, and HFF be the set of all hybrid formula functions. The intermediate operator S_P is defined as a mapping $S_P : HFF \rightarrow HFF$ such that $S_P(h)(F) = c_{\rho_F} M$ where

$M = \{ \mu \mid (F : \mu \leftarrow F_1 : \mu_1, \dots, F_n : \mu_n; \rho_F) \in P \text{ and for all } i, 1 \leq i \leq n, \mu_i \leq_t h(F_i) \}$. If $M = \emptyset$, $S_P(h)(F) = [0, 0]$ and ρ_F is assumed to be the disjunctive ignorance p-strategy.

Definition 13. Let P be a ground h-program and h be a hybrid formula function. Then, h satisfies P if h satisfies every h-rule in P and for all $F \in \text{bf}_S(B_P)$ the following condition is satisfied:

1. If F is atomic then

$$c_{\rho_F}(c_{\rho_F} \{ \text{md}_\rho(\mu) \mid \langle \mu, \rho \rangle \in M_1^h \}, S_P(h)(F)) \leq_t h(F).$$
2. If $F = F_1 *_\rho \dots *_\rho F_n$ is not atomic then

$$c_{\rho_F}(S_P(h)(F), c_{\rho_F}(c_{\rho_F} \{ c_\rho(h(G), h(H)) \mid G \oplus_\rho H = F \}, c_{\rho_F} \{ \text{md}_\rho(\mu) \mid \langle \mu, \rho \rangle \in M_2^h \})) \leq_t h(F).$$

Definition 14. Let P be a ground h-program and h be a hybrid formula function. Then, h is a probabilistic model of P iff h satisfies P .

Definition 15 ([17]). Let $[a_1, a_2], [b_1, b_2] \in C[0, 1]$. The meet \otimes_t and the join \oplus_t operations corresponding to \leq_t are defined as:

1. $[a_1, a_2] \otimes_t [b_1, b_2] = [\min\{a_1, b_1\}, \min\{a_2, b_2\}]$.
2. $[a_1, a_2] \oplus_t [b_1, b_2] = [\max\{a_1, b_1\}, \max\{a_2, b_2\}]$.

Definition 16. Let h_1 and h_2 be formula functions. $h_1 \leq_t h_2$ iff, for all $F \in \text{bf}_S(B_L)$, $h_1(F) \leq_t h_2(F)$.

Definition 17. Let $h_1, h_2 \in \text{HFF}$. The meet \otimes_t and the join \oplus_t operations with respect to \leq_t are defined respectively as follows, for all $F \in \text{bf}_S(B_L)$,

1. $(h_1 \otimes_t h_2)(F) = h_1(F) \otimes_t h_2(F)$
2. $(h_1 \oplus_t h_2)(F) = h_1(F) \oplus_t h_2(F)$.

Lemma 1. The set of all hybrid formula functions HFF along with the truth order \leq_t forms a complete lattice.

The top element of the lattice, $\langle \text{HFF}, \leq_t \rangle$, is the mapping $\text{bf}_S(B_L) \rightarrow [1, 1]$ and the bottom element is the mapping $\text{bf}_S(B_L) \rightarrow [0, 0]$.

Lemma 2. Let P be a ground h -program and h_1, h_2 be two probabilistic models satisfying P . Then, $h_1 \otimes_t h_2$ is also a probabilistic model satisfying P .

Lemma 3 (The Least Model h_P). Let P be a ground h -program and H_P be the set of all probabilistic models satisfying P . Then, $h_P = \otimes_t \{h \mid h \in H_P\}$ is the least probabilistic model satisfying P .

3.3 Fixpoint Semantics

Associated with each h -program, P , is an operator, T_P , called the fixpoint operator which takes a hybrid formula function as an argument and returns a hybrid formula function.

Definition 18. Let P be a ground h -program and h be a hybrid formula function. The fixpoint operator T_P is a mapping $T_P : \text{HFF} \rightarrow \text{HFF}$ which is defined as follows:

1. If F is atomic then

$$T_P(h)(F) = c_{\rho_F}(c_{\rho_F}\{md_\rho(\mu) \mid \mu, \rho \in M_1^h\}, S_P(h)(F)).$$
2. If $F = F_1 *_\rho \dots *_\rho F_n$ is not atomic then

$$T_P(h)(F) = c_{\rho_F}(S_P(h)(F), c_{\rho_F}(c_{\rho_F}\{c_\rho(T_P(h)(G), T_P(h)(H)) \mid G \oplus_\rho H = F\}, c_{\rho_F}\{md_\rho(\mu) \mid \mu, \rho \in M_2^h\})).$$

Lemma 4. The S_P operator is monotonic and continuous.

