Revisiting the Semantics of Interval Probabilistic Logic Programs

Alex Dekhtyar,

Department of Computer Science
University of Kentucky
dekhtyar@cs.uky.edu

Michael I. Dekhtyar

Department of Computer Science
Tver State University
Michael.Dekhtyar@tversu.ru

Abstract. Two approaches to logic programming with probabilities emerged over time: bayesian reasoning and probabilistic satisfiability (PSAT). The attractiveness of the former is in tying the logic programming research to the body of work on Bayes networks. The second approach ties computationally reasoning about probabilities with linear programming, and allows for natural expression of imprecision in probabilities via the use of intervals.
In this paper we construct precise semantics for one PSAT-based formalism for reasoning with inteval probabilities, probabilistic logic programs (p-programs), orignally considered by Ng and Subrahmanian. We show that the probability ranges of atoms and formulas in p-programs cannot be expressed as single intervals. We construct the prescise description of the set of models of p-programs and study the computational complexity if this problem, as well as the problem of consistency of a p-program. We also study the conditions under which our semantics coincides with the single-interval semantics originally proposed by Ng and Subrahmanian for p-programs. Our work sheds light on the complexity of construction of reasoning formalisms for imprecise probabilities and suggests that interval probabilities alone are inadequate to support such reasoning.

1 Introduction

Reasoning with probabilistic information, in the context of logic programming, has two distinct origins: bayesian reasoning and probabilistic satisfiability. The former is based on interpreting statements about conditional probability of event A given event B as an implication of a special kind (if B then the probability of A is equal to p). Among the logic programming frameworks following this idea are the work of Poole[15], Ngo and Haddawy [13], and more recently, and in the context of answer set programming, of Baral, Gelfond and Rushton [2].

The second approach to reasoning with probabilistic information starts with Porbabilistic Satisfiability (PSAT), a problem originally formulated by Boole in [1] and "resurrected" by Georgakopoulos, Kavvadis and Papadimitriou[8] more than a century later. PSAT is the problem of determining, whether a set $\{P(F) = p_F\}$, of assignments of probabilities to a collection $\mathcal{F} = \{F\}$ of boolean formulas over atomic events is consistent, i.e., whether there exists a way to assign probabilities to all atomic events in a way that $P(F) = p_F$ for all formulas F in $\mathcal{F}$. Nilsson's probabilistic logic[14] is based on PSAT and uses the semantics of possible worlds (world probability functions) to model probabilities of events. In [8] it is shown that PSAT is NP-complete.

The attractiveness of building logic programming frameworks based on bayesian reasoning lies in direct relationship to the large body of work on Bayesian networks and Markov Decision Processes. The attractiveness of PSAT-based logic programs is in the fact that PSAT has a natural extension to the case of imprecise probabilities. The importance of imprecise probabilities has been observed by numerous researchers in the past 10-15 years [16, 3] and lead to the establishment of the Imprecise Probabilities Project [9].

Interval PSAT is a reformulation of PSAT, in which probability assignments of the form $P(F) = p_F$ are relplaced with inequalities of the form $l_F \leq P(F) \leq u_F$. The underlying semantics and the methodology for solving Interval PSAT is the same as for PSAT. Logic programming frameworks inspired by PSAT consider rules of the form "$P(F) = \mu$ if $P(F_1) = \mu_1$ and $\ldots$ and $P(F_n) = \mu_n$". Unlike in bayesian-inspired frameworks, here "if" is the classical logical implication. Logic programming formalisms stemming from PSAT, in which probabilities of events are expressed as intervals, have been considered by Ng and Subrahmanian[10, 11] and by Dekhtyar and Subrahmanian[6]. In these frameworks, the fixpoint semantics of formulas, i.e., the set of possible probability assignments for them, had been represented using a single interval.

In [5] we have established that even for simple logic programs (a subset of programs considered by [10]), which contain only atomic events in heads and bodies, the single-interval fixpoint *does not adequately describe* the exact set of possible probability assignments. We have shown that the "real" possible-world semantics is a union of a set of sub-intervals of [0,1].

In this paper, we extend the results of [5] onto the general case of propositional interval probabilistic logic programs as defined in [11]. We formally define the propositional interval probabilistic logic programs of [11][1] in Section 2, where we also show that the single-interval fixpoint is not precise. In Section 3 we provide the precise description of the set of models for an interval logic program. In Section 4 we address the problem of determining if an interval logic program has a model. In Section 5 we study the problem of when the single-interval fixpoint describes all the models of an interval logic program precisely, and prove a number of sufficient conditions.

2 Interval Probabilistic Logic Programs

2.1 Syntax

In this section we describe interval Probabilistic Logic Programs of Ng and Subrahmanian [10, 11]. Let L be some first order language containing infinitely many variable symbols, finitely many predicate symbols and no function symbols. Let $B_L = \{A_1, \ldots, A_N\}$ be the Herbrand base of L. A *basic formula* is either an atom from B_L or a conjunction or disjunction of two or more atoms. The set of all basic formulas is denoted $bf(B_L)$. Formulas of the form $(B_1 \wedge \ldots \wedge B_n) : \mu$ and $(B'_1 \vee \ldots \vee B'_m) : \mu'$,

[1] Ng and Subrahmanian consider in [11] probabilistic logic programs with variables in the probability intervals. In this paper, we consider only constant probability intervals, leaving the rest of the syntax from [11] the same.

where $B_1, \ldots, B_n, B'_1, \ldots, B'_m \in B_L$ and $\mu = [l, u], \mu' = [l', u'] \subseteq [0, 1]$ are called *p-annotated conjunctions* and *p-annotated disjunctions* respectively.

P-annotated conjunctions and disjunctions represent probabilistic information. Every atom in B_L is assumed to represent an (uncertain) event or statement. A *p-annotated conjunction* $A_1 \wedge \ldots \wedge A_n : [l, u]$ is read as "the probability of the joint occurrence of the events corresponding to $A_1, \ldots, A_n$ lies in the interval $[l, u]$". Similarly, $A_1 \vee \ldots \vee A_n : [l, u]$ is read as "the probability of the occurrence of at least one of the events corresponding to $A_1, \ldots, A_n$ lies in the interval $[l, u]$".

