Hybrid Probabilistic Programs*

Alex Dekhtyar† and *V.S.Subrahmanian*‡

Department of Computer Science
A. V. Williams Building
University of Maryland
College Park, Maryland 20742, U.S.A.

Abstract

The precise probability of a compound event (e.g. $e_1 \vee e_2, e_1 \wedge e_2$) depends upon the known relationships (e.g. independence, mutual exclusion, ignorance of any relationship, etc.) between the primitive events that constitute the compound event. To date, most research on probabilistic logic programming [12, 11, 13, 14, 15] has assumed that we are ignorant of the relationship between primitive events. Likewise, most research in AI (e.g. Bayesian approaches) have assumed that primitive events are independent. In this paper, we propose a *hybrid* probabilistic logic programming language in which the user can explicitly associate, with any given probabilistic strategy, a conjunction and disjunction operator, and then write programs using these operators. We describe the syntax of hybrid probabilistic programs, and develop a model theory, fixpoint theory, and proof theory for such programs.

1 Introduction

Though there has now been considerable work in the area of *quantitative logic programming* [1, 6, 18, 21, 10], there has been relatively little work in the area of probabilistic logic programming [12, 11, 13, 14, 15]. The reason for this is that while connectives in multivalued logics can be interpreted in terms of the lattice's LUB (for disjunction) and GLB (for conjunction) operators, the same is not true in the case of probabilities. In particular, there is no single "formula" for computing the probability of a complex event $(e_1 \wedge e_2)$ where e_1, e_2 are primitive events. For instance:

1. If e_1, e_2 are *independent*, then $\mathbf{Prob}(e_1 \wedge e_2) = \mathbf{Prob}(e_1) \times \mathbf{Prob}(e_2)$.
2. If we are *ignorant* about the relationship between e_1, e_2, then all we can say[13] is that $\mathbf{Prob}(e_1 \wedge e_2)$ lies in the interval:

$$[\max(0, \mathbf{Prob}(e_1) + \mathbf{Prob}(e_2) - 1), \min(\mathbf{Prob}(e_1), \mathbf{Prob}(e_2))]$$

3. If we know that e_1, e_2 are *mutually exclusive*, then $\mathbf{Prob}(e_1 \wedge e_2) = 0$.
4. If we know that event e_1 implies event e_2 (called *positive correlation*), then $\mathbf{Prob}(e_1 \wedge e_2) = \mathbf{Prob}(e_1)$.

*http://www.cs.umd.edu/users/vs/papers/probpapers.html contains proofs of all results in this paper.
†Electronic Mail: dekhtyar@cs.umd.edu
‡Electronic Mail: vs@cs.umd.edu

The above list represents a small fraction of relationships between events, each leading to different possible probabilities for complex events such as $(e_1 \wedge e_2)$. *The same holds for disjunctive events as well.*

In previous efforts, probabilistic logic programming has assumed a fixed probabilistic strategy[12, 11, 13, 14, 15], such as ignorance. However, an end user writing a probabilistic logic program should have the flexibility to write rules that reflect his/her specific knowledge about dependencies between events. For instance, the user should be able to express statements such as the two given below, that allow the user to explicitly articulate the probabilistic dependencies between events.

- "If the chairman of company C sells his stock and the chairman retires, *and we are ignorant of the dependencies between these two events*, then conclude that the stock in company C will drop, with probability between 40–90%."
- "If the chairman of company C sells his stock and the chairman retires, *and the retirement implies sale of stock* (e.g. in an employee owned company), then conclude that the stock in company C will drop, with probability between 5–20%."

Both rules above refer to the same two events, viz. sale of stock by the chairman, and retirement of the chairman. However, the first rules specifies what to conclude if we are ignorant of the relationship between these two events, while the second explicitly encodes specific knowledge about the dependencies between events. The rules lead to very different conclusions.

In this paper, we make the following contributions:

1. First, we define a general axiomatic notion of a *probabilistic strategy*. We show how a number of well known probabilistic strategies are special cases of our definition.
2. We then define the concept of a *hybrid probabilistic program* (*hp-program*, for short). If the user selects a set of probabilistic strategies $i_1, \ldots, i_k$ for use in an hp-program (s/he may select these in any way, as long as these selections satisfy the axioms defining probabilistic strategies), then this automatically defines a set of conjunction and disjunction connectives.
3. Subsequently, we define a fixpoint semantics for hp-programs, a model theoretic semantics for hp-programs, and a proof procedure, and prove that the fixpoint theory, model theory, and proof theory all lead to equivalent characterizations. *This applies to any selection of probabilistic strategies made by the user, as long as these selections satisfy the axioms defining probabilistic strategies.*

2 Probabilistic Strategies (p-strategies)

In this section, we provide an axiomatic definition of probabilistic strategies (p-strategies). As we have already seen in the Introduction (cf. the ignorance strategy), the probability of a compound event may be an *interval*, rather than a point, even if point probabilities are known for the primitive events involved. Thus, p-strategies will be defined on intervals – points, in any case, are special cases of intervals.

Let $\mathcal{C}[0,1]$ denote the set of all closed intervals of $[0,1]$. Let $\mathcal{PC}[0,1]$ denote the powerset of $\mathcal{C}[0,1]$. If $[a,b] \in \mathcal{C}[0,1], [c,d] \in \mathcal{C}[0,1]$ then we write $[a,b] \leq_t [c,d]$ if $a \leq c$ and $b \leq d$. A probabilistic strategy, defined below, is a pair of functions that satisfy certain properties.

