Non-monotonic Negation in Hybrid Probabilistic Logic Programs

Emad Saad
Computer Science Department
Abu Dhabi University
emad.saad@adu.ac.ae

Enrico Pontelli
Computer Science Department
New Mexico State University
epontell@cs.nmsu.edu

Abstract—**In [23], a new Hybrid Probabilistic Logic Programs framework has been proposed, and a new semantics has been developed to enable encoding and reasoning about real-world applications. In this paper, the language of Hybrid Probabilistic Logic Programs framework of [23] is extended to allow non-monotonic negation, and two alternative semantics are defined: stable probabilistic model semantics and probabilistic well-founded semantics. Stable probabilistic model semantics and probabilistic well-founded semantics generalize stable model semantics and well-founded semantics of traditional normal logic programs, and they reduce to the semantics of original Hybrid Probabilistic Logic programs framework of [23] for programs without negation. It is the first time that two different semantics for Hybrid Probabilistic Programs with non-monotonic negation as well as their relationships are described. This development provides a foundational ground for developing computational methods for computing the proposed semantics. Furthermore, it makes it clearer how to characterize non-monotonic negation in probabilistic logic programming frameworks for commonsense reasoning.**

I. Introduction

Hybrid Probabilistic Programs (HPP) [5] is a probabilistic logic programming framework that enables the user to *explicitly* encode his/her knowledge about the type of dependencies existing between the probabilistic events being described by the programs. HPP generalizes the *probabilistic annotated logic programming framework*, originally proposed in [18] and further extended in [19]. Since the aim of probabilistic logic programming in general, and of the HPP framework in particular, is to allow reasoning and decision making under probabilistic and statistical knowledge, a generalization and a new semantics for HPP have been defined in [23]. The idea in [23] comes upon observing that commonsense reasoning about probabilities relies on how *likely* (probable) are the various events to occur, rather than how precise our knowledge about these probabilities is. The generalization includes adding the ability to encode the user's knowledge about how to combine the probabilities of the same event derived from different rules in HPP. In addition, the new semantics, intuitively, captures the probabilistic reasoning according to how likely are the various events to occur, by employing the truth order instead of the knowledge order [5]. It was shown that the modified HPP framework is more suitable for reasoning and decision making tasks, including those arising from probabilistic planning. In addition, it was shown that the new HPP framework subsumes Lakshmanan and Sadri's [11] probabilistic implication-based framework as well as it is a natural extension of classical logic programming.

It is known that non-monotonic negation is vital to capture the principles of commonsense reasoning [1]. Moreover, it is important to provide the ability to derive negative conclusions in the absence of positive information [20]. Therefore, it is essential to extend the different probabilistic logic programming frameworks to deal with non-monotonic negation. In this view, the probabilistic logic programming framework in [19] was extended in [20] to allow this important feature by developing a semantics based on a notion of stable models [7]. However, the stable model semantics extension in [20] is computationally expensive [15], since at every fixpoint iteration an exponential number of linear programs, each having an exponential number of variables, needs to be solved [15]. The main reason for this computational complexity arises from the fact that [20] allows annotated conjunctions and disjunctions to appear as heads of the rules. Also, it is worth noting that knowledge order was used in defining the stable model semantics in [20] as well as a fixed assumption (ignorance) is postulated among the dependencies of the various events encoded by the logic programs.

In [13], [14], a well-founded like semantics [8] extension to the probabilistic logic programming framework of [11] was introduced along with well-founded like semantics [8] extension to various non-probabilistic logic programming frameworks with uncertainty [12]. Although the notion of non-monotonic negation in [20] is more natural and closer to the classical notion of non-monotonic negation, the notion of non-monotonic negation in [13], [14] is closer to classical negation (since $Prob(\neg A) = 1 - Prob(A)$). An alternating fixpoint semantics [8] was introduced in [13] to describe the well-founded like semantics. It must be noted that no declarative account was given for the well-founded semantics [9] of the probabilistic logic programming in [13], [14], due to the way non-monotonic negation is interpreted, which has an operational nature. This interpretation of non-monotonic negation makes it less natural to define stable model like semantics [7] and well-founded like semantics [9]. In [14], a framework to approximate the well-founded semantics for [12], including the probabilistic logic programming framework of [11], was described. The approximate well-founded semantics is based on the idea that uncertainty values are assigned as approximations to atoms of the Herbrand base, in the form of intervals, where the certainty values of the atoms lie within these intervals.

However, it is not clear how the approximate well-founded semantics extension [14] works for the probabilistic logic programming framework of [11], where probabilities are originally represented as intervals. To this end, it seems that it is not feasible to define a natural notion of non-monotonic negation as well as stable model semantics [7] extension for the probabilistic logic programming framework in [11].

In this paper, we extend the language of Hybrid Probabilistic Logic Programs of [23], which allows multiple modes of probabilistic combinations and employs the truth order, to support non-monotonic negation, by considering only annotated atoms as heads of rules. This is to avoid the computational complexity inherited from allowing annotated conjunctions or disjunctions to appear as heads of rules. In [23], we have shown that it is possible to develop an algorithm to compute the least fixpoint of HPP without negation and with only annotated atoms as heads of rules in worst-case time complexity $O(n^2)$, where n is the size of a program. In addition, we define two alternative semantics for the extended language; the stable probabilistic model semantics and the probabilistic well-founded semantics and study their relationships. We show that the stable probabilistic model semantics and the probabilistic well-founded semantics generalize the stable model semantics [7] and the well-founded semantics [9] for normal logic programs, and they reduce to the semantics of HPP [23] in the absence of non-monotonic negation. An important result is that the relationship between the stable probabilistic model semantics and the probabilistic well-founded semantics *preserves* the relationship between the stable model semantics and the well-founded semantics for normal logic programs [9].

Another reason why these proposed semantics are interesting is that they provide a foundational ground for building algorithms and systems for computing the meaning of HPP with non-monotonic negation based on the stable probabilistic model semantics and the probabilistic well-founded semantics. The fact that these proposed semantics naturally generalize their classical counterparts suggests that efficient algorithms and implementations can be developed by *extending* the existing efficient algorithms and implementations developed for the stable models and the well-founded semantics for normal logic programs, such as SMODELS [17]. To show this point, an algorithm for computing the least fixpoint for HPP is described in [23], that extends Dowling-Gallier algorithm for computing the satisfiability of a set of Horn formulae [6], which is the ground base for developing the various auxiliary functions in SMODELS (preliminary design of these algorithms has been presented in [21]).