Definition 19. Let P be a ground h -program. Then:

1. $T_P \uparrow 0 = h_\perp$ where h_\perp is the mapping $h_\perp : \text{bf}_S(B_P) \rightarrow [0, 0]$.
2. $T_P \uparrow \alpha = T_P(T_P \uparrow (\alpha - 1))$ where α is a successor ordinal whose immediate predecessor is $(\alpha - 1)$.
3. $T_P \uparrow \omega = \text{lub}\{T_P \uparrow \alpha \mid \alpha < \omega\}$ where ω is a limit ordinal.

Theorem 1. The T_P operator is monotonic and continuous.

Proposition 1. *Let P be an h-program and h be hybrid formula function. Then h is a model of P iff $T_P(h) \leq_t h$.*

Theorem 2. *Let P be an h-program. Then, $h_P = lfp(T_P)$.*

To illustrate how the new semantics is more sophisticated and intuitive, let us reconsider the robot gripper planning example, introduced earlier in section 1, P , which can be encoded as the following h-program.

Example 3.

$$\begin{aligned}
(\text{holdBlock} : [0.95 \times V, 0.95 \times V]) &\leftarrow \text{pickup} : [1, 1], \text{gripperDry} : [V, V]; \text{med} \\
(\text{holdBlock} : [0.5 \times V, 0.5 \times V]) &\leftarrow \text{pickup} : [1, 1], \text{not_gripperDry} : [V, V]; \text{med} \\
(\text{not_gripperDry} : [1 - V, 1 - V]) &\leftarrow \text{gripperDry} : [V, V]; _ \\
(\text{pickup} : [1, 1]) &\leftarrow; _ \\
(\text{gripperDry} : [0.7, 0.7]) &\leftarrow; _
\end{aligned}$$

Since, having a disjunctive p-strategy in the last three h-rules is irrelevant, the underscore symbol is substituted. According to the new semantics of hybrid probabilistic programs, the least fixpoint of P assigns $[1, 1]$ to pickup , $[0.7, 0.7]$ to gripperDry , $[0.3, 0.3]$ to not_gripperDry , and $[0.815, 0.815]$ to holdBlock . This is because by applying the first h-rule in the program, holdBlock is concluded with the probability interval $[0.665, 0.665]$ and by applying the second h-rule, holdBlock is concluded with the probability interval $[0.15, 0.15]$. But, since the disjunctive mutual exclusion p-strategy is associated to the hybrid basic formula holdBlock , holdBlock is assigned $c_{\text{med}}\{[0.665, 0.665], [0.15, 0.15]\} = c_{\text{med}}([0.665, 0.665], [0.15, 0.15]) = [0.665 + 0.15, 0.665 + 0.15] = [0.815, 0.815]$ which coincides with the actual solution of the problem.

In general, the upward iteration operator does not always terminate. Consider the following example adapted from [12].

Example 4 ([12]).

$$\begin{aligned}
(r : [0.5, 0.5]) &\leftarrow; _ \\
(q : [0.6, 0.6]) &\leftarrow; \text{ind} \\
(q : [V_1 \times V_2, V_1 \times V_2]) &\leftarrow r : [V_1, V_1], q : [V_2, V_2]; \text{ind}
\end{aligned}$$

In the first iteration $T_P \uparrow 1$ assigns $[0.5, 0.5]$ to r , $[0.6, 0.6]$ to q . $T_P \uparrow 2$ assigns $[0.72, 0.72]$ to q while $T_P \uparrow 2(r)$ is the same. After the third iteration $T_P \uparrow 3(q) = [0.744, 0.744]$. This process infinitely continues in which a better approximation of q is obtained in each iteration. Similar to [12], this problem arises from the existence of a cyclic inference of the same hybrid basic formula with a slightly higher probability interval. This problem does not arise with h-programs that satisfy the *finite termination property* [12, 13, 25, 26, 5].

Definition 20 ([5]). *Let P be an h-program. P is said to satisfy the the finite termination property iff*

$$(\forall F \in bfs(B_P))(\exists n < \omega)(lfp(T_P)(F) = T_P \uparrow n(F)).$$

If P satisfies the the finite termination property, then the least fixpoint is guaranteed in a finite number of steps. Obviously, all non-recursive programs has the finite termination property [12].