Definition 1 *A probabilistic strategy (p-strategy) is a pair of functions: $i = < c, d >$, such that:*

1. *$c : \mathcal{C}[0,1] \times \mathcal{C}[0,1] \longrightarrow \mathcal{C}[0,1]$ is called a probabilistic composition function satisfying 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 $[a_1, b_1] \subseteq [a_3, b_3]$
2. *$d : \mathcal{C}[0,1] \times \{\vee, \wedge\} \longrightarrow 2^{\mathcal{C}[0,1] \times \mathcal{C}[0,1]}$ is called a probabilistic decomposition function.*

The function c above is a *composition* function that generates a new interval from two input intervals. In contrast, the *decomposition* function d takes an interval, and a connective $(\wedge, \vee)$ as input, and returns as output, a set of pairs. For now, there is no "connection" that ties c and d together: this will be made later through the concept of *coherence* (Definition 3). P-strategies are of two types, depending upon whether they satisfy certain extra axioms.

Definition 2 Conjunctive and Disjunctive p-strategies

- A *p-strategy $< c, d >$ is called* a conjunctive p-strategy if it satisfies the following axioms:
  1. Bottomline : *It is always the case that* $c([a_1, b_1], [a_2, b_2]) \leq_t [min(a_1, a_2), min(b_1, b_2)]$
  2. Identity : $c([a, b], [1, 1]) = [a, b]$
  3. Annihilator : $c([a, b], [0, 0]) = [0, 0]$
- A *p-strategy $< c, d >$ is called* a disjunctive p-strategy if c satisfies the following axioms:
  1. Bottomline : $[max(a_1, a_2), max(b_1, b_2)] \leq_t c([a_1, b_1], [a_2, b_2])$
  2. Identity : $c([a, b], [0, 0]) = [a, b]$
  3. Annihilator : $c([a, b], [1, 1]) = [1, 1]$

Intuitively, a composition function determines, given the probability ranges of two events, the probability range of their (either and- or or-composition). A decomposition function may be thought of as the inverse of composition: given the probability range of the result (and/or-composition of two events) it returns the set of all possible pairs of initial probabilistic ranges for the two events. To ensure that this holds we need the following definition:

Definition 3 A *p-strategy $< c, d >$ is called* coherent if

$$(\forall [a, b] \in \mathcal{C}[0,1])(([a_1, b_1], [a_2, b_2]) \in d([a, b]) \textbf{ iff } c([a_1, b_1], [a_2, b_2]) = [a, b].$$

Throughout the rest of this paper, we will use the expression p-strategy to refer to coherent p-strategies, i.e. only coherent p-strategies will be considered. Before investigating the properties of p-strategies, we present some simple examples below.

2.1 Examples of P-strategies

Independence: The strategy of independence may be described as the conjunctive p-strategy $inc =< c_{inc}, d_{inc} >$ and the disjunctive p-strategy $ind =< c_{ind}, d_{ind} >$, where:

- The conjunctive p-strategy $inc =< c_{inc}, d_{inc} >$ is given by:

$$\begin{aligned} c_{inc}([a_1, b_1], [a_2, b_2]) &= [a_1 a_2, b_1 b_2]. \\ d_{inc}(a, b) &= \{< [a_1, b_1], [a_2, b_2] > \mid (a_1 a_2 = a \ \textbf{and}\ b_1 b_2 = b)\}. \end{aligned}$$

- The disjunctive p-strategy $ind =< c_{ind}, d_{ind} >$ is given by:

$$c_{ind}([a_1, b_1], [a_2, b_2]) = [\min(1, a_1 + a_2 - a_1 a_2), \min(1, b_1 + b_2 - b_1 b_2)]$$

$$d_{ind}([a, b]) \text{ contains } < [a_1, b_1], [a_2, b_2] > \in \mathcal{C}[0, 1] \times \mathcal{C}[0, 1]$$

  iff

$$\textbf{if } a = 1 \textbf{ then } a_1 + a_2 - a_1 a_2 \geq 1$$
$$\textbf{if } a < 1 \textbf{ then } a_1 + a_2 - a_1 a_2 = a$$
$$\textbf{if } b = 1 \textbf{ then } b_1 + b_2 - b_1 b_2 \geq 1$$
$$\textbf{if } b = 1 \textbf{ then } b_1 + b_2 - b_1 b_2 = b$$

Ignorance: When nothing is known about the relationship between the events we are forced to use p-strategies that reflect *ignorance*[13]. $igc =< c_{igc}, d_{igc} >$ below is a *conjunctive* ignorance strategy, while $igd =< c_{igd}, d_{igd} >$ is a disjunctive ignorance strategy.

- Conjunctive ignorance p-strategy
  $igc =< c_{igc}, d_{igc} >$, where

$$c_{igc}([a_1, b_1], [a_2, b_2]) = [\max(0, a_1 + a_2 - 1), \min(b_1, b_2)]$$

$$d_{igc}([a, b]) \text{ contains } < [a_1, b_1], [a_2, b_2] >$$

  iff

$$\textbf{if } a = 0 \textbf{ then } a_1 + a_2 \leq 1$$
$$\textbf{if } a > 0 \textbf{ then } a_1 + a_2 - 1 = a$$
$$(b = b_1 \textbf{ and } b_2 \geq b_1) \textbf{ or } (b = b_2 \textbf{ and } b_1 \geq b_2)$$

- Disjunctive ignorance p-strategy
  $igd =< c_{igd}, d_{igd} >$, where

$$c_{igd}([a_1, b_1], [a_2, b_2]) = [\max(a_1, a_2), \min(1, b_1 + b_2)]$$

$$d_{igd}([a, b]) \text{ contains } < [a_1, b_1], [a_2, b_2] >$$

  iff

$$(a = a_1 \textbf{ and } a_2 \leq a_1) \textbf{ or } (a = a_2 \textbf{ and } a_1 \leq a_2)$$
$$\textbf{if } b = 1 \textbf{ then } b_1 + b_2 \geq 1$$
$$\textbf{if } b < 1 \textbf{ then } b_1 + b_2 = b$$

Proposition 1 *inc and igc are* conjunctive coherent p-strategies. *Similarly, ind and igd are* disjunctive coherent p-strategies.