Probabilistic Logic Programs (p-programs) are constructed from *p-annotated* formulas as follows. Let $F, F_1, \ldots, F_n$ be some basic formulas and $\mu, \mu_1, \ldots, \mu_n$ be subintervals of $[0, 1]$ (also called *annotations*). Then, a *p-clause* is an expression of the form $F : \mu \longleftarrow F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n$ (if $n = 0$, as usual, the *p-clause* $F : \mu \longleftarrow$ is referred to as a *fact*). A Probabilistic Logic Program (*p-program*) is a finite collection of *p-clauses*. In this paper, we call a p-program in which all clauses consist of atoms from B_L only *a simple p-program*[5]. We also call a p-program in which the heads of all clauses are atoms from B_L a *factored p-program*. Given a p-program P, we denote the set of basic formulas found in it as $bf(P)$.

In [10] Ng and Subrahmanian considered factored p-programs. In [12] they considered a framework, in which variables were allowed in the probability annotations. Our definition of p-programs allows arbitrary heads of p-clauses, but does does not consider variable annotations.

2.2 Model Theory

The model theory assumes that in the real world each atom from B_L, and therefore each basic formula, is either true or false. However, exact information about the real world is not known. The uncertainty about the world is represented in a form of a probability distribution over the set of 2^n possible worlds. In addition, p-programs introduce uncertainty about the probability distribution itself.

More formally, given B_L, a world probability density function WP is defined as $WP : 2^{B_L} \rightarrow [0, 1]$, $\sum_{W \subseteq 2^{B_L}} WP(W) = 1$. Each subset W of B_L is considered to be a *possible world* and WP associates a point probability with it. $W \models A$ iff $A \in W$; $W \models A_1 \wedge \ldots \wedge A_n$ iff $(\forall 1 \leq i \leq n) W \models A_i$ and $W \models A_1 \vee \ldots \vee A_n$ iff $(\exists 1 \leq i \leq n) W \models A_i$. We fix an enumeration $W_1, \ldots W_M$, $M = 2^N$ of the possible worlds and denote $WP(W_i)$ as p_i.

Given a function WP, *probabilistic interpretation* (*p-interpretation*) I_{WP} is defined on the set of all basic formulas as follows: $I_{WP} : bf(B_L) \rightarrow [0, 1]$, $I_{WP}(F) = \sum_{W \models F} WP(W)$[2]. P-interpretations assign probabilities to basic formulas by adding up the probabilities of all worlds in which they are true. We note that the mapping from world probability density functions onto p-interpretations is many-to-one: given WP, I_{WP} is defined uniquely, but different world probability density functions can yield the same p-interpretation I.

[2] Note, that each world probability density function WP has a unique p-interpretation I_{WP} associated with it. However, in general, a p-interpretation I can be induced by more than one world probability density function.

P-interpretations specify the model-theoretic semantics of p-programs. Given a p-interpretation I, the following definitions of satisfaction are given:
– $I \models F : \mu$ **iff** $I(F) \in \mu$;
– $I \models F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n$ **iff** $(\forall 1 \leq i \leq n)(I \models F_i : \mu_i)$;
– $I \models F : \mu \longleftarrow F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n$ **iff** either $I \models F : \mu$ or $I \not\models F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n$.

Now, given a p-program P, $I \models P$ (I *is a model of* P) iff for all p-clauses $C \in P$, $I \models C$. Let $Mod(P)$ denote the set of all models of p-program P. It is convenient to view a single p-interpretation I as a point $(I(F_1), \ldots, I(F_M))$ in $M = 2^N$-dimensional unit cube E^M. Then, $Mod(P)$ can be viewed as a subset of E^M. P is called *consistent* iff $Mod(P) \neq \emptyset$, otherwise P is called *inconsistent*.

2.3 Interval Fixpoint

In this section we give a brief definition of the fixpoint semantics proposed in [10]. The fixpoint semantics of defined on *atomic functions* and *formula functions*.

Let $\mathcal{C}[0,1]$ denote the set of all subintervals of the interval $[0,1]$. An atomic function is a mapping $f : B_L \rightarrow \mathcal{C}[0,1]$. A formula function h is a mapping $h : bf(B_L) \rightarrow \mathcal{C}[0,1]$. Given a set $\mathcal{F} \subseteq bf(B_L)$ a *restricted formula function* is a mapping $f_{\mathcal{F}} : \mathcal{F} \rightarrow \mathcal{C}[0,1]$. Intuitively atomic and formula functions assign probability intervals to atoms and basic formulas: $h(F) = [l,u]$ can be interpreted as the statement ``probability of formula F lies in the interval $[l,u]$".

Each formula function $h_{\mathcal{F}}$ induces a set $\mathcal{LL}(h_{\mathcal{F}})$ of linear inequalities on the probabilities $p_1, \ldots, p_M$ of possible worlds. $\mathcal{LL}(h_{\mathcal{F}})$ consists of the following types of inequalities:

- $l_F \leq \sum_{W_j \models F} p_j \leq u_F$, for all $F \in \mathcal{F}$, $h_{\mathcal{F}}(F) = [l_F, u_F]$;
- $\sum_{j=1}^{M} p_j = 1$;
- $p_j \geq 0$, for all $1 \leq j \leq M$.

Given a p-program P, two operators, S_P and T_P are defined. They map formula functions to formula functions in the following manner. For a basic formula F, $S_P(h)(F) = \cap M_F$, where $M_F = \{\mu | F : \mu \longleftarrow F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n \in P$, and $(\forall 1 \leq i \leq n)(h(F_i) \subseteq \mu_i)\}$. If $M_F = \emptyset$ then $S_P(h)(F) = [0,1]$. The T_P operator is defined as follows: $T_P(h)(F) = [l_F, u_F]$, where $l_F = \min\left(\sum_{W_j \models F} p_j\right)$, subject to $\mathcal{LL}(S_P(h))$ and $u_F = \max\left(\sum_{W_j \models F} p_j\right)$, subject to $\mathcal{LL}(S_P(h))$.

Intuitively, S_P computes the intervals of formulas based on the p-clauses that fired. However, because basic formulas are not, in general, independent (e.g. such formulas as $a \wedge b$ and $a \wedge c$), the ranges computed by S_P may need tightening, performed by T_P. The work of these operators is illustrated on the following example.

