Qualitative and Quantitative Reasoning in Hybrid Probabilistic Logic Programs

Emad Saad
College of Computer Science and Information Technology
Abu Dhabi University
Abu Dhabi, UAE
emad.saad@adu.ac.ae

Abstract

Reasoning with qualitative and quantitative uncertainty is required in some real-world applications [6]. However, current extensions to logic programming with uncertainty support representing and reasoning with either qualitative or quantitative uncertainty. In this paper we extend the language of Hybrid Probabilistic Logic programs [28, 25], originally introduced for reasoning with quantitative uncertainty, to support both qualitative and quantitative uncertainty. We propose to combine disjunctive logic programs [10, 17] with Extended and Normal Hybrid Probabilistic Logic Programs (EHPP [25] and NHPP [28]) in a unified logic programming framework, to allow directly and intuitively to represent and reason in the presence of both qualitative and quantitative uncertainty. The semantics of the proposed languages are based on the answer set semantics and stable model semantics of extended and normal disjunctive logic programs [10, 17]. In addition, they also rely on the probabilistic answer set semantics and the stable probabilistic model semantics of EHPP [25] and NHPP [28].

Keywords. Probabilistic reasoning, probabilistic logic programming, knowledge representation.

1 Introduction

Reasoning under uncertainty is crucial in most real-world applications such as planning with uncertain domains and reasoning about actions with uncertain effects—such as the actions that arise from robotics in real-world environments. The literature is rich with different forms of uncertainty in logic programming. These forms of uncertainty can be classified into qualitative and quantitative models of uncertainty. Qualitative uncertainty is represented in logic programming using disjunctive logic programs [17, 10, 2]. It often happens that $a \vee b \vee c$ occurs because we are uncertain which of these propositions is true [2]. There might be states of the world where a is true or b is true or c is true or any combinations of them might also be true [2]. Quantitative uncertainty is represented in logic programming by means of different formalisms including probability theory (see [27] for a survey). Probabilistic logic programming is motivated by the need to provide the ability to represent both logical as well as probabilistic knowledge by logic programs (see [28] for a survey). The semantics of such frameworks provide ways to systematically derive logical conclusions along with their associated probabilistic properties. Although, representing and reasoning with both forms of uncertainty is needed in some real-world applications [6], this issue has not been addressed by the current work in qualitative or quantitative uncertainty in logic programming.

We propose to combine disjunctive logic programs [10, 2] with Extended and Normal Hybrid Probabilistic Logic Programs (EHPP [25] and NHPP [28]) in a unified logic programming framework, to allow directly and intuitively to represent and reason in the presence of both qualitative and quantitative uncertainty. This is achieved by introducing the notions of *Extended and Normal Disjunctive Hybrid Probabilistic Logic Programs* (EDHPP and NDHPP). EDHPP and NDHPP generalize extended and normal disjunctive logic programs of classical logic programming [10, 2], respectively, as well as, generalizing EHPP and NHPP [25, 28]. The semantics of EDHPP and NDHPP are based on the answer set semantics and stable model semantics of extended and normal disjunctive logic programs [10, 2], as well as the probabilistic answer set semantics and the stable probabilistic model semantics of EHPP and NHPP [25, 28]. The semantics of EDHPP employs the *Open World Assumption*, whereas, the semantics of NDHPP employs the *Closed World Assumption*. Therefore, any event represented by a program in NDHPP is associated with a probability interval. Any event that cannot be derived from a program in NDHPP is assigned the probability $[0, 0]$, by default. But, an event

that can be derived from the program is assigned a probability $[a, b] \neq [0, 0]$. However, in EDHPP events may not be assigned probability intervals to represent information incompleteness. If this is the case we say that the probabilities associated with these events are *unknown* or *undecidable*. We show that EDHPP naturally subsumes extended disjunctive logic programs[10] and EHPP [25], and NDHPP naturally subsumes normal disjunctive logic programs [2] and NHPP [28]. Moreover, we show that the probabilistic answer set semantics of EDHPP is reduced to the stable probabilistic model semantics of NDHPP. The importance of that is, computational methods developed for NDHPP can be applied to the language of EDHPP. Moreover, we show that EDHPP subsumes Baral et al.'s answer set programming approach for probabilistic reasoning with causal Bayes networks [1]. We show that some commonsense probabilistic knowledge can be easily represented in EDHPP and NDHPP.

Another reason why the proposed languages are interesting is that, in addition to allowing representing and reasoning with both qualitative and quantitative uncertainty, they can also be used in some real-world applications in which quantitative uncertainty needs to be defined over qualitative uncertainty, where probabilistic measures are assigned over the possible outcomes of qualitative uncertainty. For example, flipping a fair coin leads to a head or tail with 0.5 probability each. This fact can be implicitly represented as a disjunctive logic program (since both events are equally likely) as $head(coin)$ *or* $tail(coin)$ with $\{head(coin)\}$ and $\{tail(coin)\}$ as the possible answer sets, according to the answer set semantics [10]. However, the explicit representation of probabilities and the explicit assignment of probabilities to the possible outcome of flipping the coin cannot be presented by disjunctive logic programs syntax and semantics. Moreover, consider if the coin is biased to the head, where flipping the coin produced a head with 0.58 probability or a tail with 0.42 probability. In this case a disjunctive logic program cannot represent it neither implicitly nor explicitly. On the other hand, the coin-flipping example cannot be represented intuitively and directly in NHPP or EHPP either, since a corresponding notion of disjunctions is not allowed in NHPP or EHPP.

1.1 Probabilistic Logic Programming Approaches

The current work in the literature supports either qualitative uncertainty [17, 10, 2] or quantitative uncertainty [12, 18, 23, 24, 29, 19, 20, 21, 4, 15, 16, 3, 27, 28, 25]. The closest to our work are the frameworks presented in [27, 28, 25, 29, 16, 1].