The full version of this paper, [3], shows how other strategies such as positive correlation, negative correlation, and mutual exclusion may be expressed as appropriate p-strategies. Due to space limitations, we are unable to do so here.

2.2 Properties of P-Strategies

In this section, we define various aspects of p-strategies that will play a key role in the definition of our fixpoint semantics and our model theory. First, we need the following

Claim 1 *Let* $i =< c, d >$ *be a coherent p-strategy. Then a pair* $< [a_1, b_1], [a_2, b_2] > \in d([a, b])$ **iff** $< [a_2, b_2], [a_1, b_1] > \in d([a, b])$

The simple claim above merely assures us that if $< [a_1, b_1], [a_2, b_2] > \in d([a, b])$, then so is $< [a_2, b_2], [a_1, b_1] >$. Given a pair $[a, b]$, the projection set of decomposition function d w.r.t. $[a, b]$ is, the set of all $[a', b']$'s such that $[a', b']$ can be composed with some $[a'', b'']$ via the composition function c to yield $[a, b]$.

Definition 4 *Let* $i =< c, d >$. *The "decomposition projection set"* πD *is defined to be:*

$$\pi D_i([a, b]) = \{[a', b'] \in \mathcal{C}[0, 1] | (\exists [a'', b''] \in \mathcal{C}[0, 1])(< [a', b'], [a'', b''] > \in d([a, b]))\}$$

Intuitively speaking, projection functions are used as follows: suppose we know that the probability of (say) some compound event $(e_1 \wedge e_2)$ lies in the interval $[a, b]$, when $\wedge$ is computed w.r.t. some conjunctive p-strategy $i =< c, d >$. In this case, $\pi D_i([a, b])$ specifies the set of all possible probability intervals for e_1 (and likewise for e_2) that could possibly have led to $(e_1 \wedge e_2)$'s probability interval being $[a, b]$. In other words, in order for $(e_1 \wedge e_2)$'s probability interval to be $[a, b]$, e_1's probability interval must have been an element in $\pi D_i([a, b])$, but we do not know which one.

As a consequence, e_1's probability may be as low as the smallest point in $\bigcup_{[x,y] \in \pi D_i([a,b])} [x, y]$, or as large as the largest member of $\bigcup_{[x,y] \in \pi D_i([a,b])} [x, y]$. This yields an interval for e_1's probability, and motivates the following definition.

Definition 5 *Let* $i =< c, d >$ *be a p-strategy. A "maximal interval"* md *for* $d([a, b])$ *is defined as*

$$md_i([a, b]) = [\min_{[a',b'] \in D_i([a,b])} (a'), \max_{[a',b'] \in D_i([a,b])} (b')].$$

When computing probabilities of primitive events from known probabilities of more complex events, we need to be able to compute "maximal intervals" efficiently. The following two theorems give us a constant time method to compute "maximal intervals" w.r.t. conjunctive and disjunctive p-strategies.

Theorem 1 *Suppose* $i =< c, d >$ is any conjunctive coherent p-strategy *and* $j =< c', d' >$ is any disjunctive coherent p-strategy. *Then:*

1. $(\forall [a, b] \in \mathcal{C}[0, 1])(md_i[a, b] = [a, 1])$.
2. $(\forall [a, b] \in \mathcal{C}[0, 1])(md_j[a, b] = [0, b])$.

3 Syntax of hp-programs

In hybrid probabilistic programs, we assume the existence of an arbitrary, but fixed set of conjunctive and disjunctive p-strategies. The programmer may augment this set with new strategies when s/he needs new ones for their application. The following definition says that each strategy has an associatated conjunction operator, and a disjunction operator.

Definition 6 *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* $\mathcal{CONJ} \cup \mathcal{DISJ}$.

- Let $i \in \mathcal{CONJ}$. *Connective* $\wedge_i$ *is called an* i-annotated conjunction
- Let $i \in \mathcal{DISJ}$. *Connective* $\vee_i$ *is called an* i-annotated disjunction

Hybrid basic formulas, defined below, are either conjunctions of atoms, or disjunctions of atoms (but not mixes of both) w.r.t. a single connective.

Definition 7 *Let i be a* p-strategy*, $A_1, \ldots, A_k$ be atoms. A hybrid basic formula B is either*

$$A_1 \wedge_i A_2 \ldots \wedge_i A_k$$

or

$$A_1 \vee_i A_2 \ldots \vee_i A_k$$

Let $bf_i(B_L)$ denote the set of all ground hybrid basic formulas for the $\vee_i$ and $\wedge_i$ connectives. Let $bf_{\mathcal{S}}(B_L) = \cup_{i \in \mathcal{S}} bf_i(B_L)$. Similarly, $bf_{\mathcal{CONJ}} = \cup_{i \in \mathcal{CONJ}} bf_i(B_L)$ and $bf_{\mathcal{DISJ}} = \cup_{i \in \mathcal{DISJ}} bf_i(B_L)$.

For instance, returning to our stock example, the formulas (ch-sells-stock(C) $\vee_{ig}$ ch-retires(C)) and (price-drop(C) $\wedge_{in}$ stable(C)) are basic formulas involving the ignorance and independence p-strategies. We may also annotate a basic formula with an interval, as specified below.