II. Hybrid Probabilistic Programs

In the following subsections, we present the syntax of the proposed Hybrid Probabilistic Programs with non-monotonic negation. We also review the basic syntax, as presented in [5], [23], and the semantics, as described in [23], of Hybrid Probabilistic Programs without negation.

A. Probabilistic Strategies

Let $C[0,1]$ denotes the set of all closed intervals in $[0,1]$. In the context of HPP, probabilities are assigned to primitive events (atoms) and compound events (conjunctions or disjunctions of atoms) as intervals in $C[0,1]$. Let $[a_1,b_1],[a_2,b_2] \in C[0,1]$. Then the *truth order* asserts that $[a_1,b_1] \leq_t [a_2,b_2]$ iff $a_1 \leq a_2$ and $b_1 \leq b_2$. The *knowledge order* states that $[a_1,b_1] \leq_k [a_2,b_2]$ iff $[a_2,b_2] \subseteq [a_1,b_1]$. The set $C[0,1]$ and the relation $\leq_t$ form a complete lattice. In particular, the join ($\oplus_t$) operation is defined as $[a_1,b_1] \oplus_t [a_2,b_2] = [max\{a_1,a_2\}, max\{b_1,b_2\}]$ and the meet ($\otimes_t$) is defined as $[a_1,b_1] \otimes_t [a_2,b_2] = [min\{a_1,a_2\}, min\{b_1,b_2\}]$ w.r.t. $\leq_t$. The type of dependency among the primitive events within a compound event is described by *probabilistic strategies*, which are explicitly selected by the user. We call ρ, a pair of functions $\langle c, md \rangle$, a probabilistic strategy (p-strategy), where $c : C[0,1] \times C[0,1] \rightarrow C[0,1]$, the *probabilistic composition function*, which is *commutative*, *associative*, *monotonic* w.r.t. $\leq_t$, and meets the following *separation* criteria: there are two functions c_1, c_2 such that $c([a_1,b_1],[a_2,b_2]) = [c_1(a_1,a_2), c_2(b_1,b_2)]$. Whereas, $md : C[0,1] \rightarrow C[0,1]$ is the *maximal interval function.* The maximal interval function md of a certain p-strategy returns an estimate of the probability range of a primitive event, A, from the probability range of a compound event that contains A. The composition function c returns the probability range of a conjunction (disjunction) of two events given the ranges of its constituents. For convenience, given a multiset of probability intervals $M = \{\!\{[a_1,b_1], \ldots, [a_n,b_n]\}\!\}$, we use cM to denote $c([a_1,b_1], c([a_2,b_2], \ldots, c([a_{n-1},b_{n-1}],[a_n,b_n])) \ldots)$. According to the type of combination among events, p-strategies are classified into *conjunctive* p-strategies and *disjunctive* p-strategies. Conjunctive (disjunctive) p-strategies are employed to compose events belonging to a conjunctive (disjunctive) formula (please see [5], [23] for the formal definitions).

B. Language Syntax

In this subsection, we describe the syntax of Hybrid Probabilistic Programs and define a syntax for Hybrid Probabilistic Programs with non-monotonic negation. Let L be an arbitrary first-order language with finitely many predicate symbols, constants, and infinitely many variables. Function symbols are disallowed. In addition, let $S = S_{conj} \cup S_{disj}$ be an arbitrary set of p-strategies, where S_{conj} (S_{disj}) is the set of all conjunctive (disjunctive) p-strategies in S. The Herbrand base of L is denoted by B_L. An *annotation* denotes a probability interval and it is represented by $[\alpha_1, \alpha_2]$, where α_1, α_2 are called annotation items. An *annotation item* is either a constant in $[0,1]$, a variable (*annotation variable*) ranging over $[0,1]$, or $f(\alpha_1, \ldots, \alpha_n)$ (called *annotation function*) where f is a representation of a computable total function $f : ([0,1])^n \rightarrow [0,1]$ and $\alpha_1, \ldots, \alpha_n$ are annotation items. The building blocks of the language of HPP are *hybrid basic formulae.* Let us consider a collection of atoms $A_1, \ldots, A_n$, a conjunctive p-strategy ρ, and a disjunctive p-strategy ρ'. Then $A_1 \wedge_\rho \ldots \wedge_\rho A_n$ and $A_1 \vee_{\rho'} \ldots \vee_{\rho'} A_n$ are called *hybrid basic formulae*, and $bf_S(B_L)$ is the set of all ground hybrid basic formulae formed using distinct atoms from B_L and p-strategies from S. An annotated hybrid basic formula is an expression of the form $F : \mu$ where F is a hybrid basic formula and μ is an annotation. A *hybrid literal* is an annotated hybrid basic formula $F : \mu$ (positive annotated hybrid basic formula or positive hybrid literal) or the negation of an annotated hybrid basic formula *not* $(F : \mu)$ (negative annotated hybrid basic formula or negative hybrid literal).

Definition 1 (Rules) A normal hybrid probabilistic rule (nh-rule) is an expression of the form

$$A : \mu \leftarrow F_1 : \mu_1, \ldots, F_n : \mu_n, not\,(G_1 : \mu_{n+1}), \ldots, not\,(G_m : \mu_{n+m})$$

where A is an atom, $F_1, \ldots, F_n, G_1, \ldots, G_m$ are hybrid basic formulae, and μ, μ_i $(1 \leq i \leq m+n)$ are annotations.

A hybrid probabilistic rule (h-rule) is a nh-rule where $m = 0$—i.e., there are no negative hybrid literals.

The intuitive meaning of a nh-rule, in Definition 1, is that, if for each $F_i : \mu_i$, the probability interval of F_i is at least μ_i and for each *not* $(G_j : \mu_j)$, it is not *provable* that the probability interval of G_j is at least μ_j, then the probability interval of A is μ.

Definition 2 (Programs) A normal hybrid probabilistic program over S (*nh-program*) is a pair $P = \langle R, \tau \rangle$, where R is a finite set of nh-rules with p-strategies from S, and τ is a mapping $\tau : B_L \rightarrow S_{disj}$. A hybrid probabilistic program (*h-program*) is a nh-program where all the rules are h-rules.

The mapping τ in the above definition associates to each atomic hybrid basic formula A a disjunctive p-strategy that will be employed to combine the probability intervals obtained from different rules having A in their heads. A nh-program is ground if no variables appear in any of its rules. The following is a typical nh-program.