Hybrid Probabilistic Logic Programs (HPP) [27] is probabilistic logic programming frameworks that modifies the original Hybrid Probabilistic Logic Programming framework of [4], and generalizes and modifies the *probabilistic annotated logic programming framework*, originally proposed in [19] and further extended in [20]. Probabilities in [27] are presented as intervals where a probability interval represents the bounds on the degree of belief a rational agent has about the truth of an event. The semantics of HPP [27], intuitively, captures the probabilistic reasoning about how likely are the various events to occur. It was shown that the HPP [27] framework is more suitable than [4] for reasoning and decision making tasks. In addition, it subsumes Lakshmanan and Sadri's [14] probabilistic implication-based framework as well as being a natural extension of classical logic programming. As a step towards enhancing its reasoning capabilities, the framework of HPP was extended to cope with non-monotonic negation [28] by introducing the notion of Normal Hybrid Probabilistic Logic Programs (NHPP) and to provide two different semantics, namely stable probabilistic model semantics and well-founded probabilistic model semantics. Furthermore, NHPP was extended to Extended Hybrid Probabilistic Logic Programs (EHPP) [25] to cope directly with classical negation as well as non-monotonic negation to allow reasoning in the presence of incomplete knowledge.

In [28], it was shown that the relationship between the stable probabilistic model semantics and the well-founded probabilistic model semantics of NHPP *preserves* the relationship between the stable model semantics and the well-founded semantics for normal logic programs [8]. More importantly, the stable probabilistic models semantics *naturally extends* the stable model semantics [9] of normal logic programs and the well-founded probabilistic model semantics *naturally extends* the well-founded semantics [8] of normal logic programs. A consequence of this is that efficient algorithms and implementations for computing those semantics can be developed by extending the existing efficient algorithms and implementations for computing the stable model semantics and the well-found semantics for normal logic programs, e.g., SMODELS [22]. However, NHPP is developed to represent and reason in the presence of quantitative uncertainty.

However, in [25], it was shown that EHPP explicitly encodes negative information, which is important to provide the capability to reason with incomplete knowledge. The semantics of EHPP relies on a probabilistic generalization of the answer set semantics, originally developed for extended logic programs [10].

The probabilistic answer set semantics of EHPP naturally extends the answer set semantics for classical extended logic programs [10]. Moreover, it was shown that Baral et al.'s probabilistic logic programming approach for reasoning with causal Bayes networks (P-log) [1] is naturally subsumed by EHPP. Furthermore, it was shown that the probabilistic answer set semantics of EHPP is reduced to the stable probabilistic model semantics of NHPP proposed in [28]. The importance of that is computational methods developed for NHPP can be applied to the language of EHPP. Moreover, it was described in [25] that some commonsense probabilistic knowledge can be easily represented in EHPP. Similar to NHPP, EHPP is used to represent and reason in the presence of quantitative uncertainty.

Although [29] allows disjunctions in the head of rules, the probabilistic logic programming framework in [29] is used to represent and reason with quantitative uncertainty to reason with Bayes networks. In addition, EDHPP (NDHPP) is more expressive than [29], since, for example EDHPP, unlike [29], allows classical negation, non-monotonic negation, different modes of probabilistic combinations (since [29] considers independence of probabilities which is a fixed mode of probabilistic combination), and compound events to appear in the body of rules, as well as, Bayes reasoning and representation.

Similar to [29], another approach for probabilistic logic programming has been provided in [16] for quantitative uncertainty reasoning. In [16], a possible world semantics for reasoning about probabilities has been introduced by assigning probabilistic measures over the possible worlds using normal disjunctive logic programs. A probabilistic logic program in [16] consists of a set of normal disjunctive logic program clauses with associated probabilities. A normal disjunctive clause in [16] is treated as a classical formula with an associated probability, where the implication in such a clause is treated as material implication. In addition, an approximate semantics for probabilistic logic programming in [16] has been presented, where probabilities are treated as a lattice of truth values. In this case, the probability of a conjunction $Prob(A \wedge B) = min(Prob(A), Prob(B))$ and the probability of a 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 addition, unlike [16], we allow classical negation and compound events to appear in the body of rules.

A logical approach has been presented in [1] to reason with causal Bayes networks by considering a body of logical knowledge, by using the answer set semantics of classical answer set programming [10]. Although, full answer set programming (logic programs with classical negation, non-monotonic negation, and disjunctions) is used, the probabilistic logic programming framework in [1] is used to reason in the presence of quantitative uncertainty. Answer set semantics [10] has been used to emulate the possible world semantics. Probabilistic logic programs of [1] is expressive and straightforward and relaxed some restrictions on the logical knowledge representation part existed in similar approaches to Bayesian reasoning, e.g., [12, 18, 23, 24, 29]. Since [19, 20, 21, 4] provided a different semantical characterization to probabilistic logic programming, it was not clear that how these proposals relate to [1]. However, the work presented in this paper and [28, 25], which are modification and generalization of the work presented in [19, 20, 21, 4], are closely related to [1]. The framework presented in this paper, as well as the framework of [25], is strictly syntactically and semantically subsumes probabilistic logic programs of [1]. This can be argued by the fact that EDHPP naturally extends classical extended disjunctive logic programs with answer set semantics [10], and probabilistic logic programs of [1] mainly rely on extended disjunctive logic programs with answer set semantics [10] as a knowledge representation and inference mechanism for reasoning with causal Bayes networks. In this sense, the comparisons established between [1] and the existing probabilistic logic programming approaches such as [12, 18, 23, 24, 29, 19, 20, 21, 4, 15, 16, 3] also carry over to EDHPP and these approaches. In addition, unlike [1], EDHPP does not put any restriction on the type of dependency existing among events.

1.2 Paper Organization

This paper is organized as follows. Sections 2 and 3 describe the syntax, stable probabilistic model semantics of NDHPP, and the probabilistic answer set semantics of EDHPP. Finally, conclusions with some perspectives are presented in section 4.

2 Syntax

In this section we introduce the basic notions associated to the languages of EDHPP and NDHPP described throughout the rest of the paper [4, 28, 25]. EDHPP (NDHPP) is EHPP (NHPP) with disjunctions of annotated literals (atoms) in the head of rules.