Definition 8 *A hybrid probabilistic annotated basic formula (hp-annotated basic formula) is an expression of the form $B : \mu$ where B is a hybrid basic formula and $\mu \in \mathcal{C}[0, 1]$.*

Informally speaking, $B : \mu$ may be read as "The probability of B occuring lies in the interval μ." For example, the annotated basic formula (ch-sells-stock(C) $\vee_{ig}$ ch-retires(C)):$[0.4, 0.9]$ may be read as: "The probability that the chairman sells stock or the chairman retires lies in the 40–90% interval, assuming (no knowledge) ignorance of the relationship between these two primitive events." Hybrid rules may now be constructed from hybrid annotated formulas as follows.

Definition 9 *Let $B_0, B_1, \ldots, B_k$ be hybrid basic formulas. Let $\mu_0, \mu_1, \ldots, \mu_k \in \mathcal{C}[0, 1]$. A hybrid probabilistic clause (hp-clause) is a construction of the form:*

$$B_0 : \mu_0 \leftarrow B_1 : \mu_1 \wedge, \ldots \wedge B_k : \mu_k$$

Informally speaking, the above rule is read: "If the probability of B_1 falls in the interval μ_1 and $\cdots$ the probability of B_k falls within the interval μ_k, then the probability of B_0 falls within the interval μ_0.

Definition 10 *A hybrid probabilistic program (hp-program) over set $\mathcal{S}$ of p-strategies is a finite set of hp-clauses involving only connectives from $\mathcal{S}$.*

For example, the following four clauses constitute a simple hp-program using the p-strategies of ignorance and independence.

price-drop(C):$[0.4, 0.9]$ $\longleftarrow$ (ch-sells-stock(C) $\vee_{ig}$ ch-retires(C)):$[0.4, 0.9]$.
price-drop(C):$[0.3, 0.8]$ $\longleftarrow$ (strike(C) $\vee_{in}$ accident(C)):$[0.3, 1]$.
buy-stock(C):$[0.5, 0.9]$ $\longleftarrow$ (price-drop(C) $\wedge_{in}$ stable(C)):$[0.3, 1]$.
sell-stock(C):$[0.5, 1]$ $\longleftarrow$ (price-drop(C) $\wedge_{in}$ nonstable(C)):$[0.4, 1]$ $\wedge$ have-stock(C):$[1, 1]$.

4 Semantics of hp-programs

Having completed the definition of the syntax of hp-programs, we are now in a position to develop the semantics of such programs.

4.1 Fixpoint Semantics

As usual, suppose we have a logical language L consisting of variable symbols, constant symbols, function symbols, and predicate symbols, and that B_L denotes the Herbrand base of this language. An atomic function, defined below, merely assigns closed intervals to ground atoms.

Definition 11 *A function* $f : B_L \longrightarrow \mathcal{C}[0,1]$ *is called an* atomic function *.*

Intuitively, if $f(A) = \mu$, then this means that according to the atomic function f, the probability of the ground atom A lies in the interval μ. We may impose on ordering, $\leq$, on the set of atomic functions as follows:

$$f_1 \leq f_2 \textbf{ iff } (\forall A \in (B_L)) f_1(A) \supseteq f_2(A).$$

Intuitively, as one "climbs up" the $\leq$-ordering, the atomic functions involved assign "sharper" probability ranges to ground atoms. Though atomic functions do not, by themselves, make assignments to basic formulas, they may be extended to do so,

Definition 12 Let f be *an atomic function.* Then, the function
$h_f : bf_{\mathcal{S}}(B_L) \longrightarrow \mathcal{C}[0,1]$, defined as follows:

1. $h_f(F) = f(F)$, if $F \in B_L$
2. $h_f(F_1 \wedge_i F_2) = c_i(h_f(F_1), h_f(F_2))$ where $i =< c_i, d_i > \in \mathcal{CONJ}$
3. $h_f(F_1 \vee_i F_2) = c_i(h_f(F_1), h_f(F_2))$ where $i =< c_i, d_i > \in \mathcal{DISJ}$

is called a hybrid formula function (based on the atomic function f).

Suppose $\mathcal{HFF}$ denotes the set of all hybrid formula functions. The $\leq$-ordering on atomic functions may be extended to basic formulas in the obvious way: $h_1 \leq h_2$ iff $(\forall F \in bf_{\mathcal{S}}(B_L)) h_1(F) \supseteq h_2(F)$.

Lemma 1 *1. If* f, g are atomic functions such that $f \leq g$*, then* $h_f \leq h_g$*.*

2. $< \mathcal{HFF}, \leq >$ *is a complete lattice.*

Given any hp-program P, we wish to associate with P, an operator T_P that maps hybrid formula functions to hybrid formula functions. We do this by first defining a (similar) intermediate operator S_P that is used subsequently to define T_P.

Definition 13 *Let* P *be* a hybrid probabilistic program*. Operator* $S_P : \mathcal{HFF} \longrightarrow \mathcal{HFF}$ *is defined as follows (where* F *is a basic formula):*

$$S_P(h)(F) = \cap M \textit{ where}$$

$$M = \{\mu | F : \mu \longleftarrow F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n$$

$$\text{is a ground instance of some clause in } P \textit{ and} (\forall j \leq n) h(F_j) \subseteq \mu_j \}$$

if $M = \emptyset$ $S_P(h)(F) = [0,1]$*.*

The operator S_P is very simple. Given $h \in \mathcal{HFF}$ and a basic formula F, it proceeds as follows: (i) First, it finds all ground instances of rules in P such that the head of the rule instance is of the form $F : \mu$ and such that for each $F_i : \mu_i$ in the body, $h(F_i) \subseteq \mu_i$, i.e. h says that F_i's probability does in fact lie within the interval μ_i. (ii) It then takes the intersection of the intervals associated with the heads of all rules identified in the preceding step. Note that in the above definition, it is entirely possible that $S_P(h)(F)$ could be the empty set. In this case, there is an intuitive inconsistency, because the formula function $S_P(h)$ is saying that F's probability lies in the empty set. However, this is absurd, as the empty set cannot contain anything. This will be discussed in further detail in Section 4.2.