Example 1: Consider an insurance company which determines the premium categories, by calculating the risk factor according to a genetic test for cancer and the family history for this disease. Assume that customers who have a family history of the disease have a probability of developing cancer with at least 92%. The insurance company will assign high premiums to the customers who have family history of the disease and tested positive as long as their risk conditions are unchanged. Risk conditions can be changed by taking specific medications. This situation can be represented by the following nh-rules:

$$risk(X) : [0.9, 1] \leftarrow (test(X) \wedge_{pc} history(X)) : [0.60, 0.75],$$
$$not\ changeRisk(X)[0.8, 1]$$
$$risk(X) : [0, 0.1] \leftarrow (test(X) \wedge_{pc} history(X)) : [0.60, 0.75],$$
$$changeRisk(X) : [0.8, 1]$$
$$changeRisk(X) : [0.9, 1] \leftarrow medicine(X, Med) : [0.65, 1]$$
$$highPremium(X) : [1, 1] \leftarrow risk(X) : [0.9, 1]$$
$$lowPremium(X) : [1, 1] \leftarrow risk(X) : [0, 0.1]$$
$$test(sam) : [0.92, 1] \leftarrow$$
$$history(sam) : [0.95, 1] \leftarrow$$
$$medicine(sam, medication) : [0.98, 1] \leftarrow$$

and the mapping τ assigns ncd to $risk(sam)$ and an arbitrary disjunctive p-strategy [5], [23] to the other hybrid basic formulae. The ncd denotes the disjunctive negative correlation p-strategy, which is defined as: $c_{ncd}([a_1, b_1], [a_2, b_2]) = [\min(1, a_1 + a_2), \min(1, b_1 + b_2)]$. The first nh-rule asserts that the risk factor is at least 90% whenever the cancer genetic test for a customer is positive and that customer has a family history of cancer with probability between 60% and 75%, and it is not provable that his risk conditions have changed with probability at least 80%. Observe that *test* and *history* events are conjoined according to the *positive correlation* p-strategy (denoted by $\wedge_{pc}$) where $c_{pcc}([a_1, b_1], [a_2, b_2]) = [\min(a_1, a_2), \min(b_1, b_2)]$. The second rule says that the risk factor is at most 10% whenever the customer risk conditions are changed, even though the person tested positive and have a family history of the disease with probability between 60% and 75%. The third nh-rule describes the change of the risk conditions of a customer with probability at least 90% if a medication for the disease becomes available with probability at least 65%. The fourth and fifth nh-rules assert that definite high premium and low premium are considered whenever the probability of risk factors are at least 90% and at most 10% respectively. The last three nh-rules represent the facts available about a specific customer named sam.

C. Satisfaction and Models

In this subsection, we review the declarative semantics and the fixpoint semantics of h-programs [23] and we generalize the notions of interpretations, models, and satisfaction to deal with nh-programs. The notion of a probabilistic model (p-model) is based on *hybrid formula functions*.

Definition 3: A hybrid formula function is a mapping $h : bf_S(B_L) \rightarrow C[0, 1]$ that satisfies the following conditions:

- Commutativity: $h(G_1 *_\rho G_2) = h(G_2 *_\rho G_1)$, $* \in \{\wedge, \vee\}$, $\rho \in S$
- Composition: $c_\rho(h(G_1), h(G_2)) \leq_t h(G_1 *_\rho G_2)$, $* \in \{\wedge, \vee\}$, $\rho \in S$
- Decomposition. For any hybrid basic formula F, $\rho \in S$, and $G \in bf_S(B_L)$:
  $md_\rho(h(F *_\rho G)) \leq_t h(F)$.

The notion of truth order can be extended to hybrid formula functions. Given hybrid formula functions h_1 and h_2, we say $(h_1 \leq_t h_2) \Leftrightarrow (\forall F \in bf_S(B_L) : h_1(F) \leq_t h_2(F))$. The set of all hybrid formula functions, HFF, and the truth order $\leq_t$ form a complete lattice. The meet $\otimes_t$ and the join $\oplus_t$ operations are defined respectively as: for all $F \in bf_S(B_L), (h_1 \otimes_t h_2)(F) = h_1(F) \otimes_t h_2(F)$ and $(h_1 \oplus_t h_2)(F) = h_1(F) \oplus_t h_2(F)$.

Definition 4 (Probabilistic Interpretation) A total (partial) probabilistic interpretation of a nh-program P is a total (partial) hybrid formula function.

Before defining the notion of satisfaction for nh-programs, we introduce the following notations. Let h be a probabilistic interpretation, then $dom(h) \subseteq bf_S(B_L)$ denotes the domain of h ($dom(h) \subsetneqq bf_S(B_L)$ if h is a partial probabilistic interpretation). We use $negdom(h)$ to denote the set $\{F \mid F \in dom(h), h(F) = [0, 0]\}$. We also define

$posdom(h) = dom(h) \backslash negdom(h)$.

Definition 5 (Probabilistic Satisfaction) Let $P = \langle R, \tau \rangle$ be a ground nh-program, h be a probabilistic interpretation, and
$r \equiv A : \mu \leftarrow F_1 : \mu_1, \ldots, F_n : \mu_n, not\ (G_1 : \beta_1), \ldots, not\ (G_m : \beta_m) \in R$. Then

- h satisfies $F_i : \mu_i$ (denoted by $h \models F_i : \mu_i$) iff $F_i \in dom(h)$ and $\mu_i \leq_t h(F_i)$.
- h satisfies $not\ (G_j : \beta_j)$ (denoted by $h \models not\ (G_j : \beta_j)$) iff $G_j \in dom(h)$ and $\beta_j \nleq_t h(G_j)$.
- h satisfies $Body \equiv F_1 : \mu_1, \ldots, F_n : \mu_n, not\ (G_1 : \beta_1), \ldots, not\ (G_m : \beta_m)$ (denoted by $h \models Body$) iff $\forall(1 \leq i \leq n), h \models F_i : \mu_i$ and $\forall(1 \leq j \leq m), h \models not\ (G_j : \beta_j)$.
- h satisfies $A : \mu \leftarrow Body$ iff $h \models A : \mu$ or h does not satisfy $Body$.
- h satisfies P iff h satisfies every nh-rule in R and for every atomic formula $A \in dom(h)$, $c_{\tau(A)}\{\!\{\mu | A : \mu \leftarrow Body \in R\ and\ h \models Body\}\!\} \leq_t h(A)$.

Definition 6 (Models) Let P be a nh-program. A total probabilistic model of P (*p-model*) is a total probabilistic interpretation of P that satisfies P. A partial probabilistic model of P is a partial probabilistic interpretation of P that can be extended to a total probabilistic model of P.

Proposition 1: Let P be an *h-program.* $h_P = \otimes_t\{h | h$ is a p-model of P $\}$ is the least p-model of P.
Associated with each h-program P, is an operator, T_P, called the *fixpoint operator*, which maps total probabilistic interpretations to total probabilistic interpretations.