Example 1. Consider the p-program P_1 shown in Figure 1. Let $h(F) = [0,1]$ for all $F \in bf_(B_L)$. $S_P(h)(a \wedge b) = [0, 0.5] \cap [0.5, 1] = [0.5, 0.5]$. $S_P(h)(a \wedge c) = [0.5, 0.5]$; $S_P(h)(b \wedge c) = [0.5, 0.5]$ and $S_P(h)(a \wedge b \wedge c) = [0.1, 0.2]$. To compute $T_P(h)$ we first construct $\mathcal{LL}(S_P(h)$. Let $W_1 = \{a,b,c\}$, $W_2 = \{a,b\}$, $W_3 = \{a,c\}$ and

P_1 :	$p_1 + p_2 = 0.5$
$C_1 : (a \wedge b) : [0.5, 1] \longleftarrow .$	$p_1 + p_3 = 0.5$
$C_2 : (a \wedge b) : [0, 0.5] \longleftarrow .$	$p_1 + p_4 = 0.5$
$C_3 : (a \wedge c) : [0.5, 0.5] \longleftarrow .$	$0.1 \leq p_1 \leq 0.2$
$C_4 : (b \wedge c) : [0.5, 0.5] \longleftarrow .$	$p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8 = 1$
$C_5 : (a \wedge b \wedge c) : [0.1, 0.2] \longleftarrow .$	$p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8 \geq 0$

Fig. 1. Sample p-program P_1, and the set of inequalities $\mathcal{LL}(S_P)$ it induces.

P_3 :	P_4 :
$C_1 : (a \wedge b) : [0.2, 0.5] \longleftarrow .$	$C_1 : (a \wedge b) : [0.2, 0.5] \longleftarrow .$
$C_2 : (c \vee d) : [0.4, 0.6] \longleftarrow .$	$C_2 : (c \vee d) : [0.4, 0.6] \longleftarrow .$
$C_3 : (c \vee d) : [0.5, 0.6] \longleftarrow (a \wedge b) : [0.2, 0.4].$	$C_3 : (c \vee d) : [0.7, 0.8] \longleftarrow (a \wedge b) : [0.2, 0.4].$
$C_4 : (c \vee d) : [0.4, 0.5] \longleftarrow (a \wedge b) : [0.4, 0.5].$	$C_4 : (c \vee d) : [0.7, 0.8] \longleftarrow (a \wedge b) : [0.4, 0.5].$

Fig. 2. Fixpoint does not describe exactly all models for p-programs P_3 and P_4.

$W_4 = \{b, c\}$. The set of inequalities $\mathcal{LL}(S_P)(h)$ is shown in Figure 1 (for simplicity replace constraints of the form $a \leq X \leq a$ with $X = a$).

Combining the first three constraints with the fifth we get $2p_1 - 0.5 = p_5 + p_6 + p_7 + p_8$ or $p_1 = 0.25 + p_5 + p_6 + p_7 + p_8$. Because all $p_i \geq 0$, $\min(p_1)$ subject to the latter constraint is 0.25 (when all p_5,p_6,p_7,$p_8 = 0$). However, this contradicts the fourth constraint above which says, in particular $p_1 \leq 0.2$. Thus, $\mathcal{LL}(h)(S_P)$ has no solutions.

Example 2. Consider the p-program $P_2 = P_1 - \{C_5\}$. The computation of $S_P(h)$ will be the same as in the previous example, except $S_P(h)(a \wedge b \wedge c) = [0, 1]$. Now, $T_P(h)(a \wedge b \wedge c)$ is defined: $\min(p_1)$ subject to $\mathcal{LL}(S_P)(h)$ is 0.25 (see previous example for derivation). $\max(p_1) = 0.5$ and it is reached when $p_2 = p_3 = p_4 = 0$. Thus, $T_P(h)(a \wedge b \wedge c) = [0.25, 0.5]$.

The set of all formula functions over $bf(B_L)$ forms a complete lattice $\mathcal{FF}$ w.r.t. the subset inclusion: $h_1 \leq h_2$ iff $(\forall F \in bf(B_L))(h_1(F) \supseteq h_2(F))$. The bottom element $\perp$ of this lattice is the function that assigns $[0, 1]$ interval to all formulas, and the top element $\top$ is the atomic function that assigns $\emptyset$ to all formulas. Ng and Subrahmanian show that T_P is monotonic [10] w.r.t. $\mathcal{FF}$. The iterations of T_P are defined in a standard way: (i) $T_P^0 = \perp$; (ii) $T_P^{\alpha+1} = T_P(T_P^\alpha)$, where $\alpha + 1$ is the successor ordinal whose immediate predecessor is α; (iii) $T_P^\lambda = \sqcup\{T_P^\alpha | \alpha \leq \lambda\}$, where λ is a limit ordinal. Ng and Subrahmanian show that, the least fixpoint $lfp(T_P)$ of the T_P operator is reachable after a finite number of iterations ([10], Theorem 2). They also show that if a p-program P is consistent, then $\mathcal{I}(lfp(T_P))$, the set of all p-interpretations satisfying $lfp(T_P)$[3], contains $Mod(P)$ ([10] Corollary 3).

[3] $I \models h$ iff for all $F \in bf(B_L)$, $I(F) \in h(F)$ and there exists WP, s.t., WP satisfies $\mathcal{LL}(h)$ and $I = I_{WP}$.

2.4 Fixpoint is not enough

The inverse of the latter statement, however, is not true. We illustrate it on the examples below. There, and elsewhere in the paper, we use the following conventions concerning the possible worlds $W_1, \ldots, W_M$ over which world probability functions are defined. Let $B_L = \{A_1, \ldots, A_N\}$. The mapping of indexes i of worlds W_i to subsets of B_L is the reverse lexicografical order: $W_1 = B_L$, $W_2 = B_L - \{A_N\}, \ldots, W_M = \emptyset$.

Consider now the p-program P_3 in Figure 2.