However, this operator is not quite "right." The reason is that in order to determine F's probability, it is not enough to merely look for rule instances whose head is identical to F. For instance, F might be $(p \wedge_{ig} q)$. The probability of $(p \wedge_{ig} q)$ may certainly be influenced by rules with head $p : \mu'$ because such rules may impose lower bounds on p's probability – and hence on $(p \wedge_{ig} q)$'s probability. Thus, S_P, by itself, does not allow us to accurately infer the probability associated with a formula F. S_P needs to be *augmented appropriately* in order to do so. However, before defining T_P, we present a simple monotonicity property of S_P. *Note that S_P is monotonic <u>regardless</u> of what p-strategies appear in P.*

Lemma 2 S_P is monotonic, i.e., if h_1, h_2 are two formula functions and $h_1 \leq h_2$ then, $S_P(h_1) \leq S_P(h_2)$.

Let us now define the T_P operator.

Definition 14 *Let P be* a hybrid probabilistic program. We define operator $T_P : \mathcal{HFF} \longrightarrow \mathcal{HFF}$ as follows:

1. *Let F be an <u>atomic</u> formula.*
   - *if $S_P(h)(F) = \emptyset$ then $T_P(h)(F) = \emptyset$.*
   - *if $S_P(h)(F) \neq \emptyset$, then let*

   $$M = \{\mu | (F *_i G) : \mu \longleftarrow F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n$$
   $$\text{where } * \in \{\vee, \wedge\} \textit{ and } i \in \mathcal{S} \textit{ and } (\forall j \leq n) h(F_j) \subseteq \mu_j\}$$

   . We define
   $$T_P(h)(F) = (\cap\{md(\mu) | \mu \in M\}) \cap S_P(h)(F)$$
2. *(F <u>not atomic</u>) Let $F = F_1 \wedge_i F_2$ or $F = F_1 \vee_i F_2$. Then:*
   $$T_P(h)(F) = S_P(h)(F) \cap c_i(T_P(h)(F_1), T_P(h)(F_2)).$$

The T_P operator may be justified as follows: (i) Consider an atomic formula A: if $S_P(h)(F) = \emptyset$, then this means that an inconsistency (to be made more formal in Section 4.2) has occurred. For instance, if we have an hp-program containing two facts $a : [0, 0]$ and $a : [1, 1]$, then whatever h we pick, $S_P(h)(a) = \emptyset$, reflecting the (in this case flagrant) inconsistency in P. Thus, $T_P(h)$ must also assign $\emptyset$ to F. If $S_P(h)(F) \neq \emptyset$, then it may be case that $S_P(h)$ has assigned too "wide" an interval to F, because it ignores rules that are "associated" with F. As F is atomic, there might be rules whose bodies are satisfied by h. that include F in its head. We must find all such rules, and "split" the rule head into its F part, and the non-F part, say G. Clearly, the rule head must be of the form $(F *_i G)$ where $*$ is either $\wedge$ or $\vee$. As the rule's body is satisfied by h, it means that the head of this rule, viz. $(F *_i G)$ has probability in the interval μ. The rule in question thus allows us to conclude that F's probability ranges anywhere in $md(\mu)$ which is the "maximal interval" associated with F w.r.t. the connection $*_i$. We repeat this for each rule with F as part of the head.

(ii) When F is not a ground atom, we repeat this by (inductively) splitting F and computing the T_P operator for the atomic constituents of F, and merging them back, using the composition operator associated with the connective $*_i$.

Theorem 2 T_P is monotonic,i.e., if $h_1, h_2$are two formula functions and $h_1 \leq h_2$ then, $T_P(h_1) \leq T_P(h_2)$.

Again, note that the above result applies <u>*regardless*</u> *of what set of p-strategies occur in program P.* It is easy to see now that we may define the *iterations* of T_P as: (i) $T_P^0 = h_\perp$ where $\perp$ is the atomic function that assigns $[0,1]$ to all ground atoms A. (ii) $T_P^\alpha = T_P(T_P^{\alpha-1})$ where α is a successor ordinal whose predecessor is denoted by $\alpha - 1$. (iii) $T_P^\gamma = \sqcup\{T_P^\alpha | \alpha < \gamma\}$, where γ is limit ordinal. The following result is straightforward, but specified below for the sake of completeness.

Proposition 2 *$T_P^\omega = lfp(T_P)$ where lfp denotes the least fixpoint operator.*

4.2 Probabilistic Model Theory

We are now ready to define a logical model theory for hp-programs. For this purpose, hybrid basic formula functions will play the role of an "interpretation." The key definition of satisfaction is given below.

Definition 15 Satisfaction. Let h be hybrid basic formula function, $F \in bf_{\mathcal{S}}(B_L)$, $\mu \in \mathcal{C}[0,1]$. We say that

- $h \models F : \mu$ **iff** $h(F) \subseteq \mu$.
- $h \models F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n$ **iff** $(\forall\, 1 \leq j \leq n) h \models F_j : \mu_j$.
- $h \models F : \mu \longleftarrow F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n$ **iff** either $h \models F : \mu$ or $h \not\models F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n$.
- $h \models (\exists x)(F : \mu)$ **iff** $h \models F(t/x) : \mu$ *for some ground term t.*
- $h \models (\forall x)(F : \mu)$ **iff** $h \models F(t/x) : \mu$ *for every ground term t.*

A formula function h is called a model *of an hp-program P ($h \models P$)* **iff** $(\forall p \in P)(h \models p)$.

