Probability Aggregates in Probability Answer Set Programming

Emad Saad
emsaad@gmail.com

Abstract
Probability answer set programming [Saad and Pontelli, 2006] is a declarative programming framework which aims to solve hard search problems in probability environments, and shown effective for probability knowledge representation and probability reasoning applications. It has been shown that many interesting probability reasoning problems are represented and solved by probability answer set programming, where probability answer sets describe the set of possible solutions to the problem. These probability reasoning problems include, but not limited to, reasoning about actions with probability effects and probability planning [Saad, 2007b], reinforcement learning in MDP environments [Saad, 2008a], reinforcement learning in POMDP environments [Saad, 2011], contingent probability planning [Saad, 2009], and Bayesian reasoning [Saad, 2008b]. However, the unavailability of probability aggregates, e.g., expected values, in the language of probability answer set programming disallows the natural and concise representation of many interesting problems. In this paper, we extend DHPP to allow arbitrary probability aggregates. We introduce two types of probability aggregates: a type that computes the expected value of a classical aggregate, e.g., the expected value of the minimum, and a type that computes the probability of a classical aggregate, e.g., the probability of sum of values. In addition, we define a probability answer set semantics for DHPP with arbitrary probability aggregates including monotone, antimonotone, and nonmonotone probability aggregates. We show that the proposed probability answer set semantics of DHPP subsumes both the original probability answer set semantics of DHPP [Saad, 2007a] and the classical answer set semantics of classical disjunctive logic programs with classical aggregates [Faber et al., 2010], and consequently subsumes the classical answer set semantics of the original disjunctive logic programs [Gelfond and Lifschitz, 1991]. We show that the proposed probability answer sets of DHPP with probability aggregates are minimal probability models and hence incomparable, which is an important property for nonmonotonic probability reasoning.

1 Introduction
Probability answer set programming [Saad and Pontelli, 2006] is a declarative programming framework which aims to solve hard search problems in probability environments, and shown effective for probability knowledge representation and probability reasoning applications. It has been shown that many interesting probability reasoning problems are represented and solved by probability answer set programming, where probability answer sets describe the set of possible solutions to the problem. These probability reasoning problems include, but not limited to, reasoning about actions with probability effects and probability planning [Saad, 2007b], reinforcement learning in MDP environments [Saad, 2008a], reinforcement learning in POMDP environments [Saad, 2011], contingent probability planning [Saad, 2009], and Bayesian reasoning [Saad, 2008b]. However, the unavailability of probability aggregates, e.g., expected values, in the language of probability answer set programming disallows the natural and concise representation of many interesting problems. This requires probability answer set programs to be capable of representing and reasoning in the presence of probability aggregates. The following stochastic dietary problem illuminates the need for probability aggregates.

Example 1 Suppose we have three kinds of food: beef, fish, and turkey, where the amounts of vitamins of A, B, and C per unit of each of these foods are uncertain. Two scenarios are available for each amount of units of vitamins for each unit of food. The amounts of units of vitamins A, B, and C per unit of beef are believed to be (60, 10, 20) with (0.7, 0.6, 0.8) probability and (50, 8, 15) with (0.3, 0.4, 0.2) probability. Per unit of fish, the amounts of units of vitamins are believed to be (8, 15, 10) with (0.8, 0.5, 0.4) probability and (11, 18, 13) with (0.2, 0.5, 0.6) probability. Per unit of turkey, the amounts of units of vitamins are believed to be (60, 15, 20) with (0.8, 0.7, 0.9) probability and (55, 20, 25) with (0.2, 0.3, 0.1) probability.

Assume each kind of food is available in packages of 1 or 2 units, presented by the predicate pckg(F, N, S), where F is a food, N is the number of units of the food F, and S is the scenario in which the package is selected. We use units(F, V, U, S) : P to represent a unit of food F has U units of vitamin V with probability P in a scenario.
S. The minimum daily requirement of vitamins A, B, and C is 230, 75, and 95 units, respectively.

The target is to find combinations of units of food that meet the minimum daily requirement of each vitamin. This requires finding the expected value of units of vitamins for each vitamin collected from each available food in every possible scenario, and compare this expected value with the minimum daily requirement of each vitamin.