2.1 Probabilistic Strategies

Let $C[0,1]$ denotes the set of all closed intervals in $[0,1]$. In the context of EDHPP, probabilities are assigned to primitive events (literals) and compound events (conjunctions or disjunctions of literals) as intervals in $C[0,1]$. Let $[\alpha_1,\beta_1],[\alpha_2,\beta_2] \in C[0,1]$. Then the *truth order* asserts that $[\alpha_1,\beta_1] \leq_t [\alpha_2,\beta_2]$ iff $\alpha_1 \leq \alpha_2$ and $\beta_1 \leq \beta_2$. The set $C[0,1]$ and the relation $\leq_t$ form a complete lattice. 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 $c([\alpha_1,\beta_1],[\alpha_2,\beta_2]) = [c_1(\alpha_1,\alpha_2), c_2(\beta_1,\beta_2)]$. 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, e, from the probability range of a compound event that contains e. 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 $M = \{\!\{[\alpha_1,\beta_1],\ldots,[\alpha_n,\beta_n]\}\!\}$, we use cM to denote $c([\alpha_1,\beta_1], c([\alpha_2,\beta_2],\ldots,c([\alpha_{n-1},\beta_{n-1}],[\alpha_n,\beta_n]))\ldots)$. 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 [4, 27] for the formal definitions).

2.2 The Languages of EDHPP and NDHPP

Let $\mathcal{L}$ be an arbitrary first-order language with finitely many predicate symbols, function symbols, constants, and infinitely many variables. 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 $\mathcal{L}$ is denoted by $\mathcal{B}_\mathcal{L}$. A literal is either an atom a or the negation of an atom $\neg a$, where $\neg$ is the classical negation. We denote the set of all literals in $\mathcal{L}$ by Lit. More formally, $Lit = \{a | a \in \mathcal{B}_\mathcal{L}\} \cup \{\neg a | a \in \mathcal{B}_\mathcal{L}\}$. 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 monotonic 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 EDHPP are *hybrid literals*. Let us consider a set of literals $l_1,\ldots,l_n$ and the p-strategies ρ and ρ'. Then $l_1 \wedge_\rho \ldots \wedge_\rho l_n$ and $l_1 \vee_{\rho'} \ldots \vee_{\rho'} l_n$ are called *hybrid literals*. A hybrid literal L is ground if each literal l_i in L is ground. $bf_S(Lit)$ is the set of all ground hybrid literals formed using distinct literals from Lit and p-strategies from S, such that for any collection of equivalent hybrid literals, $\mathsf{Y} = \{l_1 *_\rho l_2 *_\rho \ldots *_\rho l_n,\ l_2 *_\rho l_1 *_\rho \ldots *_\rho l_n, \ldots\}$, where $* \in \{\wedge,\vee\}$, only one $l_{i_1} *_\rho l_{i_2} *_\rho \ldots *_\rho l_{i_n} \in \mathsf{Y}$ is in $bf_S(Lit)$. An *annotated hybrid literal* is an expression of the form $L : \mu$, where L is a hybrid literal and μ is an annotation. Note that any hybrid literal L can be represented in terms of another hybrid literal L' such that $L = \neg L'$, since $\neg\neg a = a$, $(a_1 \wedge_\rho a_2) = \neg(\neg a_1 \vee_\rho \neg a_2)$ and $(a_1 \vee_{\rho'} a_2) = \neg(\neg a_1 \wedge_{\rho'} \neg a_2)$ and $\wedge_\rho, \vee_\rho, \vee_{\rho'}$, and $\wedge_{\rho'}$ are associative and commutative.

However, the building blocks of the language of NDHPP are *hybrid basic formulae*. Let us consider a collection of atoms $a_1,\ldots,a_n$ and the p-strategies ρ and 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*. A hybrid basic formula F is ground if each atom A_i in F is ground. $bf_S(\mathcal{B}_\mathcal{L})$ is the set of all ground hybrid basic formulae formed using distinct atoms from $\mathcal{B}_\mathcal{L}$ and p-strategies from S, such that for any collection of equivalent hybrid basic formulae, $\mathsf{X} = \{a_1 *_\rho a_2 *_\rho \ldots *_\rho a_n,\ a_2 *_\rho a_1 *_\rho \ldots *_\rho a_n, \ldots\}$, where $* \in \{\wedge,\vee\}$, only one $a_{i_1} *_\rho a_{i_2} *_\rho \ldots *_\rho a_{i_n} \in \mathsf{X}$ is in $bf_S(\mathcal{B}_\mathcal{L})$. 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.

3 Extended and Normal Disjunctive Hybrid Probabilistic Logic Programs

In this section we define the syntax, declarative semantics, the probabilistic answer set semantics of *Extended Disjunctive Hybrid Probabilistic Logic Programs (EDHPP)*, and the stable probabilistic model semantics of *Normal Disjunctive Hybrid Probabilistic Logic Programs (NDHPP)*.

Definition 1 (Rules) *An extended disjunctive hybrid probabilistic rule (ed-rule) is an expression of the form*

$$l_1 : \nu_1 \ or \ldots or\ l_k : \nu_k \leftarrow L_1 : \mu_1,\ldots,L_m : \mu_m,\ not\ (L_{m+1} : \mu_{m+1}),\ldots,not\ (L_n : \mu_n),$$

whereas a normal disjunctive hybrid probabilistic rule (nd-rule) is an expression of the form

$$A_1 : \nu_1 \ or \ldots or\ A_k : \nu_k \leftarrow F_1 : \mu_1,\ldots,F_m : \mu_m,\ not\ (F_{m+1} : \mu_{m+1}),\ldots,not\ (F_n : \mu_n),$$

where $l_1, \ldots, l_k$ are literals, $A_1, \ldots, A_k$ are atoms, L_i $(1 \leq i \leq n)$ are hybrid literals, F_i $(1 \leq i \leq n)$ are hybrid basic formulae, and $\nu_i (1 \leq i \leq k), \mu_i$ $(1 \leq i \leq n)$ are annotations.