Proposition 1. *There exists a p-interpretation I, such that $I \models lfp(T_{P_3})$, but $I \not\models P_3$.*

Proof. First, we compute $lfp(T_{P_3})$. On step 1 of the itrative process, $S_{P_3}(\perp)(a \wedge b) = [0.2, 0.5]$ and $S_{P_3}(\perp)(c \vee d) = [0.4, 0.6]$, i.e., clauses C_1 and C_2 of the program will fire. The following constraints are present in $\mathcal{LC}(S_{P_3}(\perp))$.
$0.2 \leq p_1 + p_2 + p_3 + p_4 \leq 0.5$
$0.4 \leq p_1 + p_2 + p_3 + p_5 + p_6 + p_7 + p_9 + p_{10} + p_{11} + p_{13} + p_{14} + p_{15} \leq 0.6$

From these constraints we can find the upper and lower bounds of T_{P_3} on individual atoms. For a we get $l_a = \min(p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8) = 0.2$, while $u_a = \max(p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8) = 1$. The probability range for b, $[l_b, u_b] = [l_a, u_a] = [0.2, 1]$ due to symmetricity of conjunction of two events. Similarly, we can discover that $l_c = l_d = 0$ and $u_c = u_d = 0.6$.

On the second step, no new rules will fire. Indeed, for the p-clause C_3 to fire, we must have $T^1_{P_3}(a \wedge b) \subseteq [0.2, 0.4]$, and for C_4 to fire, it should be $T^1_{P_3}(a \wedge b) \subseteq [0.4, 0.5]$. But $T^1_{P_3}(a \wedge b) = S_P(\perp)(a \wedge b) = [0.2, 0.5]$, which is a subset of neither $[0.2, 0.4]$ nor $[0.4, 0.5]$. Thus, $lfp(T_{P_3}) = T^1_{P_3}$.

We now show, that there exist a p-interpretation I, such that $I \models T^1_{P_3}$ but $I \not\models P_3$. Consider a (partially defined) p-interpretation I, such that $I(a \wedge b) = 0.3$ and $I(c \vee d) = 0.4$. We complete the construction of I to ensure that it satisfies $T^1_{P_3}$ as follows.
$I(a \wedge b) = p_1 + p_2 + p_3 + p_4 = 0.3$
$I(c \vee d) = p_1 + p_2 + p_3 + p_5 + p_6 + p_7 + p_9 + p_{10} + p_{11} + p_{13} + p_{14} + p_{15} = 0.4$

Let $p_1 = p_2 = p_3 = 0.05$, $p_4 = 0.15$, $p_5 = 0.05$, $p_6 = 0.1$, $p_7 = 0.1$, $p_9 = p_{10} = p_{11} = p_{13} = p_{14} = p_{15} = 0$, $p_{12} = 0.15$, $p_{16} = 0.3$. This assignment satisfies all constraints in $\mathcal{LC}S_{P_3}(\perp)$, which means that $I \models T^1_{P_3}$.

$I \models P_3$ iff $I \models C_1$, $I \models C_2$, $I \models C_3$ and $I \models C_4$. We can see easilty that $I \models C_1$ and $I \models C_2$: $I(a \wedge b) = 0.3 \in [0.2, 0.5]$ and $I(c \vee d) = 0.4 \in [0.4, 0.6]$. However, $I \not\models C_3$. Indeed, $I(a \wedge b) = 0.3 \in [0.2, 0.4]$, i.e., the body of C_3 is **satisfied**, but $I(c \vee d) = 0.4 \notin [0.5, 0.6]$, i.e., the head of C_3 is **not satisfied**.■

Proposition 1 shows that not all p-interpretations satisfying $lfp(T_P)$ satisfy the program itself, i.e., $Mod(P) \neq lfp(T_P)$. As it turns out, there exist p-programs with non-empty $lfp(T_P)$ for which $Mod(P) = \emptyset$. One such example is program P_4 shown in Figure 2.

Proposition 2. *$lfp(T_{P_4})$ is not empty, while $Mod(P_4) = \emptyset$.*

Proof. First we show that there are p-interpretations satisfying $lfp(T_{P_4})$. Using reasoning similar to that in the proof of Proposition 1 we see that on the first step of the fix-point computation process, clauses C_1 and C_2 will fire, giving rise to $T_{P_4}(\perp)(a \wedge b) =$

$S_{P_4}(\bot)(a \wedge b) = [0.2, 0.5]$ and $T_{P_4}(\bot)(c \vee d) = S_{P_4}(\bot)(c \vee d) = [0.4, 0.6]$. On the second step neither C_3 nor C_4 will fire as $T_{P_4}(\bot)(a \wedge b) = [0.2, 0.5] \not\subseteq [0.2, 0.4]$ and $T_{P_4}(\bot)(a \wedge b) = [0.2, 0.5] \not\subseteq [0.4, 0.5]$. This means $lfp_{T_{P_4}} = T^1_{P_4} = T_{P_4}(\bot)$, which is not empty and thus contains satisfying p-interpretations.

Now we show that $Mod(P_4) = \emptyset$. Let $I \models P_4$ be a p-interpretation. $I \models P_4$, means $I \models C_1$, $I \models C_2$, $I \models C_3$ and $I \models C_3$. From $I \models C_1$ we obtain $I(a \wedge b) \in [0.2, 0.5]$. From $I \models C_2$ we obtain $I(c \vee d) \in [0.4, 0.6]$. **Now, we observe that** $I(a \wedge b) \in [0.2, 0.5]$ **implies that either** $I(a \wedge b) \in [0.2, 0.4)$ **or** $I(a \wedge b) \in (0.4, 0.5]$ **or** $I(a \wedge b) = 0.4$**.** Consider each case separately.