This probability optimization problem can be represented by a disjunctive hybrid probability logic program with probability answer set semantics, DHPP [Saad, 2007a]. DHPP is an expressive probability answer set programming framework [Saad and Pontelli, 2006; Saad, 2006] that allows disjunctions in the head of rules. We assume that atoms appearing without annotations, in DHPP programs, are associated with the annotation \([1,1]\), and annotated atoms of the form \(A : [\alpha, \alpha]\) are simply represented as \(A : \alpha\). The DHPP program representation, \(\Pi = \langle R, \tau \rangle\), of the stochastic dietary problem, is given as follows, where \(\tau = \{\text{any arbitrary assignment of disjunctive p-strategies and } R\) contains rules of the form:

\[
\begin{align*}
\text{food} & (\text{bee}f) \leftarrow \text{food} (\text{fish}) \leftarrow \text{food} (\text{turkey}) \leftarrow \\
\text{units} (\text{bee}f, a, 60, s_1) : 0.7 & \leftarrow \text{units} (\text{bee}f, b, 10, s_1) : 0.6 \\
\text{units} (\text{bee}f, a, 50, s_2) : 0.3 & \leftarrow \text{units} (\text{bee}f, b, 8, s_2) : 0.4 \\
\text{units} (\text{fish}, a, 8, s_1) : 0.8 & \leftarrow \text{units} (\text{fish}, b, 15, s_1) : 0.5 \\
\text{units} (\text{fish}, a, 11, s_2) : 0.2 & \leftarrow \text{units} (\text{fish}, b, 8, s_2) : 0.5 \\
\text{units} (\text{turk}, a, 60, s_1) : 0.8 & \leftarrow \text{units} (\text{turk}, b, 15, s_1) : 0.7 \\
\text{units} (\text{turk}, a, 55, s_2) : 0.2 & \leftarrow \text{units} (\text{turk}, b, 20, s_2) : 0.3 \\
\text{units} (\text{bee}f, a, 20, s_1) : 0.8 & \leftarrow \text{units} (\text{bee}f, c, 15, s_2) : 0.2 \\
\text{units} (\text{fish}, c, 10, s_1) : 0.4 & \leftarrow \text{units} (\text{fish}, c, 13, s_2) : 0.6 \\
\text{units} (\text{turk}, c, 20, s_1) : 0.9 & \leftarrow \text{units} (\text{turk}, c, 25, s_2) : 0.1 \\
\text{pkcg} (F, 1, S) \lor \text{pkcg} (F, 2, S) & \leftarrow \text{food} (F) \\
\text{nutr} (F, V, U \times N, S) : P & \leftarrow \text{units} (F, V, U, S) : P, \text{pkcg} (F, N, S) \\
\text{expected} (a, U_1 \ast P_1 + U_2 \ast P_2 + U_3 \ast P_3 + U_4 \ast P_4 + U_5 \ast P_5 + U_6 \ast P_6) & \leftarrow \text{nutr} (\text{bee}f, a, U_1, s_1) : P_1, \text{nutr} (\text{bee}f, a, U_2, s_2) : P_2, \text{nutr} (\text{fish}, a, U_3, s_1) : P_3, \text{nutr} (\text{fish}, a, U_4, s_2) : P_4, \text{nutr} (\text{turk}, a, U_5, s_1) : P_5, \text{nutr} (\text{turk}, a, U_6, s_2) : P_6 \\
\text{expected} (b, U_1 \ast P_1 + U_2 \ast P_2 + U_3 \ast P_3 + U_4 \ast P_4 + U_5 \ast P_5 + U_6 \ast P_6) & \leftarrow \text{nutr} (\text{bee}f, b, U_1, s_1) : P_1, \text{nutr} (\text{bee}f, b, U_2, s_2) : P_2, \text{nutr} (\text{fish}, b, U_3, s_1) : P_3, \text{nutr} (\text{fish}, b, U_4, s_2) : P_4, \text{nutr} (\text{turk}, b, U_5, s_1) : P_5, \text{nutr} (\text{turk}, b, U_6, s_2) : P_6 \\
\text{expected} (c, U_1 \ast P_1 + U_2 \ast P_2 + U_3 \ast P_3 + U_4 \ast P_4 + U_5 \ast P_5 + U_6 \ast P_6) & \leftarrow \text{nutr} (\text{bee}f, c, U_1, s_1) : P_1, \text{nutr} (\text{bee}f, c, U_2, s_2) : P_2, \text{nutr} (\text{fish}, c, U_3, s_1) : P_3, \text{nutr} (\text{fish}, c, U_4, s_2) : P_4, \text{nutr} (\text{turk}, c, U_5, s_1) : P_5, \text{nutr} (\text{turk}, c, U_6, s_2) : P_6 \\
\Gamma & \leftarrow \text{not } \Gamma, \text{expected} (a, X) \land X < 230 \\
\Gamma & \leftarrow \text{not } \Gamma, \text{expected} (b, X) \land X < 75 \\
\Gamma & \leftarrow \text{not } \Gamma, \text{expected} (c, X) \land X < 95
\end{align*}
\]