An ed-rule^{not} is an ed-rule without non-monotonic negation—i.e., $n = m$, and a d-rule is an nd-rule without non-monotonic negation—i.e., $n = m$.

The intuitive meaning of an ed-rule, in Definition 1, is that, if for each $L_i : \mu_i$, where $1 \leq i \leq m$, the probability interval of L_i is at least μ_i and for each *not* $(L_j : \mu_j)$, where $m+1 \leq j \leq n$, it is *not known (undecidable)* that the probability interval of L_j is at least μ_j, then there exist at least l_i, where $1 \leq i \leq k$, such that the probability interval of l_i is at least ν_i. However, the meaning of an nd-rule, is that, if for each $F_i : \mu_i$, where $1 \leq i \leq m$, the probability interval of F_i is at least μ_i and for each *not* $(F_j : \mu_j)$, where $m+1 \leq j \leq n$, it is *not provable* that the probability interval of F_j is at least μ_j, then there exist at least A_i, where $1 \leq i \leq k$, such that the probability interval of A_i is at least ν_i.

Definition 2 (Programs) *An extended (normal) disjunctive hybrid probabilistic logic program over S,* ed-program *(*nd-program*), is a pair $P = \langle R, \tau \rangle$, where R is a finite set of ed-rules (nd-rules) with p-strategies from S, and τ is a mapping $\tau : Lit \rightarrow S_{disj}$ $(\tau : \mathcal{B}_\mathcal{L} \rightarrow S_{disj})$. An extended (normal) disjunctive hybrid probabilistic logic program without non-monotonic negation is an ed-program (nd-program) where each rule in the program is an ed-rule^{not} (d-rule).*

The mapping τ in the above definition associates to each literal l_i (similarly for atoms in nd-programs) a disjunctive p-strategy that will be employed to combine the probability intervals obtained from different rules having l_i in their heads. An ed-program (nd-program) is ground if no variables appear in any of its rules.

3.1 Satisfaction and Models

In this subsection, we define the declarative semantics of EDHPP and NDHPP. We define the notions of interpretations, models, and satisfaction of ed-programs and nd-programs.

Definition 3 *A probabilistic interpretation (p-interpretation) of an ed-program is a partial or total mapping $h : bf_S(Lit) \rightarrow C[0,1]$. A probabilistic interpretation (p-interpretation) for an nd-program is a total mapping $h : bf_S(\mathcal{B}_\mathcal{L}) \rightarrow C[0,1]$.*

Since we allow both an event and its negation to be defined in p-interpretations for ed-programs, more conditions need to be imposed on p-interpretations to ensure their consistency. This can be characterized by the following definitions.

Definition 4 *A total (partial) p-interpretation h for an ed-program is inconsistent if there exists $L, \neg L \in bf_S(Lit)$ $(L, \neg L \in dom(h))$ such that $h(\neg L) \neq [1,1] - h(L)$.*

Definition 5 *We say a set C, a subset of Lit, is a set of consistent literals if there is no pair of complementary literals a and $\neg a$ belonging to C. Similarly, a consistent set of hybrid literals C^* is a subset of $bf_S(Lit)$ such that there is no pair of complementary hybrid literals F and $\neg F$ belonging to C^*.*

Definition 6 *A consistent p-interpretation h of an ed-program is either not inconsistent or maps a consistent set of hybrid literals C^* to $C[0,1]$.*

The notion of truth order can be extended to p-interpretations of nd-programs. Given p-interpretations h_1 and h_2 of an nd-program P, we say $(h_1 \leq_t h_2) \Leftrightarrow (\forall F \in bf_S(\mathcal{B}_\mathcal{L}) : h_1(F) \leq_t h_2(F))$. The set of all p-interpretations of P and the truth order $\leq_t$ form a complete lattice. In addition, given the p-interpretations h_1 and h_2 for an ed-program P', we say $(h_1 \leq_o h_2) \Leftrightarrow (dom(h_1) \subseteq dom(h_2)$ *and* $\forall L \in dom(h_1),\ h_1(L) \leq_t h_2(L))$. The set of all p-interpretations of P' and the partial order $\leq_o$ form a complete lattice.

Definition 7 (Probabilistic Satisfaction) *Let $P = \langle R, \tau \rangle$ be a ground ed-program, h be a p-interpretation, and*

$$r \equiv l_1 : \nu_1 \text{ or } \ldots \text{ or } l_k : \nu_k \leftarrow L_1 : \mu_1, \ldots, L_m : \mu_m, \text{not } (L_{m+1} : \mu_{m+1}), \ldots, \text{not } (L_n : \mu_n).$$

Then

- *h satisfies $L_i : \mu_i (l_i : \nu_i)$ (denoted by $h \models L_i : \mu_i (h \models l_i : \nu_i)$) iff $L_i \in dom(h) (l_j \in dom(h))$ and $\mu_i \leq_t h(L_i) (\nu_i \leq_t h(l_i))$.*
- *h satisfies not $(L_j : \mu_j)$ (denoted by $h \models$ not $(L_j : \mu_j)$) iff $L_j \in dom(h)$ and $h(L_j) <_t \mu_j$ or $L_j \notin dom(h)$.*
- *h satisfies*

  $$Body \equiv L_1 : \mu_1, \ldots, L_m : \mu_m, \text{not } (L_{m+1} : \mu_{m+1}), \ldots, \text{not } (L_n : \mu_n)$$

  (denoted by $h \models Body$) iff $\forall (1 \leq i \leq m), h \models L_i : \mu_i$ and $\forall (m+1 \leq j \leq n), h \models$ not $(L_j : \mu_j)$.
- *h satisfies $Head \equiv l_1 : \nu_1$ or $\ldots$ or $l_k : \nu_k$ (denoted by $h \models Head$) iff there exists at least i $(1 \leq i \leq k)$ such that $h \models l_i : \nu_i$.*
- *h satisfies $Head \leftarrow Body$ iff $h \models Head$ whenever $h \models Body$ or h does not satisfy $Body$.*
- *h satisfies P iff h satisfies every ed-rule in R and for every literal $l_i \in dom(h)$,*

$c_{\tau(l_i)}\{\!\{\nu_i \ (1 \leq i \leq k) \mid l_1 : \nu_1 \ or \ldots or \ l_k : \nu_k \leftarrow Body \in R, h \models Body, \ and \ h \models l_i : \nu_i\}\!\} \leq_t h(l_i)$.