As usual, we say that $F : \mu$ is a *consequence* of P iff for every model h of P, it is the case that $h(F) \subseteq \mu$.

Recall, from Section 4.1, that we can have cases where a hybrid formula function, h, could assign $\emptyset$ to some formula. When $h(F) = \emptyset$, h is "saying" that F's probability lies in the empty set. This corresponds to an inconsistency because, by definition, nothing is in the empty set.

Definition 16 *Formula function h is called* fully defined *iff*
$\forall(F \in bf_{\mathcal{S}}(B_L))(h(F) \neq \emptyset)$.

The following important result fully ties together, the fixpoint theory associated with hp-programs, and the model theoretical characterization of hp-programs, <u>regardless</u> of which p-strategies occur in the hp-program being considered.

Theorem 3 *Let P be any hp-program. Then:*

1. *h is a model of P iff $T_P(h) \leq h$.*
2. *P has a model iff T_P^{ω} is fully defined.*
3. *If T_P^{ω} is fully defined, then it is the least model of P, and $F : \mu$ is a logical consequence of P iff $T_P^{\omega}(F) \subseteq \mu$.*

The second result above links consistency of P programs with the fully defined-ness property of T_P^{ω}. Of course, if there exists an integer i such that either S_P^i or T_P^i are not fully defined, then T_P^{ω} cannot be fully defined either, and hence, P would not have a model.

4.3 Proof Procedure

At this stage, we have provided a complete description of what is a logical consequence of an hp-program P. In this section, we present a query processing procedure. Unlike classical resolution, when dealing with annotated conjunctions and disjunctions, unifiers may not be unique. We summarize this briefly below.

Definition 17
- Θ is a *unifier* of annotated conjunctions $C_1 \equiv A_1 \wedge_i \ldots \wedge_i A_{n_1}$ *and* $C_2 \equiv B_1 \wedge_j \ldots \wedge_j A_{n_2}$ **iff** $i \in \mathcal{CONJ}, j \in \mathcal{CONJ}$ and $i = j$ and $\{A_k\Theta | 1 \leq k \leq n_1\} = \{B_k\Theta | 1 \leq k \leq n_2\}$
- Θ is a *unifier* of annotated disjunctions $D_1 \equiv A_1 \vee_i \ldots \vee_i A_{n_1}$ and $D_2 \equiv B_1 \vee_j \ldots \vee_j A_{n_2}$ **iff** $i \in \mathcal{DISJ}, j \in \mathcal{DISJ}$ and $i = j$ and $\{A_k\Theta | 1 \leq k \leq n_1\} = \{B_k\Theta | 1 \leq k \leq n_2\}$

In order to proceed we need to define a notion of maximally general unifier.

Definition 18 Let $U(C_1, C_2)$ denote the set of all unifiers of C_1 and C_2. Let $\Theta_1, \Theta_2 \in U(C_1, C_2)$.

1. $\Theta_1 \leq \Theta_2$ **iff** there exists a substitution γ, such that $\Theta_1 = \Theta_2\gamma$.
2. $\Theta_1 \equiv \Theta_2$ **iff** $\Theta_1 \leq \Theta_2$ and $\Theta_2 \leq \Theta_1$.
3. Let $[\Theta] = \{\Theta' \in U(C_1, C_2) | \Theta \equiv \Theta'\}$.
4. $[\Theta_1] \leq [\Theta_2]$ **iff** there exists such γ that $[\Theta_1] = [\Theta_2\gamma]$.
5. $[\Theta_1] < [\Theta_2]$ **iff** $[\Theta_1] \leq [\Theta_2]$ and $[\Theta_2] \neq [\Theta_1]$.

From the above definition, it is easy to see that $\equiv$ is an equivalence relation on elements of $U(C_1, C_2)$ and $\leq$ is a partial order on $\{[\Theta] | \Theta \in U(C_1, C_2)\}$. We can define a notion of maximally general unifier.

Definition 19 $\Theta \in U(C_1, C_2)$ is a maximally general unifier (max-gu) of C_1 and C_2 **iff** there is **no** such other unifier $\Theta' \in U(C_1, C_2)$ that $[\Theta] \leq [\Theta']$.

Lemma 3 *If two basic formulas are unifiable then they have a max-gu (not necessarily unique).*

In general, in the presence of basic formulas, just "straight" resolution is not enough. The reason is that to establish a basic formula, e.g. $(p \wedge_i q) : \mu$, we might need to *separately* prove $p : \mu_1$ and $q : \mu_2$ and then combine μ_1, μ_2 using the composition function associated with p-strategy i. There are two ways to do this: (i) allow resolution not against hp-clauses in P, but against hp-clauses in an *expanded* version of P, or (ii) introduce, in addition to resolution, a new rule of inference corresponding the the "expansion" steps alluded above. Both cases are essentially equivalent from the point of view of completeness. Due to space restrictions, we present only the former. The reader can easily see how the latter than be defined as well.

First, we add to P all "tautologies". Any formula of the form $F : [0,1]$ is a tautology as F's probability must, certainly, lie in the $[0,1]$ interval.

Definition 20 Let P be an hp-program. Then REDUN(P) is defined as

$$REDUN(P) = P \cup \{A : [0,1] \longleftarrow |A \in B_L\}.$$

In addition to the above tautologies, we need to "merge" rules together and/or infer "implied" rules. For example, if one rule has $F_1 : \mu_1$ in the head, and another has $F_2 : \mu_2$ in the head, and these are unifiable via max-gu Θ, then these two rules may jointly provide some information on the probability of $(F_1 \wedge_i F_2)$ where i is some p-strategy. Likewise, if $(F_1 *_i F_2) : \mu'$ is in the head of some rule, then this rule certainly provides some information about F_1's probability, and F_2's probability. The closure of P, defined below, expands the rules in P by performing such merges and/or inferences.