Definition 7: Let $P = \langle R, \tau \rangle$ be a ground h-program and h be a total probabilistic interpretation. The fixpoint operator T_P is a mapping $T_P : HFF \rightarrow HFF$ which is defined as follows:

1. if A is an atom, $T_P(h)(A) = c_{\tau(A)}\ M_A$ where $M_A = \{\!\{\mu | A : \mu \leftarrow Body \in R$ such that $h \models Body\}\!\}$ and $M_A \neq \emptyset$. If $M_A = \emptyset$, then $T_P(h)(A) = [0,0]$
2. $T_P(h)(G_1 \wedge_\rho G_2) = c_\rho(T_P(h)(G_1), T_P(h)(G_2))$ where $(G_1 \wedge_\rho G_2) \in bf_S(B_L)$
3. $T_P(h)(G_1 \vee_{\rho'} G_2) = c_{\rho'}(T_P(h)(G_1), T_P(h)(G_2))$ where $(G_1 \vee_{\rho'} G_2) \in bf_S(B_L)$.

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

III. Probabilistic Well-Founded Semantics

In this section we define the probabilistic well-founded semantics for nh-programs. We start by defining the notion of probabilistic unfounded set and the immediate consequence operator of nh-programs with respect to a given probabilistic interpretation. Then the probabilistic well-founded semantics is defined inductively in terms of these two operators, which are natural extensions of their classical counterparts used in the well-founded semantics for normal logic programs [9].

Definition 8: Let P be a nh-program, H_P be the set of all partial probabilistic interpretations of P, and $h_1, h_2 \in H_P$. We define the following partial order ($\leq_w$) on H_P: $h_1 \leq_w h_2$ iff $posdom(h_1) \subseteq posdom(h_2)$, $negdom(h_1) \subseteq negdom(h_2)$, and $\forall\ F \in dom(h_1), h_1(F) \leq_t h_2(F)$.

Definition 9: Let h_1 and h_2 be two partial probabilistic interpretations. The meet $\otimes_w$ and join $\oplus_w$ operation corresponding to the partial order $\leq_w$ are defined respectively as:

- $(h_1 \otimes_w h_2)(F) = h_1(F) \otimes_t h_2(F)$ for all $F \in ((posdom(h_1) \cap posdom(h_2)) \cup (negdom(h_1) \cap negdom(h_2)))$, otherwise, undefined.
- $(h_1 \oplus_w h_2)(F) = h_1(F) \oplus_t h_2(F)$ for all $F \in ((posdom(h_1) \cap posdom(h_2)) \cup (negdom(h_1) \cap negdom(h_2)))$,
  $(h_1 \oplus_w h_2)(F) = h_1(F)$ for all $F \in ((posdom(h_1) \setminus posdom(h_2)) \cup (negdom(h_1) \setminus negdom(h_2)))$, and
  $(h_1 \oplus_w h_2)(F) = h_2(F)$ for all $F \in ((posdom(h_2) \setminus posdom(h_1)) \cup (negdom(h_2) \setminus negdom(h_1)))$, otherwise, undefined.

Note that, the pair $\langle H_P, \leq_w \rangle$ *does not* form a lattice. In fact, if $h_1, h_2 \in H_P$ are probabilistic interpretations and $h_1 \nleq_w h_2$, then $lub\{h_1, h_2\}$ may not exist. Consider $B_L = \{a, b, c, d\}$, $h_1(a) = h_1(b) = [0,0], h_1(c) = h_1(d) = [1,1]$, and $h_2(a) = [0,0], h_2(b) = h_2(c) = h_2(d) = [1,1]$. Then, according to the definition of $\leq_w$, $lub\{h_1, h_2\}$ must assign $[0,0]$ to a, b and assign $[1,1]$ to b, c and d which does not exist. However, $\langle H_P, \leq_w \rangle$ is *a complete partial order (cpo)*, i.e., a partial order in which the limit of each increasing chain exists. This is sufficient to allow the inductive construction of well-founded probabilistic models. The bottom element in this partial order is the partial probabilistic interpretation Φ whose domain is the empty set, and its top element is the total probabilistic interpretation which assigns $[1,1]$ to each element in $bf_S(B_L)$. We say that a nh-program globally satisfies $F : \nu$ ($not\ (G : \beta)$) if the nh-program as a whole provides evidence for satisfying $F : \nu$ ($not\ (G : \beta)$).

Definition 10 (Global Satisfaction) Let P be a nh-program and $F : \nu$ ($not\ (G : \beta)$) be a positive (negative) hybrid literal. We say that $F : \nu$ ($not\ (G : \beta)$) is *globally satisfied by* P if every minimal probabilistic interpretation that satisfies P also satisfies $F : \nu$ ($not\ (G : \beta)$).

Definition 11 (Probabilistic Unfounded Sets) Let $P = \langle R, \tau \rangle$ be a ground nh-program, $h \in H_P$, and $U \subseteq bf_S(B_L)$ such that for each non-atomic $F = A_1 \vee_{\rho'} \ldots \vee_{\rho'} A_n \in U$, all its constituent atoms belong to U and for each non-atomic $F = A_1 \wedge_\rho \ldots \wedge_\rho A_n \in U$ at least one $A_i \in U$ and the others are defined in h. U is called a *Probabilistic Unfounded Set* of P w.r.t. h if for each atomic $A \in U$ we have that for each nh-rule r in R whose head is $A : \mu$, at least one of the following conditions holds:

- there exist some positive hybrid literal $F : \nu$ in the body of r such that $F \in U$;
- h does not satisfy some hybrid literal $F : \nu$ or $not\ (G : \beta)$ in the body of r and P does not globally satisfy $F : \nu$.

We consider h, in the above definition, to be what we already know about the intended probabilistic model of P. The idea is that the probabilistic unfounded set corresponds to the set of negative conclusions of the nh-program P. Therefore, if a hybrid basic formula F is in a probabilistic unfounded set of P, then F should be assigned the probability interval $[0,0]$ (representing absolute falsity) by the total or partial probabilistic model of P. The condition "P does not globally satisfies $F:\nu$" in the above definition is not required in the case of negative hybrid literals ($not\,(G:\beta)$). The reason is that if $not\,(G:\beta)$ is not satisfied by h, then $not\,(G:\beta)$ is not going to be satisfied by any $h \leq_w h'$. Instead, for positive literals $F:\nu$ we need to enforce the additional condition, used to guarantee that even by accumulating more knowledge, the probability interval assigned to F will not cover ν.