Observe that the definition of probabilistic satisfaction for nd-programs is the same as the definition of probabilistic satisfaction for ed-programs described in Definition 7. The only difference is that classical negation is not allowed in nd-programs, in addition, p-interpretations of nd-programs are total mappings from $bf_S(\mathcal{B}_\mathcal{L})$ to $C[0,1]$.

Definition 8 (Models) *A probabilistic model (*p-model*) of an ed-program (nd-program), with or without non-monotonic negation, P is a p-interpretation of P that satisfies P.*

Definition 9 (Minimal Models) *Let P be an ed-program (nd-program). A p-model h of P is* minimal *w.r.t. $\leq_o$ ($\leq_t$) iff there does not exist a p-model h' of P such that $h' <_o h$ ($h' <_t h$).*

We call a minimal p-model of an ed-program a *probabilistic answer set*. It is possible to get a probabilistic answer set of an ed-program, P, and this probabilistic answer set is inconsistent. If this is the case, we say P is inconsistent. If P is inconsistent, LIT, where $LIT : bf_S(Lit) \rightarrow [1,1]$, is the probabilistic answer set of P. We adopt this view from the answer set semantics of classical logic programming [10].

Example 1 *Consider the following ed-program $P = \langle R, \tau\rangle$, without non-monotonic negation, where R contains*

$a : [0.1, 0.2]$	*or*	$\neg b : [0.15, 0.3]$
$\neg c : [0, 0.21]$	$\leftarrow$	$a : [0.1, 0.13]$
$d : [0.12, 0.18]$	$\leftarrow$	$\neg b : [0.1, 0.21]$
$\neg d : [0.45, 0.55]$	$\leftarrow$	$a : [0, 0.15], \neg b : [0.02, 0.22],$ $\neg c : [0.1, 0.1]$

and τ is any arbitrary assignment of disjunctive p-strategies. It is easy to verify that P has two probabilistic answer sets h_1 and h_2, where $h_1(a) = [0.1, 0.2]$ $h_1(\neg\ c) = [0, 0.21]$ *and* $h_2(\neg\, b) = [0.15, 0.3]$ $h_2(d) = [0.12, 0.18]$.

3.2 Probabilistic Answer Set and Stable Probabilistic Model Semantics

In this subsection we define the *probabilistic answer set* and the *stable probabilistic model* semantics of ed-programs and nd-programs respectively. The semantics are defined in two steps. First, we guess a probabilistic answer set (stable probabilistic model) h for a certain ed-program (nd-program) P, then we define the notion of the probabilistic reduct of P with respect to h. The probabilistic reduct is an ed-program (nd-program) without non-monotonic negation. Second, we determine whether h is a probabilistic answer set (stable probabilistic model) for P. This is verified by determining whether h is a probabilistic answer set (minimal p-model) of the probabilistic reduct of P w.r.t. h.

Definition 10 (Probabilistic Reduct) *Let $P = \langle R, \tau\rangle$ be a ground ed-program (nd-program) and h be a p-interpretation. The probabilistic reduct P^h of P w.r.t. h is $P^h = \langle R^h, \tau\rangle$ where:*

$$R^h = \left\{ \begin{array}{l} l_1 : \nu_1 \ or \ldots or \ l_k : \nu_k \leftarrow L_1 : \mu_1, \ldots, L_m : \mu_m \mid \\ \quad l_1 : \nu_1 \ or \ldots or \ l_k : \nu_k \leftarrow L_1 : \mu_1, \ldots, L_m : \mu_m, \\ \quad not \ (L_{m+1} : \mu_{m+1}), \ldots, not \ (L_n : \mu_n) \in R \ and \\ \quad \forall (m+1 \leq j \leq n),\ h(L_j) <_t \mu_j \ or \ L_j \notin dom(h) \end{array} \right.$$

Note that the definitions of the probabilistic reduct for ed-programs and nd-programs are similar. Except that classical negation is not allowed in nd-programs. In addition, p-interpretations in nd-programs are total mappings from $bf_S(\mathcal{B}_\mathcal{L})$ to $C[0,1]$, therefore, for nd-programs, the condition $L_j \notin dom(h)$ is not applicable.

The probabilistic reduct P^h is an ed-program (nd-program) without non-monotonic negation. For any *not* $(L_j : \mu_j)$ in the body of $r \in R$ with $h(L_j) <_t \mu_j$ means that it is *not known (not provable for nd-program)* that the probability interval of L_j is at least μ_j given the available knowledge, and *not* $(L_j : \mu_j)$ is removed from the body of r. In addition, for ed-program, if $L_j \notin dom(h)$, i.e., L_j is undefined in h, then it is completely *not known (undecidable)* that the probability interval of L_j is at least μ_j. In this case, *not* $(L_j : \mu_j)$ is also removed from the body of r. If $\mu_j \leq_t h(L_j)$ (similarly for nd-programs), then we know that the probability interval of L_j is at least μ_j and the body of r is not satisfied and r is trivially ignored.

Definition 11 *A p-interpretation h of an ed-program (nd-program) P is a probabilistic answer set (stable probabilistic model) of P if h is a minimal p-model of P^h.*

The domain of a probabilistic answer set of an ed-program or a stable probabilistic model of an nd-program represents an agent set of beliefs. However, the probability intervals associated to these beliefs bound the agents belief degrees on these beliefs. ed-programs without classical negation (nd-programs), i.e., ed-programs that contain no negative literals neither in head nor in the body of ed-rules, have probabilistic answer sets with hybrid literals consist of only atoms (hybrid basic formulae). Moreover, the definition of probabilistic answer sets coincides with the definition of stable probabilistic models for nd-programs. This means that the application of the probabilistic answer set semantics to nd-programs is reduced to the

stable probabilistic model semantics for nd-programs. However, there are a couple of main differences between the two semantics. A probabilistic answer set may be a partial p-interpretation, however, a stable probabilistic model is a total p-interpretation. In addition, each hybrid basic formula F with probability interval [0,0] in a stable probabilistic model of an nd-program corresponds to the fact that the probability interval of F is unknown, and hence undefined, in its equivalent probabilistic answer set.