The last three rules in the above DHPP program representation of the stochastic dietary problem guarantee that only probability answer sets with sufficient supply of vitamins are generated.

The DHPP representation of the stochastic dietary problem described in Example 1 is fairly intuitive but rather complex, since the rules that represent the expected value of units of vitamins for each vitamin via the predicate \(\text{expected}(V, E)\), where \(E\) is the expected value of units of vitamins for vitamin \(V\), contains complex summation that involves 12 variables. Furthermore, this representation strategy is not feasible in general, especially, in the presence of multiple scenarios for each amount of units of vitamin per unit of food, multiple numbers of vitamins, and multiple types of food, which consequently will lead to very complex rules with very complex summations.

Therefore, we propose to extend the language of DHPP with probability aggregates to allow intuitive and concise representation and reasoning about real-world applications. To the best of our knowledge, this development is the first that defines semantics for probability aggregates in a probability answer set programming framework. DHPP is expressive form of probability answer set programming [Saad and Pontelli, 2006] that allows disjunctions in the head of rules. It has been shown that; DHPP is capable of representing and reasoning with both probability uncertainty and qualitative uncertainty [Saad, 2007a]; it is a natural extension to the classical disjunctive logic programs, DLP, and its probability answer set semantics generalizes the classical answer set semantics of DLP [Saad, 2007a]; DHPP with probability answer set semantics generalizes the probability answer set programming framework of [Saad and Pontelli, 2006], which are DHPP programs with an atom appearing in the heads of rules. Moreover, it has been shown that DHPP is used in real-world applications in which quantitative probability uncertainly need to be defined over the possible outcomes of qualitative uncertainty [Saad, 2007a].

There were many proposals for defining semantics for classical aggregates in classical answer set programming [Faber et al., 2010; Niemela and Simons, 2000; Pelov et al., 2007; Pelov and Truszczynski, 2004; Ferraris and Lifschitz, 2005; Ferraris and Lifschitz, 2010; Pelov, 2004]. Among these proposals, [Faber et al., 2010] is the most general intuitive semantics for classical aggregates in DLP. In [Faber et al., 2010], declarative classical answer semantics for classical disjunctive logic program with arbitrary classical aggregates, denoted by DLP\(^A\), including monotone, antimonotone, and nonmonotone aggregates, was provided. The proposed classical answer set semantics of DLP\(^A\) generalizes the classical answer set semantics of aggregate-free DLP. Moreover, classical answer sets of DLP\(^A\) are subset-minimal [Faber et al., 2010], a vital property for nonmonotonic reasoning framework semantics.

The contributions of this paper are the following. We extend the original language of DHPP to allow any arbitrary probability annotation function including monotone, antimonotone, and nonmonotone annotation functions. We define the notions of probability aggregates
and probability aggregate atoms in DHPP. We present two types of probability aggregates; the first type computes the expected value of a classical aggregate, e.g., the expected value of the minimum, the second type computes the probability of a classical aggregate, e.g., the probability of sum of values. In addition, we define the probability answer set semantics of DHPP with arbitrary probability aggregates, denoted by DHPP\textsuperscript{PA}, including monotone, antimonotone, and nonmonotone probability aggregates. We show that the proposed probability answer set semantics of DHPP\textsuperscript{PA} subsumes both the original probability answer set semantics of DHPP [Saad, 2007a] and the classical answer set semantics of DLP\textsuperscript{A} [Paber et al., 2010], and consequently subsumes the classical answer set semantics of DLP [Gelfond and Lifschitz, 1991]. We show that the probability answer sets of DHPP\textsuperscript{PA} are minimal probability models and hence incomparable, which is an important property for nonmonotonic probability reasoning.