If $I(a \wedge b) \in [0.2, 0.4)$, then I satisfies the body of C_3. Therefore, it must be the case that $I(c \vee d) \in [0.7, 0.8]$. However, because $I \models C_2$, $I(c \vee d) \in [0.4, 0.6]$ which leads to a contradiction. Similarly, $I(a \wedge b) \in (0.4, 0.5]$ makes the body of C_4 satisfied, and thus $I(c \vee d)$ must be in $[0.7, 0.8]$ contradicting the fact that $I \models C_2$. Finally, if $I(a \wedge b) = 0.4$ then the bodies of both C_3 and C_4, and hence I must satisfy their (identical) heads, leading again to $I(c \vee d) \in [0.7, 0.8]$, which contradicts $I \models C_2$. Thus, no p-interpretation I can satisfy P_4.■

Looking at the proofs of both propositions above we see that the reason for the "bad" behavior of $lfp(T_P)$ lies in the computation of the S_P operator, namely, in the determination when p-claues fire. By definition of S_P, a p-clause C fires if current valuation for each basic formula in the body of the clause is a subinterval of its annotation in the clause. Consider, for example a clause $C : F : \mu \longleftarrow G : \mu'$ and some formula function (valuation) h, such that $h(G) \not\subseteq \mu'$ but $h(G) \cap \mu' \neq \emptyset$. This clause will not fire. However, any p-interpretation $I \models C$ such that $I(G) \in h(G) \cup \mu'$, satisfies the body of the clause, and thus, must satisfy its head, i.e., we must have $I(F) \in \mu$. This extra restriction on the probability range of F is not captured by the S_P computation.

3 Possible Worlds Semantics

We ask ourselves: given a p-program P, how do we give an exact description of $Mod(P)$? In [5] we have answered this question of simple p-programs, i.e., p-programs with only atoms in the program clauses. In this section we extend the new semantics to the full case of p-programs.

Definition 1. *Let P be a p-program over the Herbrand base $B_L = \{A_1, \ldots, A_N\}$, and let $\mathcal{W} = (W_1, \ldots, W_M)$, $M = 2^M$ be an enumeration of all subsets of B_L. With each W_j, $1 \leq j \leq M$ we associate a variable p_j with domain $[0, 1]$. Let C be a p-clause in P of the form $F : [l, u] \longleftarrow F_1 : [_1, u_1] \wedge \ldots \wedge F_n : [l_n, u_n]$.*

The family of systems of inequalities induced by *C, denoted $INEQ(C)$ is defined as follows:*

- $n = 0$ *(C is a fact).* $INEQ(C) = \left\{ l \leq \sum_{W_j \models F} p_j \leq u \right\}$.
- $n \geq 1$ *(C is a rule).* $INEQ(C) = T(C) \cup F(C)$;
  $T(C) = \left\{ \left\{ l \leq \sum_{W_j \models F} p_j \leq u; l_i \leq \sum_{W_j \models F_i} p_j \leq u_i | 1 \leq i \leq k \right\} \right\}$;
  $F(C) = \left\{ \left\{ \sum_{W_j \models F_i} p_j < l_i \right\} | 1 \leq i \leq k \right\} \cup \left\{ \left\{ \sum_{W_j \models F_i} p_j > u_i \right\} | 1 \leq i \leq k \right\}$.

Let $P = \{C_1, \ldots, C_s\}$. Then, $INEQ(P)$ is defined as follows:
$INEQ(P) = \{\alpha_1 \cup \ldots \cup \alpha_s | \alpha_i \in INEQ(C_i), 1 \leq i \leq s\}$

Informally, $INEQ(P)$ is constructed as follows: for each p-clause C in the program we select the reason, why it is true. The reason/evidence is either the statement that the head of the clause is satisfied, or that one of the conjuncts in the body is not. The set $INEQ(P)$ represents all possible systems of such evidence/restrictions on probabilities of basic formulas. Solutions of any system of inequalities in $INEQ(P)$ satisfy every clause of P. Of course, not all individual systems of inequalities have solutions, but $INEQ(P)$ *captures all the systems that do*, as shown in the following lemma and theorem.

Lemma 1. *Let C be a p-clause and I be a $p-interpretation$ (both over the same Herbrand Base B_L). Then $I \models C$ iff there exists a world probability function WP, such that $I = I_{WP}$ and $\{p_j = WP(W_j) | W_j \subseteq B_L\} \in Sol(\alpha)$ for some $\alpha \in INEQ(C)$.*

Theorem 1. *A p-interpretation I is a model of a simple p-program P **iff** there exists a world probability function WP, such that $I = I_{WP}$, and a system of inequalities $\alpha \in INEQ(P)$ such that $\mathcal{P} = \{p_j = WP(W_j) | W_j \subseteq B_L\} \in Sol(\alpha)$.*

This leads to the following description of $Mod(P)$:

Corollary 1. $Mod(P) = \bigcup_{\alpha \in INEQ(P)} \{I_{WP} | WP \in Sol(\alpha)\}$

Let $Rules(P)$ and $Facts(P)$ denote the sets of p-clauses from P with non-empty and empty bodies respectively. Let $f(P) = |Facts(P)|$ and $r(P) = |Rules(P)|$. Finally, let $k(P)$ be the maximum number of basic formulas in a body of a rule in P.

The solution of each system $\alpha \in INEQ(P)$ is a convex $M-1$-dimensional[4] (in general case) polyhedron. Given a solution WP of some $\alpha \in INEQ(P)$, I_{WP} is obtained via a linear transformation. Because linear transformations preserve convexity of regions, we can make the following statement about the geometry of the set $Mod(P)$.

Corollary 2. *Given a p-program P over the Herbrand base $B_L = \{A_1, \ldots, A_N\}$, $Mod(P)$ is a union of $S \leq (2k(P)+1)^{r(P)}$, not necessarily disjoint, convex polyhedra. Each polyhedron has a dimensionality of at most $M - 1 = 2^N - 1$.*

This corollary provides an exponental, in the size of the p-program, upper bound on the number of *disjoint* components of $Mod(P)$. In [5] we constructed a simple p-program P with $2N + 1$ clauses and $k(P) = 1$, whose $Mod(P)$ is a collection of 2^N disjoint N-dimensional parallepipeds. This shows that the exponential bound cannot be substantially decreased.

The semantics of p-programs is closely connected to Interval PSAT. As mentioned above, each system of inequalities in $INEQ(P)$ is constructed by selecting one formula from each clause (either the head or from the body) and assigning it an interval: $[l, u]$ for the head; $[0, l_i)$ or $(u_i, 1]$ for the formula F_i from the body. Theorem 1 showed that any assignment of point probabilities to the atoms, that satisfies these constraints is

[4] Because $p_1 + \ldots + p_M = 1$ is present in every $\alpha \in INEQ(P)$, the dimensionality of $Sol(\alpha)$ cannot be more than $M - 1$.

a model of P. At the same time, the set $\{F : \mu\}$ of annotated formulas for which satisfying p-interpretations are to be found is an *instance of Interval PSAT*. Thus, an instance of Interval PSAT is associated with each set of inequalities in $INEQ(P)$. We note that the sets of solutions for individual systems from $INEQ(P)$ are not disjoint, however, each system can contain unique solutions. Thus, one way of computing $Mod(P)$ is to solve $|INEQ(P)|$ Interval PSAT problems.