Definition 12 (Greatest Unfounded Set) Let P be a ground nh-program and h be a partial probabilistic interpretation. The greatest probabilistic unfounded set $U_P(h)$ of P w.r.t. h is the union of all probabilistic unfounded sets of P w.r.t. h.

Definition 13 (The Immediate Consequence Operator K_P) Let $P = \langle R, \tau \rangle$ be a ground nh-program and $h \in H_P$. The immediate consequence operator K_P is the mapping $K_P : H_P \rightarrow H_P$ defined as follows:

1. For each atom A we have that $K_P(h)(A) = c_{\tau(A)}\, M_A$, where $M_A \neq \emptyset$ contains the probability intervals μ obtained from the nh-rules $A:\mu \leftarrow Body \in R$, such that h satisfies $Body$, and for each negative hybrid literal $not\,(G_j:\beta_j)$ in $Body$ we have that P globally satisfies $not\,(G_j:\beta_j)$.
2. $K_P(h)(G_1 \wedge_\rho G_2) = c_\rho(K_P(h)(G_1), K_P(h)(G_2))$ where $(G_1 \wedge_\rho G_2)$ contains only atoms from $dom(K_P(h))$).
3. $K_P(h)(G_1 \vee_{\rho'} G_2) = c_{\rho'}(K_P(h)(G_1), K_P(h)(G_2))$ where $(G_1 \vee_{\rho'} G_2)$ contains atoms from $(dom(h) \cup dom(K_P(h)))$ and at least one atom from $dom(K_P(h))$).

Intuitively, $K_P(h)$ corresponds to the set of positive conclusions of P with respect to the probabilistic interpretation h, where for each F defined in $K_P(h)$, $K_P(h)(F) \neq [0,0]$. The condition "P globally satisfies $not\,(G_j:\beta_j)$" in (1) is not restrictive in the case of positive hybrid literals $F_i:\mu_i$. The reason is that if $F_i:\mu_i$ is satisfied by h, then $F_i:\mu_i$ is going to be satisfied by any $h \leq_w h'$. However, this is not the case with the negative hybrid literals $not\,(G_j:\beta_j)$. This is because if $not\,(G_j:\beta_j)$ in the body of a nh-rule is satisfied by h, then it might be not satisfied by some $h \leq_w h'$. Therefore, to guarantee that $not\,(G_j:\beta_j)$ is satisfied by h or by any $h \leq_w h'$, the condition in (1) is imposed. Since for any F defined in $K_P(h)$, $K_P(h)(F) \neq [0,0]$, thanks to the properties of the conjunctive and disjunctive p-strategies (see [5], [23] for more details), the condition "$(G_1 \wedge_\rho G_2)$ contains only atoms from $dom(K_P(h))$" in (2) and the condition "$(G_1 \vee_{\rho'} G_2)$ contains atoms from $(dom(h) \cup dom(K_P(h)))$ and at least one atom from $dom(K_P(h))$" in (3) are imposed to determine $K_P(h)(G_1 \wedge_\rho G_2)$ and $K_P(h)(G_1 \vee_{\rho'} G_2)$ respectively. This is because, for any $[a,b] \neq [0,0]$ and any conjunctive p-strategy ρ, $c_\rho([a,b],[0,0]) = [0,0]$. Then it must be that $K_P(h)(G_1) \neq [0,0]$ and $K_P(h)(G_2) \neq [0,0]$ for $K_P(h)(G_1 \wedge_\rho G_2) \neq [0,0]$. However, for $K_P(h)(G_1 \vee_{\rho'} G_2) \neq [0,0]$, it suffices that $K_P(h)(G_1) \neq [0,0]$ or $K_P(h)(G_2) \neq [0,0]$. Let us proceed with the definition of the probabilistic well-founded operator and the construction of the well-founded probabilistic models.

Definition 14: Let P be a nh-program, h be a partial probabilistic interpretation, $K_P(h)$ be the immediate consequence operator, and $U_P(h)$ be the greatest probabilistic unfounded set of P w.r.t. h. Then, W_P is the mapping $W_P : H_P \rightarrow H_P$ such that

- $W_P(h)(F) = K_P(h)(F)$ for all $F \in dom(K_P(h))$, and
- $W_P(h)(F) = [0,0]$ for all $F \in U_P(h)$.

Lemma 1: [1] The operators W_P and K_P are monotonic w.r.t. $\leq_w$, and U_P is monotonic w.r.t. $\subseteq$.

Definition 15: The partial probabilistic interpretations h_α and h_∞ are defined recursively as follows:

1. $h_0 = \Phi$ where Φ is a partial probabilistic interpretation with an empty domain.
2. $h_\alpha = W_P(h_{\alpha-1})$ where α is the successor ordinal of $\alpha - 1$.
3. $h_\alpha = \oplus_w \{h_\beta \mid \beta < \alpha$ and α is a limit ordinal$\}$.
4. $h_\infty = \oplus_w \{h_\alpha \mid \alpha$ is an ordinal$\}$

Definition 16: Let P be a nh-program. $h_\infty = lfp(W_P)$ is the *well-founded (partial or total) probabilistic model* of P.

Example 2: Let us consider the nh-program $P = \langle R, \tau \rangle$ from Example 1. It can be easily seen that P has a total well-founded probabilistic model h where

$$\begin{array}{lcl}
h(risk(sam)) & = & [0,0.1] \\
h(changeRisk(sam)) & = & [0.9,1] \\
h(highPremium(sam)) & = & [0,0] \\
h(lowPremium(sam)) & = & [1,1] \\
h(test(sam)) & = & [0.92,1] \\
h(history(sam)) & = & [0.95,1] \\
h(medicine(sam, medication)) & = & [0.98,1], \\
h(test(sam) \wedge_{pc} history(sam)) & = & [0.92,1].
\end{array}$$

Example 3: Consider the following nh-program $P = \langle R, \tau \rangle$ where R is

$$\begin{array}{lcl}
a:[0.89,0.91] & \leftarrow & not\,(b:[0.3,0.4]) \\
b:[0.55,0.60] & \leftarrow & not\,(a:[0.7,0.75]) \\
c:[0.2,0.3] & \leftarrow & d:[0.1,0.15] \\
d:[0.1,0.2] & \leftarrow & not\,(e:[0.1,0.3])
\end{array}$$

and $\tau(a) = \tau(b) = \tau(c) = \tau(d) = \pi$ where π is any arbitrary disjunctive p-strategy. This nh-program has a well-founded partial probabilistic model that assigns $[0.2,0.3]$ to c, $[0.1,0.2]$ to d, and $[0,0]$ to e. This is because $h_1 = W_P(\Phi)$ assigns $[0,0]$ to e since $K_P(\Phi) = \Phi$ and $U_P(\Phi) = \{e\}$. $h_2 = W_P(h_1)$ assigns $[0.1,0.2]$ to d and $[0,0]$ to e since $K_P(h_1)$ assigns $[0.1,0.2]$ to d and $U_P(h_1) = \{e\}$. $h_3 = W_P(h_2)$ assigns $[0.1,0.2]$ to d, $[0.2,0.3]$ to c, and $[0,0]$ to e since $K_P(h_2)$ assigns $[0.1,0.2]$ to d and $[0.2,0.3]$ to c, and $U_P(h_2) = \{e\}$. $h_3 = W_P(h_2)$ is the least fixpoint since $h_3 = W_P(h_2) = h_4 = W_P(h_3)$.