Definition 21 Let P be an hp-program. Then $\mathrm{CL}(P)$ (closure of P) is defined as follows

- $\mathrm{CL}^0(P) = REDUN(P)$.
-
  - For each pair of clauses $F_1 : \mu_1 \longleftarrow Body_1$ and $F_2 : \mu_1 \longleftarrow Body_2 \in \mathrm{CL}^i(P)$, such that their heads $F_1 : \mu_1$ and $F_2 : \mu_2$ are unifiable via max-gu Θ add clause $F_1\Theta : \mu_1 \cap \mu_2 \longleftarrow Body_1 \wedge Body_2$ to $\mathrm{CL}^{i+1}(P)$.
  - For each clause $F_1 *_i F_2 : \mu \longleftarrow Body \in \mathrm{CL}^i(P)$ add the following two clauses to $\mathrm{CL}^{i+1}(P)$:
    1. $F_1 : md_i(\mu) \longleftarrow Body$
    2. $F_2 : md_i(\mu) \longleftarrow Body$
  - For each two clauses $F_1 : \mu_1 \longleftarrow Body_1$ and $F_2\mu_1 \longleftarrow Body_2 \in \mathrm{CL}^i(P)$ add the clauses

    $$F_1 \wedge_i F_2 : c_i(\mu_1, \mu_2) \longleftarrow Body_1 \wedge Body_2$$

    for each $i \in \mathcal{CONJ}$ and

    $$F_1 \vee_i F_2 : c_i(\mu_1, \mu_2) \longleftarrow Body_1 \wedge Body_2$$

    for each $i \in \mathcal{DISJ}$ to $\mathrm{CL}^{i+1}(P)$.
- $\mathrm{CL}(P) = \cup \mathrm{CL}^i(P), \quad i = 0, 1 \ldots$

The following result says that the above steps are all sound. No new rule is produced that was not already a logical consequence of P.

Lemma 4 *For every clause $C \in CL(P)$, $P \models C$.*

We now present a refutation procedure for query processing.

Definition 22 A query is a formula of the form $\exists(F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n)$, where $(\forall 1 \leq i \leq n)$ $(F_i \in bf_{\mathcal{S}}(B_L))$. F_is need not be ground.

Definition 23 Suppose $C \equiv G : \lambda \longleftarrow G_1 : \lambda_1 \wedge \ldots \wedge G_m : \lambda_m \in \mathrm{CL}(P)$ and $Q \equiv \exists(F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n)$ is a query. Let C and Q be standardized apart. Let also G and F_i be unifiable for some $1 \leq i \leq n$. Then

$$\exists((F_1 : \mu_1 \wedge \ldots \wedge F_{i-1} : \mu_{i-1} \wedge G_1 : \lambda_1 \wedge \ldots \wedge G_m : \lambda_m \wedge F_{i+1} : \mu_{i+1} \wedge \ldots \wedge F_n : \mu_n)\Theta)$$

is an hp-resolvent of C and Q **iff**:

1. Θ is a max-gu of G and F_i
2. $\lambda \subseteq \mu_i$

If Θ is a unifier but not necessarily a max-gu, we call the resolvent an unrestricted hp-resolvent.

Definition 24 Let $Q \equiv \exists(F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n)$ be an initial query, and P an hp-program. An hp-deduction of Q from P is a sequence $< Q_1, C_1, \Theta_1 > \ldots < Q_r, C_r, \Theta_r > \ldots$ where, $Q = Q_1$, for all $i \geq 1$ C_i is a renamed version of a clause in $\mathrm{CL}(P)$ and Q_{i+1} is an *hp-resolvent* of Q_i and C_i via max-gu Θ_i.
If Θ_i's are not restricted to max-gu, we call the resulting sequence an unrestricted hp-deduction.

Definition 25 Let $Q \equiv \exists(F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n)$ be an initial query, and P an hp-program. An hp-refutation of Q from P is a *finite hp-deduction* $< Q_1, C_1, \Theta_1 > \ldots < Q_r, C_r, \Theta_r >$ where, the hp-resolvent of Q_r and C_r via Θ_r is the empty query. $\Theta_1 \ldots \Theta_r$ is called the *computed answer substitution* .

We are now in a position to state the soundness and completeness of hp-resolution.

Theorem 4 (Soundness of hp-refutation).
Let P be an hp-program, and Q be an initial query. If there exists an hp-refutation of $Q \equiv \exists(F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n)$ from P with the answer substitution Θ then $P \models \forall((F_1 : \mu_1 \wedge \ldots \wedge F_n : \mu_n)\Theta)$.

Theorem 5 (Completeness of hp-refutation).
Let P be a consistent hp-program and Q be a query. Then, if $P \models Q$ then there exists an hp-refutation of Q from P.

5 Conclusions

Logic knowledge bases have been extended to handle fuzzy modes of uncertainty since the early 70's with the advent of the MYCIN and Prospector systems [4]. Shapiro was one of the first to develop results in fuzzy logic programming [18]. Baldwin [1] was one of the first to introduce *evidential* logic programming and a language called FRIL. Van Emden [21] was the first to provide formal semantical foundations for logic programs that was later extended by Subrahmanian [19] and then completely generalized in a succession of papers by Blair and Subrahmanian[2], and Fitting[6], Ginsberg [7], and applied to databases by Kifer and Li [9] and Kifer and Subrahmanian [10]. *All the above works did not obey the laws of probability.*

The first works in the area of probabilistic logic programming were due to Ng and Subrahmanian who, in a series of papers [13, 15], developed techniques for probabilistic logic programming under the assumption of *ignorance*. Their work built upon earlier work on probabilistic logics due to Fagin, Halpern and and others [5] and Nilsson [16].