2 DHPP\textsuperscript{PA} : Probability Aggregates

Disjunctive Hybrid Probability Logic Programs

In this section we introduce the basic language of DHPP\textsuperscript{PA}, the notions of probability aggregates and probability aggregate atoms, and the syntax of DHPP\textsuperscript{PA} programs.

2.1 The Basic Language of DHPP\textsuperscript{PA}

Let \( \mathcal{L} \) denotes an arbitrary first-order language with finitely many predicate symbols, function symbols, constants, and infinitely many variables. A term is a constant, a variable or a function. An atom, \( a \), is a predicate in \( \mathcal{B}_\mathcal{L} \), where \( \mathcal{B}_\mathcal{L} \) is the Herbrand base of \( \mathcal{L} \). The Herbrand universe of \( \mathcal{L} \) is denoted by \( \mathcal{U}_\mathcal{L} \). Non-monotonic negation or the negation as failure is denoted by \( \neg \).

In probability aggregates disjunctive hybrid probability logic programs, DHPP\textsuperscript{PA} probabilities are assigned to primitive events (atoms) and compound events (conjunctions or disjunctions of atoms) as intervals in \( \mathcal{C}[0,1] \), where \( \mathcal{C}[0,1] \) denotes the set of all closed intervals in \([0,1]\). For \( [\alpha_1, \beta_1], [\alpha_2, \beta_2] \in \mathcal{C}[0,1] \), the truth order \( \leq_t \) on \( \mathcal{C}[0,1] \) is defined as \( [\alpha_1, \beta_1] \leq_t [\alpha_2, \beta_2] \) iff \( \alpha_1 \leq \alpha_2 \) and \( \beta_1 \leq \beta_2 \).

The type of dependency among the primitive events within a compound event is described by a probability strategy, which can be a conjunctive p-strategy or a disjunctive p-strategy. Conjunctive (disjunctive) p-strategies are used to combine events belonging to a conjunctive (disjunctive) formula [Saad and Pontelli, 2006]. The probability composition function, \( c_\rho \), of a probability strategy (p-strategy), \( \rho \), is a mapping \( c_\rho : \mathcal{C}[0,1] \times \mathcal{C}[0,1] \rightarrow \mathcal{C}[0,1] \), where the probability composition function, \( c_\rho \), computes the probability interval of a conjunction (disjunction) of two events from the probability of its components. Let \( M = \{ [\alpha_1, \beta_1], \ldots, [\alpha_n, \beta_n] \} \) be a multiset of probability intervals. For convenience, we use \( c_\rho M \) to denote \( c_\rho([\alpha_1, \beta_1], c_\rho([\alpha_2, \beta_2], \ldots, c_\rho([\alpha_n, \beta_n])) \ldots) \).

A probability annotation is a probability interval of the form \( [\alpha_1, \alpha_2] \), where \( \alpha_1, \alpha_2 \) are called probability annotation items. A probability annotation item is either a constant in \([0,1]\) (called probability annotation constant), a variable ranging over \([0,1]\) (called probability annotation variable), or \( f(\alpha_1, \ldots, \alpha_n) \) (called probability annotation function), where \( f \) is a representation of a monotone, antimonotone, or nonmonotone total or partial function \( f : ([0,1])^n \rightarrow [0,1] \) and \( \alpha_1, \ldots, \alpha_n \) are probability annotation items.

Let \( S = S_{\text{conj}} \cup S_{\text{disj}} \) be an arbitrary set of p-strategies, where \( S_{\text{conj}} (S_{\text{disj}}) \) is the set of all conjunctive (disjunctive) p-strategies in \( S \). A hybrid basic formula is an expression of the form \( \alpha_1 \land \rho \ldots \land \rho \alpha_n \) or \( \alpha_1 \lor \rho \ldots \lor \rho \alpha_n \), where \( \alpha_1, \ldots, \alpha_n \) are atoms and \( \rho \) and \( \rho' \) are p-strategies. Let \( bfs(B_L) \) be the set of all ground hybrid basic formulae formed using distinct atoms from \( B_L \) and p-strategies from \( S \). If \( A \) is a hybrid basic formula and \( \mu \) is a probability annotation then \( A : \mu \) is called a probability annotated hybrid basic formula.