Example 4: Consider the following nh-program $P =$

[1] All proofs can be found at
http://www.cs.nmsu.edu/TechReports/2005/006.pdf

$\langle R, \tau \rangle$ where R is

$$\begin{array}{lcl} a : [0.4, 0.7] & \leftarrow & not\,(b : [0.5, 0.75]) \\ b : [0.5, 0.9] & \leftarrow & not\,(a : [0.35, 0.6]) \\ r : [0.25, 60] & \leftarrow & a : [0.4, 0.65] \\ r : [0.3, 0.65] & \leftarrow & b : [0.5, 0.8] \end{array}$$

and $\tau(a) = \tau(b) = \tau(r) = \pi$. The well-founded p-model of P is Φ. This is because $W_P(\Phi) = \Phi$ since $K_P(\Phi) = \Phi$ and $U_P(\Phi) = \emptyset$.

Theorem 1: Every h-program P has a well-founded total probabilistic model h iff h is the least p-model of P.

Let us show that the probabilistic well-founded semantics generalizes the well-founded semantics of normal logic programs. A normal logic program P can be represented as a nh-program $P' = \langle R, \tau \rangle$ where each normal rule

$$a \leftarrow b_1, \ldots, b_n, not\ c_1, \ldots, not\ c_m \in P$$

can be encoded, in R, as a nh-rule of the form

$$a : [1,1] \leftarrow b_1 : [1,1], \ldots, b_n : [1,1], not\ (c_1 : [1,1]), \ldots, not\ (c_m : [1,1])$$

where $a, b_1, \ldots, b_n, c_1, \ldots, c_m$ are atomic hybrid basic formulae and $[1,1]$ represents the truth value *true*. τ is any arbitrary assignment of disjunctive p-strategies. We call the class of nh-programs that consists only of nh-rules of the above form as $NHPP_1$.

Proposition 3: Let P be a normal logic program. Then, I is a well-founded partial or total model for P iff h is a well-founded partial or total probabilistic model for P' where $h(a) = [1,1]$ iff $a \in I$ and $h(b) = [0,0]$ iff $not\, b \in I$.

IV. Stable Probabilistic Model Semantics

In this section we introduce the notion of *stable probabilistic models (sp-models)*, which extends the notion of stable models for classical logic programming [7]. The semantics is defined in two steps. First, we guess a p-model h for a certain nh-program P, then we define the notion of the probabilistic reduct of P with respect to h—which is an h-program. Second, we determine whether h is a stable p-model for P or not by employing the fixpoint operator of the probabilistic reduct to verify whether h is its least p-model. All probabilistic interpretations and models that we consider in this section are total. It must be noted that every h-program has a unique least (total) p-model [23].

Definition 17 (Probabilistic Reduct) Let $P = \langle R, \tau \rangle$ be a ground nh-program and h be a probabilistic interpretation. The probabilistic reduct P^h of P w.r.t. h is $P^h = \langle R^h, \tau \rangle$ where:

$$R^h = \left\{ \begin{array}{l} A : \mu \leftarrow F_1 : \mu_1, \ldots, F_n : \mu_n \mid \\ \qquad A : \mu \leftarrow F_1 : \mu_1, \ldots, F_n : \mu_n, \\ \qquad not\,(G_1 : \beta_1), \ldots, not\,(G_m : \beta_m) \in R\, and \\ \forall (1 \leq j \leq m),\, \beta_j \not\leq_t h(G_j) \end{array} \right\}$$

The probabilistic reduct P^h is an h-program. For any $not\ (G_j : \beta_j)$ in the body of $r \in R$ with $\beta_j \not\leq_t h(G_j)$ is simply satisfied by h, and $not\ (G_j : \beta_j)$ is removed from the body of r. If $\beta_j \leq_t h(G_j)$ then the body of r is not satisfied and r is trivially ignored.

Definition 18 (Stable Probabilistic Model) A probabilistic interpretation h is a stable p-model of a nh-program P if h is the least p-model of P^h.

Example 5: It is easy to verify that the only stable p-model of the program in Example 1 is given by:

$$\begin{array}{lcl} h(risk(sam)) & = & [0, 0.1] \\ h(changeRisk(sam)) & = & [0.9, 1] \\ h(highPremium(sam)) & = & [0, 0] \\ h(lowPremium(sam)) & = & [1, 1] \\ h(test(sam)) & = & [0.92, 1] \\ h(history(sam)) & = & [0.95, 1] \\ h(medicine(sam, medication)) & = & [0.98, 1] \\ h(test(sam) \wedge_{pc} history(sam)) & = & [0.92, 1] \end{array}$$

Example 6: The nh-program in Example 3 has two stable p-models h_1 and h_2 where $h_1(a) = [0.89, 0.91], h_1(b) = [0, 0], h_1(c) = [0.2, 0.3], h_1(d) = [0.1, 0.2], h_1(e) = [0, 0]$ and $h_2(a) = [0, 0], h_2(b) = [0.55, 0.60], h_2(c) = [0.2, 0.3], h_2(d) = [0.1, 0.2], h_2(e) = [0, 0]$. Since, for example, h_1 can be verified as a stable p-model because the probabilistic reduct of P w.r.t. h_1 contains the h-rules:

$$\begin{array}{lcl} a : [0.89, 0.91] & \leftarrow & \\ c : [0.2, 0.3] & \leftarrow & d : [0.1, 0.15] \\ d : [0.1, 0.2] & \leftarrow & \end{array}$$

and $lfp(T_{P^{h_1}}) = h_1$.