Proposition 1 *Let P be an ed-program without classical negation. Then h is a probabilistic answer set for P iff h' is a stable probabilistic model of P, where $h(F) = h'(F)$ for each $h'(F) \neq [0,0]$ and $h(F)$ is undefined for each $h'(F) = [0,0]$.*

Proposition 1 suggests that there is a simple reduction from ed-programs to nd-programs. The importance of that is, under the consistency condition, computational methods developed for nd-programs can be applied to ed-programs.

Example 2 *Consider the following example adapted from [11]. Tom and Fred are two policemen who are challenging their firing gun skills, by shooting a bottle at a quite long distance. In one of the shoots, at the same time, both Tom and Fred shoot a bottle and the bottle shattered. In fact, we cannot determine whether Tom or Fred is the one who shattered the bottle. However, from Tom's shooting experience on similar targets at similar distances, Tom is capable of hitting targets with probability interval from 75% to 80%. Similarly, Fred can hit similar targets with probability interval from 72% to 87%. Normally, a shooter shoots a target. If a shooter sneezes while shooting, it is an exception. Hence, a shooter's shoot is abnormal with probability interval from 30% to 65% if a shooter sneezes while shooting. It was heard that somebody sneezed, however, we do not know whether Tom or Fred is the one who sneezed. A shooter shatters a bottle with probability interval from 82% to 90% if a shooter is capable of hitting similar targets with probability interval from 70% to 79%, and it is not known that a shooter's shoot is abnormal with probability interval from 30% to 60%. This can be represented by the following ed-program $P = \langle R, \tau \rangle$, where R contains:*

$sneeze(tom) : [1,1] \quad or \quad sneeze(fred) : [1,1] \leftarrow$
$ab(shoot, X) : [0.3, 0.65] \leftarrow shoot(X) : [1,1],$
$\qquad sneeze(X) : [1,1]$
$shatter(X) : [0.82, 0.9] \leftarrow hit(X) : [0.7, 0.79],$
$\qquad not\ (ab(shoot, X) : [0.3, 0.6])$
$shoot(tom) : [1,1] \leftarrow$
$shoot(fred) : [1,1] \leftarrow$
$hit(tom) : [0.75, 0.8] \leftarrow$
$hit(fred) : [0.72, 0.87] \leftarrow$

and τ is any arbitrary assignment of disjunctive p-strategies. The ed-rules in Example 2 encode two forms of uncertainty. Qualitative uncertainty represented by the first ed-rule that arises from the fact that we do not know whether Tom or Fred is the one who sneezed. And quantitative uncertainty represented by the probability intervals associated to the various events presented in R. The probability interval $[1,1]$ represents the truth value *true*. Therefore, the rule $sneeze(tom) : [1,1]$ *or* $sneeze(fred) : [1,1] \leftarrow$ is intuitively interpreted as a disjunctive rule in classical disjunctive logic programming. The above ed-program P has two probabilistic answer sets h_1 and h_2, where

$h_1(sneeze(fred)) = [1,1]$
$h_1(ab(shoot, fred)) = [0.3,0.65]$
$h_1(shatter(tom)) = [0.82, 0.9]$
$h_1(shoot(tom)) = [1,1]$
$h_1(shoot(fred)) = [1,1]$
$h_1(hit(tom)) = [0.75,0.8]$
$h_1(hit(fred)) = [0.72,0.87]$

$h_2(sneeze(tom)) = [1,1]$
$h_2(ab(shoot, tom)) = [0.3,0.65]$
$h_2(shatter(fred)) = [0.82, 0.9]$
$h_2(shoot(tom)) = [1,1]$
$h_2(shoot(fred)) = [1,1]$
$h_2(hit(fred)) = [0.72,0.87]$
$h_2(hit(tom)) = [0.75,0.8]$

For example, h_1 can be verified as a probabilistic answer set of P by computing the probabilistic reduct, $P^{h_1} = \langle R^{h_1}, \tau \rangle$, of P w.r.t. h_1, where R^{h_1} contains

$sneeze(tom) : [1,1] \quad or \quad sneeze(fred) : [1,1] \leftarrow$
$ab(shoot, tom) : [0.3, 0.65] \leftarrow shoot(tom) : [1,1],$
$\qquad sneeze(tom) : [1,1]$
$ab(shoot, fred) : [0.3, 0.65] \leftarrow shoot(fred) : [1,1],$
$\qquad sneeze(fred) : [1,1]$
$shatter(tom) : [0.82, 0.9] \leftarrow hit(tom) : [0.7, 0.79]$
$shoot(tom) : [1,1] \leftarrow$
$shoot(fred) : [1,1] \leftarrow$
$hit(tom) : [0.75, 0.8] \leftarrow$
$hit(fred) : [0.72, 0.87] \leftarrow$

It can be easily seen that h_1 is a probabilistic answer set for P^{h_1}.