4 Consistency Problem

The consistency problem for p-programs is defined as follows: given a p-program P, check whether P has a model, i. e. $Mod(P) \neq \emptyset$. Let CONS-P= $\{P|Mod(P) \neq \emptyset\}$.

Theorem 2. *The set CONS-P is NP-complete.*

Proof. *Upper bound.* Let P be a p-program, $B_1, \dots, B_r$ be all basic formulas of P Then $Mod(P) \neq \emptyset$ iff there exist such probabilitues $b_1, \dots, b_r$ of $B_1, \dots, B_r$ that
(i) the system of linear equations and inequalities $EQ(P)$:

- $\sum_{W_j \models B_i} p_j = b_i$, for $i = 1, \dots, r$,
- $\sum_{j=1}^{M} p_j = 1$;
- $p_j \geq 0$, for all $1 \leq j \leq M$.

has a solution $WP = \{p'_1 \dots, p'_M\}$ defined the interpretation $I_{WP} \in Mod(P)$.

To prove the upper bound, we use the following lemma from [7] (which, in turn, cites [4]. Similar statement is also found in [8]).

Lemma 2. *If a system of r linear equations and/or inequalities with integer coefficients each of length at most l has a nonnegative solution, then it has a nonnegative solution with at most r entries positive, and where the size of each member of the solution is $O(rl + r \log r)$.*

Based on this lemma we obtain the following "small model" theorem.

Lemma 3. *p-program P including r different basic formulas is consistent iff there exists a probability distribution WP on possible worlds with no more than $r + 1$ nonzero probabilities such that $I_{WP} \models P$.*

Let the longest number in annotations of P have length l. Then the following nondeterministic procedure allows us to check whether $Mod(P) \neq \emptyset$.
1) Guess for each $B_i (i = 1, \dots, r)$ it's probability $b_i \in [0, 1]$ of the length $O(rl + r \log r)$.
2) Guess a probability distribution WP with no more than $r + 1$ positive probabilities $p_{i_1}, \dots, p_{i_{r+1}}$ of the length $O(rl + r \log r)$ and check that WP is a solution of the system $EQ(P)$.
3) If $I_{WP} \models P$ retur n "Yes".

From the lemmas above it follows that this algorithm runs in nondeterministic time bounded by a polynomial of $|P|$.

Lower bound. We show that 3-CNF $\leq_P$ CONS-P. Let $\varPhi = C_1 \wedge \dots C_m$ be a 3-CNF over the set of boolean variables $Var = \{x_1, \dots, x_n\}$. Let each clause $C_j, j = 1, \dots, m$, include 3 literals l_j^1, l_j^2, l_j^3. Define for each literal l an annotated atom $\alpha(l)$ as follows: if $l = x \in Var$ then $\alpha(l) = x : [0.5, 1]$, if $l = \neg x$ then $\alpha(l) = x : [0, 0.5]$. Let $B_L = Var \cup \{C_j | \, j = 1, \dots, m\} \cup \{\varPhi\}$. We include in p-program $P(\varPhi)$ the following p-clauses. $(f1): \;\; \varPhi : [1.1] \leftarrow .$
$(fc_j): \;\; C_j : [0, 0.1] \leftarrow . \;\; (j = 1, \dots, m)$
$(fx_i): \;\; x_i : [0, 1] \leftarrow . \;\; (i = 1, \dots, n)$
$(rc_j): \;\; C_j : [0.9, 1] \leftarrow \alpha(l_j^1) \wedge \alpha(l_j^2) \wedge \alpha(l_j^3). \;\; (j = 1, \dots, m).$
$(rf_i): \;\; \varPhi : [0, 0] \leftarrow x_i : [0.5, 0.5]. \;\; (i = 1, \dots, n)$

It is easy to see that $P(\varPhi)$ can be constructed from $\varPhi$ in polynomial time. Now the theorem follows from the following proposition.

Proposition 3. *$\varPhi \in$ 3-CNF $\iff$ $P(\varPhi) \in$ CONS-P.*

Proof. Suppose that $\varPhi \in$ 3-CNF and $\sigma : Var \rightarrow \{T, F\}$ is such truth substitution that $\sigma(\varPhi) = T$. Then for each $j = 1, \dots, m$ there is such $k_j, \; 1 \leq k_j \leq 3$, that $\sigma(l_j^{k_j}) = T$. Define an interpretation I as follows: $I(\varPhi) = 1, I(C_j) = 0$ for $j = 1, \dots, m$, $I(x_i) = 0$ if $\sigma(x_i) = T$ and $I(x_i) = 1$ if $\sigma(x_i) = F, \; i = 1, \dots, n$. Then it is easy to see that all facts $(f1), (fc_j), (fx_i)$ and all rules (rf_i) are valid on I. Consider now any rule (rc_j). Its body includes the annotated atom $\alpha(l_j^{k_j})$. If $l_j^{k_j} = x \in Var$ then $\alpha(l_j^{k_j}) = x : [0.5, 1]$. By the choice of $l_j^{k_j}$ we have that

$\sigma(x) = T$, $I(x) = 0$ and therefore $I \not\models \alpha(l_j^{k_j})$. If $l_j^{k_j} = \neg x$ then $\alpha(l_j^{k_j}) = x : [0, 0.5]$. Again, by the choice of $l_j^{k_j}$ we hav e that $\sigma(x) = F$ and $I(x) = 1$ and therefore $I \not\models \alpha(l_j^{k_j})$. We see that in the both cases $I \not\models Body(rc_j)$ and hence, $I \models (rc_j)$. Therefore, $I \models P(\varPhi)$.