Example 7: The nh-program in Example 4 has two stable p-models h_1 and h_2 where $h_1(a) = [0.4, 0.7], h_1(b) = [0,\, 0], h_1(r) = [0.25,\, 0.60]$ and $h_2(a) = [0,\, 0], h_2(b) = [0.5, 0.9]$,
$h_2(r) = [0.3, 0.65]$. Since, for example, h_2 can be verified as a stable p-model because the probabilistic reduct of P w.r.t. h_2 contains the h-rules:

$$\begin{array}{lcl} b : [0.5, 0.9] & \leftarrow & \\ r : [0.25, 0.60] & \leftarrow & a : [0.4, 0.65] \\ r : [0.3, 0.65] & \leftarrow & b : [0.5, 0.8] \end{array}$$

and $lfp(T_{P^{h_2}}) = h_2$.

Theorem 2: Every h-program P has a unique stable p-model h iff h is the least p-model of P.

The following result shows that the stable p-model semantics generalizes the stable model semantics for classical logic programming [7].

Proposition 4: Let P be a normal logic program. Then S' is a stable model of P iff h is a stable p-model of $P' \in NHPP_1$ that corresponds to P where $h(a) = [1,1]$ iff $a \in S'$ and $h(b) = [0,0]$ iff $b \in B_L \setminus S'$.

In the rest of this section we define the *immediate consequence operator* of nh-programs and study its relationship to the stable p-model semantics.

Definition 19: Let $P = \langle R, \tau \rangle$ be a ground nh-program and $h \in HFF$. The *immediate consequence operator* T'_P is a mapping $T'_P : HFF \rightarrow HFF$ defined as follows:

1. $T'_P(h)(A) = c_{\tau(A)}\, M'_A$ where

$$M'_A = \left\{\!\!\left\{ \mu \left| \begin{array}{l} A : \mu \leftarrow F_1 : \mu_1, \ldots, F_n : \mu_n, \\ not\,(G_1 : \beta_1), \ldots, not\,(G_m : \beta_m) \in R\, and \\ \forall (1 \leq i \leq n), \mu_i \leq_t h(F_i)\ and \\ \forall (1 \leq j \leq m), \beta_j \not\leq_t h(G_j) \end{array} \right. \right\}\!\!\right\}$$

and $M'_A \neq \emptyset$. $T'_P(h)(A) = [0,0]$ if $M'_A = \emptyset$.

2. $T'_P(h)(G_1 \wedge_\rho G_2) = c_\rho(T'_P(h)(G_1), T'_P(h)(G_2))$ where $(G_1 \wedge_\rho G_2) \in bf_S(B_L)$.
3. $T'_P(h)(G_1 \vee_{\rho'} G_2) = c_{\rho'}(T'_P(h)(G_1), T'_P(h)(G_2))$ where $(G_1 \vee_{\rho'} G_2) \in bf_S(B_L)$.

It is easy to see that T'_P extends T_P to handle h-rules with negative hybrid literals and, hence, $T'_P = T_P$ for any h-program P. The operator T'_P is not monotonic w.r.t. $\leq_t$. This can be seen in the following example.

Example 8: Consider the nh-program: $a : [0.2, 0.3] \leftarrow$ *not* $(b : [0.6, 0.8])$. Let h_1 be a probabilistic interpretation that assigns $[0, 0]$ to b and $[0, 0]$ to a. In addition, let h_2 be a probabilistic interpretation that assigns $[0.65, 0.9]$ to b and $[0, 0]$ to a. Hence, $h_1 \leq_t h_2$. But $T'_P(h_1)(a) = [0.2, 0.3]$ and $T'_P(h_2)(a) = [0, 0]$. Thus, $T'_P(h_1) \not\leq_t T'_P(h_2)$

Theorem 3: Let P be a nh-program and h be a stable p-model of P. Then h is a minimal fixpoint of T'_P.

It is worth noting that not every minimal fixpoint of T'_P is a stable p-model of P. Consider the following nh-program P.

Example 9: Let $P = \langle R, \tau\rangle$ where τ is arbitrary and R

$$\begin{array}{lcl} a : [0.7, 0.8] & \leftarrow & \textit{not}\,(a : [0.1, 0.17]) \\ a : [0.1, 0.33] & \leftarrow & b : [0.6, 0.8] \\ b : [1, 1] & \leftarrow & a : [0.1, 0.24] \end{array}$$

It is easy to verify that the probabilistic interpretation $h(a) = [0.1, 0.33]$ and $h(b) = [1, 1]$ is a minimal fixpoint of T'_P. However, P^h contains $a : [0.1, 0.33] \leftarrow b : [0.6, 0.8]$ and $b : [1, 1] \leftarrow a : [0.1, 0.24]$ where $lfp(T_{P^h})(a) = lfp(T_{P^h})(b) = [0, 0]$. Hence, h is not a stable p-model for P.

V. Stable P-model Semantics and Probabilistic Well-Founded Semantics Relationships

There is a close relationship between the well-founded probabilistic models and the stable probabilistic models. In this section we study this relationship. For a given nh-program P, the following results show that for every total p-model h of P, h is a stable p-model of P if and only if it is a fixpoint of the probabilistic well-founded operator W_P. However, well-founded total probabilistic models are unique stable probabilistic models.

Theorem 4: Let P be a nh-program and h be a total p-model of P. Then h is stable p-model of P iff h is a fixpoint of W_P.

Corollary 1: Let P be a nh-program and h be a well-founded total p-model of P. Then h is the unique stable p-model of P.

The following result shows that the well-founded probabilistic model approximates the stable p-models of a nh-program, since the well-founded partial p-model of a nh-program P is contained (with respect to the partial order $\leq_w$) in every stable p-model of P.

Corollary 2: Let P be a nh-program and h be a well-founded partial p-model of P. Then for every stable p-model g of P, $h \leq_w g$.

VI. Related Work

A stable model semantics extension to the probabilistic logic programming in [18], [19] was presented in [20]. The notion of non-monotonic negation presented in [20] is closer to our definition of non-monotonic negation. The main difference with respect to [20] is that we employ the truth order instead of the knowledge order as well as our framework allows reasoning with different modes of probabilistic combinations. However, [20] is limited to a single mode of probabilistic combination. Moreover, the stable model semantics in [20] is computationally expensive, due to annotated conjunctions or disjunctions are allowed as heads of rules. On the other hand, we allow only annotated atoms as heads of rules, rather than annotated conjunctions or disjunctions as in [20], to avoid the high computational complexity of the semantics [15]. Another important difference between our framework and [20] is that we do not allow hybrid basic formulae with annotation $[0, 0]$ to appear neither in the heads nor in the bodies of the rules (this is an extension to our framework that we will consider in the future according to the open world assumption), however, [18], [19], [20], [5] does, although these semantics as well as ours are based on the *closed world assumption*. The reason is that having a hybrid basic formula, A, in a nh-program with the annotation $[0, 0]$, i.e., $A : [0, 0]$, means that A is absolutely false. This $A : [0, 0]$ corresponds to classical negation $\neg A$, which in turn requires a different treatment when defining the semantics of the programs.