Example 3 *Assume that either we believe that Tom is the one who hit the bottle or we believe that Fred is the one who hit the bottle. However, if Tom is the one who hit the bottle he can only hit it with probability interval from 75% to 80%. Similarly, if Fred is the one who hit the bottle he can only hit it with probability interval from 72% to 87%. This means that either Tom hit the bottle with probability interval from 75% to 80% or Fred hit the bottle with probability interval from 72% to 87%. This leads to the following encoding of the ed-program $P = \langle R, \tau \rangle$ presented in Example 2, where R now contains:*

$$
\begin{array}{l}
hit(tom) : [0.75, 0.8] \quad or \quad hit(fred) : [0.72, 0.87] \leftarrow \\
sneeze(tom) : [1,1] \quad or \quad sneeze(fred) : [1,1] \leftarrow \\
ab(shoot, X) : [0.3, 0.65] \leftarrow shoot(X) : [1,1], \\
\qquad\qquad\qquad\qquad\qquad sneeze(X) : [1,1] \\
shatter(X) : [0.82, 0.9] \leftarrow hit(X) : [0.7, 0.79], \\
\qquad\qquad\qquad\qquad\qquad not\ (ab(shoot, X) : [0.3, 0.6]) \\
shoot(tom) : [1,1] \leftarrow \\
shoot(fred) : [1,1] \leftarrow
\end{array}
$$

and τ is any arbitrary assignment of disjunctive p-strategies. The first ed-rule in R presents that quantitative uncertainty (the probability intervals $[0.75, 0.8]$ and $[0.72, 0.87]$) can be defined over qualitative uncertainty, where probabilistic measures are assigned over the possible outcomes ($hit(tom)$ and $hit(fred)$) of qualitative uncertainty. The above ed-program P has four probabilistic answer sets h_1, h_2, h_3, and h_4, where

$$
\begin{array}{lcl}
h_1(hit(tom)) & = & [0.75,0.8] \\
h_1(sneeze(tom)) & = & [1,1] \\
h_1(ab(shoot, tom)) & = & [0.3,0.65] \\
h_1(shoot(tom)) & = & [1,1] \\
h_1(shoot(fred)) & = & [1,1]
\end{array}
$$

$$
\begin{array}{lcl}
h_2(hit(fred)) & = & [0.72,0.87] \\
h_2(sneeze(fred)) & = & [1,1] \\
h_2(ab(shoot, fred)) & = & [0.3,0.65] \\
h_2(shoot(tom)) & = & [1,1] \\
h_2(shoot(fred)) & = & [1,1]
\end{array}
$$

$$
\begin{array}{lcl}
h_3(hit(tom)) & = & [0.75,0.8] \\
h_3(sneeze(fred)) & = & [1,1] \\
h_3(ab(shoot, fred)) & = & [0.3,0.65] \\
h_3(shatter(tom)) & = & [0.82,\ 0.9] \\
h_3(shoot(tom)) & = & [1,1] \\
h_3(shoot(fred)) & = & [1,1]
\end{array}
$$

$$
\begin{array}{lcl}
h_4(hit(fred)) & = & [0.72,0.87] \\
h_4(sneeze(tom)) & = & [1,1] \\
h_4(ab(shoot, tom)) & = & [0.3,0.65] \\
h_4(shatter(fred)) & = & [0.82,\ 0.9] \\
h_4(shoot(tom)) & = & [1,1] \\
h_4(shoot(fred)) & = & [1,1]
\end{array}
$$

For example, h_3 can be verified as a probabilistic answer set of P by computing the probabilistic reduct, $P^{h_3} = \langle R^{h_3}, \tau \rangle$, of P w.r.t. h_3, where R^{h_3} contains

$$
\begin{array}{l}
hit(tom) : [0.75, 0.8] \quad or \quad hit(fred) : [0.72, 0.87] \leftarrow \\
sneeze(tom) : [1,1] or \qquad sneeze(fred) : [1,1] \leftarrow \\
ab(shoot, tom) : [0.3, 0.65] \leftarrow shoot(tom) : [1,1], \\
\qquad\qquad\qquad\qquad\qquad sneeze(tom) : [1,1] \\
ab(shoot, fred) : [0.3, 0.65] \leftarrow shoot(fred) : [1,1], \\
\qquad\qquad\qquad\qquad\qquad sneeze(fred) : [1,1] \\
shatter(tom) : [0.82, 0.9] \leftarrow hit(tom) : [0.7, 0.79] \\
shoot(tom) : [1,1] \leftarrow \\
shoot(fred) : [1,1] \leftarrow
\end{array}
$$

It can be easily seen that h_3 is a probabilistic answer set for P^{h_3}.

Now we show that EDHPP and NDHPP naturally extend EHPP and NHPP respectively.

Proposition 2 *The probabilistic answer set semantics of EDHPP is equivalent to the probabilistic answer set semantics of EHPP [25] for all ed-programs $P = \langle R, \tau \rangle$ such that $\forall\ r \in R, k = 1$. In addition, the stable probabilistic model semantics of NDHPP is equivalent to the stable probabilistic model semantics of NHPP [28] for all nd-programs $P = \langle R, \tau \rangle$ such that $\forall\ r \in R, k = 1$.*

Let us show that the probabilistic answer set semantics of EDHPP and the stable probabilistic model semantics of NDHPP generalize the answer set semantics and the stable model semantics of extended and normal disjunctive logic programs [2, 10] respectively. An extended disjunctive logic program P can be represented as an ed-program $P' = \langle R, \tau \rangle$ where each extended disjunctive rule

$$l_1\ or \ldots or\ l_k \leftarrow l'_1, \ldots, l'_m, not\ l'_{m+1}, \ldots, not\ l'_n \in P$$

can be represented, in R, as an ed-rule of the form

$$
\begin{array}{l}
l_1 : [1,1]\ or \ldots or\ l_k : [1,1] \leftarrow l'_1 : [1,1], \ldots, l'_m : [1,1], \\
\qquad\qquad not\ (l'_{m+1} : [1,1]), \ldots, not\ (l'_n : [1,1]) \in R
\end{array}
$$

where $l_1, \ldots, l_k, l'_1, \ldots, l'_n$ are literals and $[1,1]$ represents the truth value *true*. τ is any arbitrary assignment of disjunctive p-strategies. We call the class of ed-programs that consists of only ed-rules of the above form as $EDHPP_1$. Recall that nd-programs are ed-programs without classical negation. $NDHPP_1$ is the same as $EDHPP_1$, except that, only atoms (positive literals) are allowed to appear in rules of the above form. The following result shows that $EDHPP_1$ and $NDHPP_1$ subsume classical extended and normal disjunctive logic programs [2, 10].