Now suppose that there is model I of $P(\varPhi)$. Then fact $(f1)$ implies $I(\varPhi) = 1$, and it follows due to rules (rf_i) that $I(x_i) \neq 0.5$ for $i = 1, \dots, n$. For $x \in$ *Var* we define $\sigma(x) = T$ if $I(x) < 0.5$ and $\sigma(x) = F$ if $I(x) > 0.5$. Show now that each clause C_j, $j = 1, \dots, m$, includes such literal $l_j^{k_j}$ that $\sigma(l_j^{k_j}) = T$. Let us fix any j. The fact (fc_j) implies that inequality $I(C_j) \leq 0.1$ holds. Then the head $C_j : [0.9, 1]$ of the rule (rc_j) is not valid on I. Hence, there is an annotated atom in the body of (rc_j) which does not hold on I. Let it be atom $\alpha(l_j^{k_j})$. If $l_j^{k_j} = x$ for some $x \in$ *Var* then $\alpha(l_j^{k_j}) = x : [0.5, 1]$. Since $I \not\models x : [0.5, 1]$, we get that $I(x) < 0.5$ and $\sigma(l_j^{k_j}) = \sigma(x) = T$. If $l_j^{k_j} = \neg x$ for some $x \in$ *Var* then $\alpha(l_j^{k_j}) = x : [0, 0.5]$. Since $I \not\models x : [0, 0.5]$, we get that $I(x) > 0.5$. Then $\sigma(x) = F$ and $\sigma(l_j^{k_j}) = \neg\sigma(x) = T$. ■

A consistent p-program P *entails* a formula $F : [l, u]$ if for each $I \in Mod(P)$ $I \models F : [l, u]$. The entailment problem is, thus, expressed as follows: given a consistent P and a formula $F : [l, u]$, decide if P entails $F : [l, u]$?

Let $EQ_1(P, F) = EQ(P) \cup \{\sum_{W_j \models F} p_j < l\}$ and $EQ_2(P, F) = EQ(P) \cup \{\sum_{W_j \models F} p_j > u\}$. Then it easy to see that *P does not entail $F : [l, u]$ iff $EQ_1(P, F)$ is solvable or $EQ_2(P, F)$ is solvable.* Therefore we get the following complexity bounds for the entailment problem.

P_5 :	P_6 :
$a : [0.2, 0.4] \longleftarrow .$	$a : [0.2, 0.6] \longleftarrow .$
$b : [0.3, 0.7] \longleftarrow .$	$b : [0.3, 0.7] \longleftarrow .$
$c : [0.8, 0.9] \longleftarrow b : [0.6, 0.9], a : [0.5, 1].$	$c : [0.8, 0.9] \longleftarrow (a \wedge b) : [0.3, 0.6].$

Fig. 3. Programs P_5 and P_6 show that semi-strictness is not the right condition for general p-programs.

Theorem 3. *The enailment problem is co-NP-complete.*

5 When Fixpoint is enough?

In this section we study subclasses of p-programs for which simpler procedures for determining $Mod(P)$ exist. In particular, we ask ourselves a question of when $Mod(P)$, as defined here, and $lfp(T_P)$, as defined in [11] coincide. We then address the problem of complexity of detecting that $Mod(P) = \mathcal{I}(lfp(T_P))$. First, we consider the problem of $Mod(P) = lfp(T_P)$ for the case of simple p-programs.

Definition 2. *A simple p-program P is called* semi-strict *if it satifies the following condition:*
($\star$) For all atoms $A \in B_L$, and for each pair $\mu \in ha_P(a)$ and $\nu \in ba_P(a)$ either $\mu \subseteq \nu$ or $\mu \cap \nu = \emptyset$.

Intuitively, a simple p-program is called semi-strict if for all atoms their annotations in the heads of the rules are either subintervals of annotations in the bodies or do not intersect with them.

Theorem 4. *If P is a simple semi-strict p-program, then $Mod(P) = \mathcal{I}(lfp(T_P))$.*

Theorem 5. *Semi-strictness of a simple p-program P can be checked in time $O(|P|^2)$.*

Semi-strictness is a syntactic condition on simple p-programs, that can be checked in time, quadratic, in the size of the p-program in a straightforward manner. This makes it an attractive condition to use in general case. However, two drawbacks make it impossible. First, this is a sufficient, but not necessary condition, and second, for programs with non-atomic formulas, semi-strictness does not imply $Mod(P) = \mathcal{I}(lfp(T_P))$. The following two examples illustrate these drawbacks.

Example 3. To show that semi-strictness is not a necessary condition, consider p-program P_5 from Figure 3. First, we note that $lfp(T_{P_5})$ assigns intervals $[0.2, 0.4]$, $[0.3, 0.7]$ and $[0, 1]$ to atoms a, b and c respectively. We can also see that the body of the third clause of P_5 is unsatisfiable given the first two clauses, because the intervals for a, $[0.5, 1]$ in the clause and $[0.2, 0.4]$ from the first clause, do not intersect. Therefore, $Mod(P_5)$ will not differ from $lfp(T_{P_5})$. At the same time, we note that P_5 is not semi-strict, because for b the annotation of the head of the second clause, $[0.3, 0.7]$, and the annotation in the body of the third clause, $[0.6, 0.9]$ overlap.

Example 4. Consider the p-program P_6 from Figure 3. P_6 is semi-strict by definition 2. But we can show that $\mathcal{I}(lfp(T_P))$ and $Mod(P)$ differ. Indeed, because the constraints on the probabilities of a and b from the first two p-clauses do not entail the $[0.3, 0.6]$ constraint on the probability of $a \wedge b$, this rule does not fire, and therefore $lfp(T_P)(c) = [0, 1]$. In particular, a p-interpretation I, s.t., $I(a) = 0.6$, $I(b) = 0.7$, $I(c) = 0.2$ is in $\mathcal{I}(lfp(T_P))$. At the same time, if $I(a) = 0.6$ and $I(b) = 0.7$, then $I(a \wedge b) \in [0.3, 0.6]$, and therefore, in the thrid rule, the head must be satisfied, but $0.2 \notin [0.8, 0.9]$.

It turns out that it is possible to specify a sufficient condition in the general case. However, this is no longer a syntactic condition.