In contrast, Kiessling and his group [8, 20, 17] have developed a framework called DUCK for reasoning with uncertainty. They provide an elegant logical axiomatic theory for uncertain reasoning in the presence of rules, and using the *independence assumption*.

Perhaps the most significant related work is the elegant recent work of Lakshmanan's group [12, 11]. There are several differences between our approach, and theirs. First, the syntax is different. We associate probabilities with atoms, while [12] associates two intervals with any rule. Second, their work can be viewed as a *refinment* of [14] and [10]. In contrast, in this paper, we propose a generalization of [13] (though for the sake of readability, and due to space restrictions, variable annotations are missing from this paper). Third, our semantics is different. For instance, even if none of the hybrid p-strategies in our paper are present, and we just have two facts $a : [1, 1]$ and $a : [0, 0]$ in P, our semantics would declare an inconsistency, while theirs would imply $a : [1, 1]$, but would not be inconsistent. This seems quite counter-intuitive to us. Last, but not least, they only allow atoms in the head and in the body, at most one conjunction (corresponding to one of our basic formulas) is allowed. In contrast, we allow arbitrary basic formulas in rule heads, and allow arbitrary conjunctions of basic formulas reflecting different probabilistic strategies in the rule body.

References

[1] J.F. Baldwin. (1987) *Evidential Support Logic Programming*, J. of Fuzzy Sets and Systems, 24, pps 1-26.

[2] H. Blair and V.S. Subrahmanian. Paraconsistent Logic Programming, THEORETICAL COMPUTER SCIENCE, Vol. 68, pp 135–154, 1989.

[3] A. Dekhtyar and V.S. Subrahmanian. Hybrid Probabilistic Logic Programs, Technical report, available from:
`http://www.cs.umd.edu/users/vs/papers/probpapers.html.`

[4] R. O. Duda, P. E. Hart and N. J. Nilsson. (1976) *Subjective Bayesian Methods for Rule-based Inference Systems*, Proceedings of National Computer Conference, pp 1075–1082.

[5] R. Fagin and J. Halpern. (1988) *Uncertainty, Belief and Probability*, in Proc. IJCAI-89, Morgan Kaufman.

[6] M. C. Fitting. (1988) *Logic Programming on a Topological Bilattice*, Fundamenta Informatica, 11, pps 209–218.

[7] M. Ginsberg. (1988) *Multivalued Logics: A Uniform Approach to Reasoning in Artificial Intelligence*, Computational Intelligence, vol. 4, 1988, pp. 265–316.

[8] U. Guntzer, W. Kiesling and H. Thone. (1991) *New Directions for Uncertainty Reasoning in Deductive Databases*, Proc. 1991 ACM SIGMOD, pp 178–187.

[9] M. Kifer and A. Li. (1988) *On the Semantics of Rule-Based Expert Systems with Uncertainty*, 2-nd Intl. Conf. on Database Theory, Springer Verlag LNCS 326, (eds. M. Gyssens, J. Paredaens, D. Van Gucht), Bruges, Belgium, pp. 102–117.

[10] M. Kifer and V. S. Subrahmanian. (1992) Theory of Generalized Annotated Logic Programming and its Applications, JOURNAL OF LOGIC PROGRAMMING, 12, 4, pps 335–368, 1992.

[11] V.S. Lakshmanan and F. Sadri. (1994) *Modeling Uncertainty in Deductive Databases*, Proc. Int. Conf. on Database Expert Systems and Applications, (DEXA'94), September 7-9, 1994, Athens, Greece, Lecture Notes in Computer Science, Vol. 856, Springer (1994), pp. 724-733.

[12] V.S. Lakshmanan and F. Sadri. (1994) *Probabilistic Deductive Databases*, Proc. Int. Logic Programming Symp., (ILPS'94), November 1994, Ithaca, NY, MIT Press.

[13] R. Ng and V.S. Subrahmanian. (1993) Probabilistic Logic Programming, INFORMATION AND COMPUTATION, 101, 2, pps 150–201, 1993.

[14] 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.

[15] R. Ng and V.S. Subrahmanian. (1995) Stable Semantics for Probabilistic Deductive Databases, INFORMATION AND COMPUTATION, 110, 1, pps 42-83.

[16] N. Nilsson. (1986) *Probabilistic Logic*, AI Journal 28, pp 71–87.

[17] H. Schmidt, W. Kiessling, U. Guntzer and R. Bayer. (1987) *Combining Deduction by Uncertainty with the Power of Magic*, Proc. DOOD-89, pps 205–224, Kyoto, Japan.

[18] E. Shapiro. (1983) *Logic Programs with Uncertainties: A Tool for Implementing Expert Systems*, Proc. IJCAI '83, pps 529–532, William Kauffman.

[19] V.S. Subrahmanian. (1987) *On the Semantics of Quantitative Logic Programs*, Proc. 4th IEEE Symp. on Logic Programming, pps 173-182, Computer Society Press. Sep. 1987.

[20] H. Thone, W. Kiessling and U. Guntzer. (1995) *On Cautious Probabilistic Inference and Default Detachment*, Annals of Operations Research, 55, pps 195–224.

[21] M.H. van Emden. (1986) *Quantitative Deduction and its Fixpoint Theory*, Journal of Logic Programming, 4, 1, pps 37-53.