4 Probabilistic Implication-Based Approach

To show how the probabilistic implication-based framework presented in [17] is subsumed by the new hybrid probabilistic programs framework, we follow the outline of [13]. Assume that probabilities in h-programs can be represented by a pair of probabilistic intervals, where the various p-strategies used throughout this paper can be straightforwardly extended to cope with a pair of probabilistic intervals in a similar way as in [17]. Let L_C be the set $C[0, 1] \times C[0, 1]$. A probabilistic rule (p-rule) in the probabilistic IB framework [17] is an expression of the form $(r; \mu_r, \mu_p)$ where $r \equiv H \stackrel{c}{\leftarrow} A_1, \dots, A_n, H, A_1, \dots, A_n$ are atoms, $c \in L_C$, μ_r is the mode of combination associated to the body using the connective \wedge (conjunctive p-strategy) among the atoms in the body of r , and μ_p is the mode of combination associated to the head using the connective \vee (disjunctive p-strategy). A probabilistic program (p-program) is a finite set of p-rules such that p-rules with the same heads in the p-program, the mode of combination associated to their heads is the same. Let P be a p-program and B_P be its Herbrand base. The notion of interpretation in the probabilistic IB framework [17] is defined as a mapping $I : B_P \rightarrow L_C$. Associated with each p-program, P , is an operator T_P^{IB} that maps an interpretation to an interpretation which is defined as

$$T_P^{IB}(I)(H) = \vee_{\mu_p} \{c \wedge_{\mu_r} I(A_1) \wedge_{\mu_r} \dots \wedge_{\mu_r} I(A_n) | (H \stackrel{c}{\leftarrow} A_1, \dots, A_n; \mu_r, \mu_p) \text{ is a ground instance of p-rule in } P\}.$$

Computationally $\vee_{\mu_p} \{c \wedge_{\mu_r} I(A_1) \wedge_{\mu_r} \dots \wedge_{\mu_r} I(A_n) | (H \stackrel{c}{\leftarrow} A_1, \dots, A_n; \mu_r, \mu_p) \text{ is a ground instance of p-rule in } P\}$ is equivalent to $c_{\mu_p d} \{c_{\mu_r c} \{c, I(A_1), \dots, I(A_n)\} | (H \stackrel{c}{\leftarrow} A_1, \dots, A_n; \mu_r, \mu_p) \text{ is a ground instance of p-rule in } P\}$. The T_P^{IB} operator is monotonic and continuous.

Definition 21 (Translation). *Let P be a p-program. Then P can be translated into an h-program, $T_r(P)$, where*

$$T_r(P) = \{(H : c_{\mu_r c}(c, \mu_1, \dots, \mu_n) \leftarrow A_1 : \mu_1, \dots, A_n : \mu_n; \mu_p) | (H \stackrel{c}{\leftarrow} A_1, \dots, A_n; \mu_r, \mu_p) \in P \text{ and } \mu_1, \dots, \mu_n \text{ are variable annotations}\}.$$

The following theorem establishes the relation between the probabilistic IB framework and the new hybrid probabilistic programs framework given the assumption that, without a loose of generality, an interpretation in the new hybrid probabilistic programs framework is a mapping $h : B_P \rightarrow L_C$.

Theorem 3. *Let P be a p-program. Then $T_{T_r(P)} = T_P^{IB}$.*

5 Conclusions

The language of hybrid probabilistic programs has been extended as well as a new semantics for the extended language has been introduced. Having such a new language and semantics is important in order to enable more realistic applications including those in probabilistic AI planning. In addition, it has been shown that the T_P operator of the new hybrid probabilistic programs semantics is always continuous. We have also illustrated that Lakshmanan and Sadri [17] approach for probabilistic logic programming can fit in the proposed framework. A topic of future research is to extend the new hybrid probabilistic programs framework with non-monotonic negation giving the underlying semantics are the stable model semantics and the well-founded semantics.