Proposition 3 *Let P_1 be an extended disjunctive logic program. Then $S_1{}'$ is an answer set of P_1 iff h_1 is a probabilistic answer of $P_1{}' \in EDHPP_1$ that corre-*

sponds to P_1 *where* $h_1(l) = [1, 1]$ *iff* $l \in S_1{}'$ *and* $h_1(l')$ *is undefined iff* $l' \notin S_1{}'$. *Let* P_2 *be a normal disjunctive logic program. Then* $S_2{}'$ *is a stable model of* P_2 *iff* h_2 *is a stable probabilistic model of* $P_2{}' \in NDHPP_1$ *that corresponds to* P_2 *where* $h_2(a) = [1, 1]$ *iff* $a \in S_2{}'$ *and* $h_2(b) = [0, 0]$ *iff* $b \in \mathcal{B}_\mathcal{L} \setminus S_2{}'$.

In the following result, we show that EDHPP naturally subsumes the probabilistic logic programming framework (P-log) of [1]. This means that any P-log program can be represented as an ed-program. In [1], a logical approach has been presented to reason with causal Bayes networks, by considering a body of logical knowledge, using the answer set semantics of classical logic programming [1]. Answer set semantics has been used to emulate the possible world semantics in [1].

Proposition 4 *The language of EDHPP subsumes P-log, a probabilistic logic programming framework for reasoning with causal Bayes networks [1].*

4 Conclusions and Future Work

We extended Extended and Normal Hybrid Probabilistic Logic Programs [25, 28] to Extended and Normal Disjunctive Hybrid Probabilistic Logic Programs, to allow classical negation, non-monotonic negation, and disjunctions in the head of rules. The extension is necessary to provide the capability of reasoning in the presence of both qualitative and quantitative uncertainty in a unified logic programming framework. In addition to the ability to assign quantitative uncertainly over qualitative uncertainty, where probabilistic measures are assigned over the possible outcomes of qualitative uncertainty. We developed semantical characterizations of the extended languages, which rely on generalizations of the answer set semantics and the stable model semantics, originally developed for extended and normal disjunctive logic programs [10, 2], and the probabilistic answer set semantics and the stable probabilistic model semantics for Extended and Normal Hybrid Probabilistic Logic Programs [25, 28]. We showed that the probabilistic answer set semantics of EDHPP naturally generalizes the answer set semantics of extended disjunctive logic programs [10] and the probabilistic answer set semantics of EHPP [25]. In addition, the stable probabilistic model semantics of NDHPP generalizes the stable model semantics of normal disjunctive logic programs [2] and the stable probabilistic model semantics of NHPP [28]. Furthermore, we showed that the probabilistic answer set semantics of EDHPP is reduced to stable probabilistic model semantics of NDHPP. The importance of that is computational methods developed for NDHPP can be applied to the language of EDHPP. Moreover, we showed that some commonsense probabilistic knowledge can be easily represented in EDHPP and NDHPP. In addition, we showed that EDHPP naturally subsumes the probabilistic logic programming framework of [1].

The main topic of future research is to investigate the computational aspects of the probabilistic answer set semantics of EDHPP and stable probabilistic model semantics of NDHPP—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 answer set (stable model) semantics for extended (normal) disjunctive logic programs, e.g., DLV [7].

References

[1] C. Baral, M. Gelfond, and N. Rushton. Probabilistic reasoning with answer sets. *In Proc. 7th International Conference on Logic Programming and Nonmonotonic Reasoning*, Springer Verlag, 2004.

[2] G. Brewka and J. Dix. Knowledge representation with logic programs. *Third International Workshop on Logic Programming and Knowledge Representation*, 1997.

[3] A. Dekhtyar and I. Dekhtyar. Possible worlds semantics for probabilistic logic programs. *International Conference of Logic Programming*, 137-148, 2004.

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

[5] M. Dekhtyar, A. Dekhtyar, and V. S. Subrahmanian. Hybrid Probabilistic Programs: Algorithms and Complexity. *In Proc. of Uncertainty in Artificial Intelliegence*, pages 160-169, 1999.

[6] T. Eiter and T. Lukasiewicz. Probabilistic reasoning about actions in nonmonotonic causal theories. *In Proc. of Uncertainty in Artificial Intelliegence*, pp. 192-199, 2003.

[7] T. Eiter et al. Declarative problem solving in dlv. *In Logic Based Artificial Intelligence*, 2000.

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

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

[10] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9(3-4):363-385, 1991.

[11] J. Halpern and J. Pearl. Causes and explanations: A structural-model approach. Part I: Causes. *The British Journal for the Philosophy of Science*, 56(4):843-887, 2005.

[12] K. Kersting and L. De Raedt. Bayesian Logic Programs. *In Proc. Inductive Logic Programming*, 2000.

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

[14] 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.

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

[16] T. Lukasiewicz. Many-valued disjunctive logic programs with probabilistic semantics. *In Proc. International Conference on Logic Programming and Nonmonotonic Reasoning*, 1999.

[17] J. Fernandez and J. Minker. Disjunctive deductive databases. *In Proc. International Conference on Logic Programming and Automated Reasoning*, 1992.

[18] S. Muggleton. Stochastic logic programming. *In Proc. 5th International Workshop on Inductive Logic Programming*, 1995.

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

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

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

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

[23] D. Poole. The Independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1-2), 7-56, 1997.

[24] D. Poole. Abducing through negation as failure: stable models within the independent choice logic. *Journal of Logic Programming*, Vol 44, 5-35, 2000.

[25] E. Saad. Incomplete knowlege in hybrid probabilistic logic programs. *In Proc. 10th European Conference on Logics in Artificial Intelligence*, 2006.

[26] E. Saad. Classical negation in hybrid probabilistic logic programs. *Submitted*.

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

[28] E. Saad and E. Pontelli. Hybrid probabilistic logic programs with non-monotonic negation. *In Proc. International Conference of Logic Programming*. Springer Verlag, 2005.

[29] J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. *In Proc. International Conference of Logic Programming*, 431-445, 2004.