A probabilistic semantics, based on the possible world semantics, for disjunctive logic programs with non-monotonic negation has been presented in [16]. The semantics of [16] is based on multi-valued logic and a stable model semantics has been described. In addition to programs in [16] are disjunctive logic programs, probabilities are treated as a lattice of truth values, where the probability of the conjunction $Prob(A \wedge B) = min(Prob(A), Prob(B))$ and the probability of the disjunction $Prob(A \vee B) = max(Prob(A), Prob(B))$. This is considered a fixed mode of combination. Whereas, in our framework conjunctions and disjunctions are treated differently according to the type of dependency between events. In [2], a new methodology to probabilistic reasoning was presented under the possible world semantics by employing *answer set programming* for classical logic programming. Answer set programming in [2] is exploited to emulate the possible world semantics. However, [2] assumes independence of probabilities which is a fixed mode of probabilistic combination.

Our probabilistic well-founded semantics introduced in this paper differs from the well-founded semantics presented in [13], [14] in various ways. The notion of non-monotonic negation in [13], [14] is closer to the classical negation. In addition, our probabilistic well-founded semantics is based on the declarative well-founded semantics for normal logic programs [9], however, the well-founded semantics in [13], [14] is based on the alternating fixpoint semantics [8]. A stable model semantics and well-founded semantics (based on alternating fixpoint semantics) have also been presented in [24]. However, the certainty values that are reasoned about are non-probabilistic values. In addition, no annotated conjunctions or disjunctions are allowed in the body of rules [24]. Furthermore, in [4], the semantics of [24] has been extended to allow classical negation as well as non-monotonic negation by proposing alternating fixpoint like semantics. A generalization of HPP of [5] was proposed in [3] by providing a more general semantical characterization in which HPP fits. However, [3] does not allow non-monotonic negation in defining its semantics. In addition, it relies on a complex translation process which is exponential in the size of HPP.

VII. Conclusions and Future Work

We presented an extension of the language of hybrid probabilistic programs framework [23], called normal hybrid probabilistic programs, to cope with non-monotonic negation. The extension is a necessary requirement in many real-world applications (e.g., planning with incomplete and uncertain knowledge). We developed a semantical characterization of the extended framework, which relies on a probabilistic generalization of the well-founded semantics and stable model semantics, originally developed for normal logic programs. We showed that the probabilistic well-founded semantics and the stable probabilistic model semantics naturally generalize the well-founded semantics and the stable model semantics for classical logic programming. Furthermore, we showed that they naturally extend the semantics for HPP (without negation) proposed in [23]. Moreover, we showed that the relationship between the probabilistic well-founded semantics and the stable probabilistic model semantics preserves the relationship between the well-founded semantics and the stable model semantics for normal logic programs.

A topic of future research is to extend the language of normal hybrid probabilistic programs to allow classical negation and disjunctions of annotated atomic formulae in the heads of nh-rules. We plan to develop an alternating fixpoint semantics for the language of nh-programs and analytically study its relationship to the probabilistic well-founded semantics proposed in this paper. In addition, we intend to investigate the computational aspects of the stable probabilistic model semantics and the probabilistic well-founded semantics—by developing algorithms and implementations for computing these semantics. The algorithms and implementations we will develop will be based on appropriate extensions of the existing techniques for computing the stable model semantics and the well-founded semantics for normal logic programs, e.g., SMODELS [17].

References

[1] K.R. Apt and R.N. Bol. Logic programming and negation:a survey. *Journal of logic programming*, 19/20:9-71, 1994.

[2] C. Baral et al. Probabilistic reasoning with answer sets. *In 7th International Conference on Logic Programming and Nonmonotonic Reasoning*, Springer Verlag, 2004.

[3] C.V. Damasio and L. Moniz Pereira. Hybrid probabilistic logic programs as residuated logic programs. *JELIA*, 2000.

[4] C.V. Damasio et al. Coherent well-founded annotated logic programs. *LPNMR*, Springer, 1999.

[5] A. Dekhtyar and V.S. Subrahmanian. Hybrid probabilistic program. *Journal of Logic Programming*, 43(3): 187-250, 2000.

[6] W.F. Dowling and J.H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 1(3): 267-284, 1984.

[7] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. *ICSLP*, 1988, MIT Press.

[8] A. Van Gelder. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences*, 47(1):185-221, 1993.

[9] A. Van Gelder, K.A. Ross, and J.S. Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, 38(3):620-650, 1991.

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

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

[12] L.V.S. Lakshmanan and N. Shiri. A parametric approach to deductive databases with uncertainty. *IEEE TKDE*, 13(4):554–570, 2001.

[13] Y. Loyer and U. Straccia. The well-founded semantics in normal logic programs with uncertainty. *FLOPS*, 2002, Springer Verlag.

[14] Y. Loyer and U. Straccia. The approximate well-founded semantics for logic programs with uncertainty. *In 28th International Symposium on Mathematical Foundations of Computer Science*, 2003.

[15] T. Lukasiewicz. Probabilistic logic programming. *In 13th European Conference on Artificial Intelligence*, 388–392, 1998.

[16] T. Lukasiewicz. Many-valued disjunctive logic programs with probabilistic semantics. *LPNMR*, 1999.

[17] I. Niemela and P. Simons. Efficient implementation of the well-founded and stable model semantics. *In Joint International Conference and Symposium on Logic Programming*, 289-303, 1996.

[18] R.T. Ng and V.S. Subrahmanian. Probabilistic logic programming. *Information & Computation*, 101(2), 1992.

[19] R.T. Ng and V.S. Subrahmanian. A semantical framework for supporting subjective and conditional probabilities in deductive databases. *ARJ*, 10(2), 1993.

[20] R.T. Ng and V.S. Subrahmanian. Stable semantics for probabilistic deductive databases. *Information & Computation*, 110(1), 1994.

[21] E. Saad. *Hybrid probabilistic programs with non-monotonic negation: semantics and algorithms.* Ph.D. thesis, New Mexico State University, May 2005.

[22] E. Saad and E. Pontelli. Towards a more practical hybrid probabilistic logic programming framework. In *Practical Aspects of Declarative Languages.* Springer Verlag, 2005.

[23] E. Saad and E. Pontelli. Hybrid probabilistic logic programming with non-monotoic negation. In *Twenty First International Conference on Logic Programming.* Springer Verlag, 2005.

[24] V.S. Subrahmanian. Amalgamating knowledge bases. *ACM TDS*, 19(2):291–331, 1994.