Definition 3. *Let P be a p-program and let P' be the result of removing from P all p-clauses whose heads are satisfied by $lfp(T_P)$. A p-program P is called* strict *if the following condition holds:*

> *For each clause $C : F : \mu \longleftarrow F_1 : \mu_1 \wedge \ldots F_n : \mu_n$ in P', there exists an index $1 \leq i \leq n$, such that $lfp(T_P)(F_i) \cap \mu_i = \emptyset$.*

Theorem 6. *If a p-program P is strict, then $Mod(P) = \mathcal{I}(lfp(T_P))$.*

Proof.
We know that $Mod(P) \subseteq \mathcal{I}(lfp(T_P))$. Suppose now, $I \in \mathcal{I}(lfp(T_P))$. We show that $(\forall C : F : \mu \longleftarrow F_1 : \mu_1 \wedge \ldots F_n : \mu_n \in P) I \models C$. If $C \in P - P'$, then $I(F) \in lfp(T_P)(F) \subseteq \mu$, and therefore, $I \models F : \mu$. If $C \in P'$, then, because C is strict, there exists such index i, that $lfp(T_P)(F_i) \cap \mu_i = \emptyset$. Then $I \not\models F_i : \mu_i$, and therefore $I \not\models F : \mu \longleftarrow F_1 : \mu_1 \wedge \ldots F_n : \mu_n$ and $I \models C$.■

For the class of simple p-programs, strictness is a necessary condition.

Theorem 7. *For a simple p-program P, $Mod(P) = \mathcal{I}(lfp(T_P))$ **iff** P is strict.*

The following example shows that strictness is not a necessary condition for non-simple programs.

Example 5. Consider the following p-program P_7:
$a : [0.6, 0.8] \longleftarrow .$
$b : [0.6, 0.7] \longleftarrow .$
$d : [0.2, 0.3] \longleftarrow .$
$c : [0.4, 0.5] \longleftarrow (a \wedge b) : [0.65, 0.7] \wedge (b \vee d) : [0.5, 0.6].$
$lfp(T_P)$ assigns intervals $[0.2, 0.7]$ and $[0.6, 1]$ to $a \wedge b$ and $b \vee d$ respectively, and therefore, P_7 is *not strict*. However, there exists no p-interpretation I which satisfies the first three rules and the body of the fourth rule: $I(b \vee d) \in [0.5, 0.6]$ implies, $I(b \vee d) = 0.6$ and $I(b) = 0.6$, while $I(a \wedge b) \in [0.65, 0.7]$ implies that $I(b) \geq 0.65$. Therefore, $Mod(P)$ coincides with $\mathcal{I}(lfp(T_P))$.

6 Related Work and Conclusions

A survey of different approaches to probabilistic logic programming can be found in [6] and [5]. This paper studies the precise semantics of a logic programming language for reasoning about the interval probabilities of events and their combinations. This language, proposed by Ng and Subrahmanian[11] is a natural extension of Interval Probabilistic Satisfiability problem PSAT [8]: an instance of Interval PSAT is a p-program, in which all rules have no bodies. We show that for this, relatively simple language, the class of satisfying models (probabilistic interpretations) has a complex description: it is a union of a number of (closed, open, semiopen) intervals, obtained, solving an array of Interval PSAT problems. On the positive side, our results show how to compute the set of models of a p-program *precisely*. On the negative side, the complexity of the description and the computational complexity of the problem itself suggest that intervals may be inadequate as the means for specifying imprecision in probabilistic assessments.

References

1. G. Boole. (1854) *The Laws of Thought*, Macmillan, London.
2. Chitta Baral, Michael Gelfond, J. Nelson Rushton. (2004) Probabilistic Reasoning With Answer Sets, in *Proc. LPNMR-2004*, pp. 21-33.
3. Luis M. de Campos, Juan F. Huete, Serafin Moral (1994). Probability Intervals: A Tool for Uncertain Reasoning, *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS)*, Vol. 2(2), pp. 167 – 196.
4. V.Chvátal. (1983) Linear Programming. *W. Freeman and Co.*, San Fancisco, CA.
5. A. Dekhtyar, M.I. Dekhtyar. (2004) Possible Worlds Semantics for Probabilistic Logic Programs, in *Proc., International Conference on Logic Programming (ICLP)'2004, LNCS*, Vol. 3132, pp. 137-148.
6. A. Dekhtyar and V.S. Subrahmanian (2000) Hybrid Probabilistic Programs. *Journal of Logic Programming*, Volume 43, Issue 3, pp. 187 – 250 .
7. R. Fagin J. Halpern, and N. Megiddo. (1990) A logic for reasoning about probabilities, *Information and Computation*, vol. 87, no. 1,2, pp. 78-128.
8. G.Georgakopoulos, D. Kavvadias, C.H. Papadimitriou. (1988) Probabilistic Satisfiability, *Journal of Complexity*, Vol. 4, pp. 1-11.
9. H.E. Kyburg Jr. (1998) Interval-valued Probabilities, in *G. de Cooman, P. Walley and F.G. Cozman (Eds.), Imprecise Probabilities Project*, http://ippserv.rug.ac.be/documentation/interval_prob/interval_prob.html.
10. R. Ng and V.S. Subrahmanian. (1993) Probabilistic Logic Programming, *Information and Computation*, 101, 2, pps 150–201, 1993.
11. R. Ng and V.S. Subrahmanian. A Semantical Framework for Supporting Subjective and Conditional Probabilities in Deductive Databases, JOURNAL OF AUTOMATED REASONING, 10, 2, pps 191–235, 1993.
12. R. Ng and V.S. Subrahmanian. (1995) *Stable Semantics for Probabilistic Deductive Databases*, INFORMATION AND COMPUTATION, 110, 1, pps 42-83.
13. L. Ngo, P. Haddawy (1995) Probabilistic Logic Programming and Bayesian Networks, in *Proc. ASIAN-1995*, pp. 286-300.
14. N. Nilsson. (1986) *Probabilistic Logic*, AI Journal 28, pp 71–87.
15. D. Poole (1993). Probabilistic Horn Abduction and Bayesian Networks. *Artificial Intelligence*, Vol. 64(1), pp. 81-129.
16. Walley, P. (1991). Statistical Reasoning with Imprecise Probabilities. Chapman and Hall, 1991.