References

1. H. A. Blair and V. S. Subrahmanian. Paraconsistent Foundations for Logic Programming. *Journal of Non-Classical Logic*, 5(2):45-73, 1989.
2. G. Boole, G. *The Laws of Thought*. Mcmillan, London, 1854.
3. R. Carnap. *The Logical Foundations of Probability*. University of Chicago Press, 1962. 2nd. Edn.
4. P. Cheeseman. In Defense of Probability. *In Proc. of IJCAI*, pages 1002-1009, 1985.
5. A. Dekhtyar and V. S. Subrahmanian. Hybrid Probabilistic Program. *In Proc. 14th Intl. Conf. on Logic Programming*, Leuven, Belgium, July 8-12 1997.
6. D. Dubois, J. Lang, and H. Prade. Towards Possibilistic Logic Programming. *In Proceedings of the Eighth ICLP*, pages 581-595, Paris, France, June 1991. MIT Press
7. R. Fagin, J. Halpern, and N. Meggido. A Logic for Reasoning about Probabilities. *Information and Computation*, 87(1/2):78-128, 1992.
8. M. C. Fitting. Logic Programming on A Topological Bilattice. *Fundamenta Informaticae*, 11:209-218, 1988.
9. M.C. Fitting. Bilattices and The Semantics of Logic Programming. *Journal of Logic Programming*, 11:91-16, 1991.
10. B. V. Gnedenko and A. Y. Khinchin. *An Elementary Introduction to the Theory of Probability*. Dover Publications, 1962
11. J. Y. Halpern, J.Y. An analysis of First-Order Logics of Probability. *Journal of AI*, 46:311-350, 1990.
12. M. Kifer and A. Li. On The Semantics of Rule-Based Expert Systems with Uncertainty. In M. Gyssens, J. Paradaens, and D. van Gucht, editors, *In Proc. 2nd Intl. Conf. on Database Theory*, pages 102-117, Bruges, Belgium, August 31-September 2 1988. Springer-Verlag LNCS-326.
13. M. Kifer and V. S. Subrahmanian. Theory of Generalized Annotated Logic Programming and Its Applications. *Journal of Logic Programming*, 12:335-367, 1992.
14. A. N. Kolmogorov. *Foundations of the Theory of Probability*. Chelsea Publishing Co, 1956.
15. N. Kushmerick, S. Hanks, and D. Weld. An Algorithm for Probabilistic Planning. *Artificial Intelligence*, 76(1-2):239-286, 1995.

16. V. S. L. Lakshmanan and F. Sadri. Modeling Uncertainty in Deductive Databases. *In Proc. Intl. Conf. on Database Expert Systems and Applications (DEXA '94)*, Athens, Greece, September 1994. Springer-Verlag, LNCS-856.
17. V. S. L. Lakshmanan and F. Sadri. Probabilistic Deductive Databases. *In Proc. Intl. Logic Programming Symposium*, pages 254-268, Ithaca, NY, November 1994. MIT Press.
18. V. S. L. Lakshmanan. An Epistemic Foundation for Logic Programming with Uncertainty. *In Proc. Intl. Conf. on Foundations of Software Technology and Theoretical Computer Science*, Madras, India, December 1994. Springer Verlag. Lecture Notes in Computer Science, vol. 880.
19. L. V. S. Lakshmanan and F. Sadri. Uncertain Deductive Databases: A Hybrid Approach. *Information Systems*, 22(8):483-508, December 1997.
20. V. S. L. Lakshmanan and N. Shiri. A Parametric Approach to Deductive Databases with Uncertainty. *IEEE Transactions on Knowledge and Data Engineering*, 13(4):554-570, 2001.
21. S. M. Leach and J. J. Lu. Query Processing in Annotated Logic Programming: Theory and Implementation. *Journal of Intelligent Information Systems*, 6(1):33-58, January 1996.
22. Y. Loyer and U. Straccia. Default Knowledge in Logic Programs with Uncertainty. *In Proceedings of the 19th International Conference on Logic Programming*, 2003.
23. Y. Loyer and U. Straccia. The Approximate Well-founded Semantics for Logic Programs with Uncertainty. *In Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science*, 2003.
24. J. J. Lu and S. M. Leach. Computing Annotated Logic Programs. *In Proceedings International Conference on Logic Programming*, Pascal van Hentenryck (ed) MIT press, Cambridge, MA, 1994.
25. R. T. Ng and V. S. Subrahmanian. Probabilistic Logic Programming. *Information and Computation*, 101(2):150-201, December 1992.
26. R. T. Ng and V. S. Subrahmanian. A Semantical Framework for Supporting Subjective and Conditional Probabilities in Deductive Databases. *Automated Reasoning Journal*, 10(2):191-235, 1993.
27. R. T. Ng and V. S. Subrahmanian. Stable Semantics for Probabilistic Deductive Databases. *Information and Computation*, 110(1):42-83, 1994.
28. N. Nilsson. Probabilistic Logic. *AI Journal*, 28:71-87, 1986.
29. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
30. E. Shapiro. Logic Programs with Uncertainties: A Tool for Implementing Expert Systems. *In Proc. of IJCAI*, pages 529-532, 1983.
31. V. S. Subrahmanian. On The Semantics of Quantitative Logic Programs. *In Proc. 4th IEEE Symposium on Logic Programming*, pages 173-182, Computer Society Press, Washington DC, 1987.
32. M. H. van Emden. Quantitative Deduction and Its Fixpoint Theory. *Journal of Logic Programming*, 4(1):37-53, 1986.