2.2 Probability Aggregate Atoms

A symbolic probability set is an expression of the form \( \{ F : [P_1, P_2] \mid C \} \), where \( F \) is a variable or a function term and \( P_1, P_2 \) are probability annotation variables or probability annotation functions, and \( C \) is a conjunction of probability annotated hybrid basic formulae. A ground probability set is a set of pairs of the form \( \{ F^g : [P_1^g, P_2^g] \mid C^g \} \) such that \( F^g \) is a constant term and \( P_1^g, P_2^g \) are probability annotation constants, and \( C^g \) is a ground conjunction of probability annotated hybrid basic formulae. A symbolic probability set or ground probability set is called a probability set term. Let \( f \) be a probability aggregate function symbol and \( S \) be a probability set term, then \( f(S) \) is said a probability aggregate, where \( f \in \{ \text{val}, \text{sum}, \text{times}, \text{minE}, \text{maxE}, \text{countE}, \text{sumP}, \text{timesP}, \text{minP}, \text{maxP} \} \). If \( f(S) \) is a probability aggregate and \( T \) is an interval \( [\theta_1, \theta_2] \), called guard, where \( \theta_1, \theta_2 \) are constants, variables or functions terms, then we say \( f(S) < T \) is a probability aggregate atom, where \( \prec \in \{ =, \neq, <, >, \leq, \geq \} \).

Example 2 The following examples are representation for probability aggregate atoms.

\[
\begin{align*}
\sum_E \{ X : [P_1, P_2] \mid \text{demand}(X) : [P_1, P_2] \} & \leq \{190, 230\}, \\
\min_P \{ T : [0.2, 0.3] \mid a(T, 1) : [0.2, 0.3] \} & \geq \{0.45, 0.6\}
\end{align*}
\]

Definition 1 below specifies that every probability aggregate function \( f(S) \) has its own set of local variables.

Definition 1 Let \( f(S) \) be a probability aggregate. A variable, \( X \), is a local variable to \( f(S) \) if and only if \( X \) appears in \( S \) and \( X \) does not appear in the DHPP\textsuperscript{PA} rule that contains \( f(S) \).

For example, for the first probability aggregate atom in Example 2, the variables \( X, P_1 \), and \( P_2 \) are local variables to the probability aggregate \( \sum_E \).
Definition 2 A global variable is a variable that is not a local variable.

2.3 DHPP\textsuperscript{PA} Program Syntax

A DHPP\textsuperscript{PA} rule is an expression of the form

\[ a_1 : \mu_1 \lor \ldots \lor a_k : \mu_k \leftarrow A_{k+1} : \mu_{k+1}, \ldots, A_m : \mu_m, \]

not \( A_{m+1} : \mu_{m+1}, \ldots, \) not \( A_n : \mu_n, \)

where \( \forall (1 \leq i \leq k) \ a_i \) are atoms, \( \forall (k+1 \leq i \leq n) \ A_i \) are hybrid basic formulae or probability aggregate atoms, and \( \forall (1 \leq i \leq n) \ \mu_i \) are probability annotations.

A DHPP\textsuperscript{PA} rule says that if for each \( A_i : \mu_i, \) where \( k+1 \leq i \leq m, \) it is \textit{believable} that the probability interval of \( A_i \) is at least \( \mu_i \) \( \mu_i \) w.r.t. \( \leq_c \) and for each not \( A_j : \mu_j, \) where \( m+1 \leq j \leq n, \) it is not \textit{believable} that the probability interval of \( A_j \) is at least \( \mu_j \) \( \mu_j \) w.r.t. \( \leq_c, \) then there exists at least \( a_i, \) where \( 1 \leq i \leq k, \) such that the probability interval of \( a_i \) is at least \( \mu_i. \)

Definition 3 A DHPP\textsuperscript{PA} program over a set of arbitrary p-strategies, \( S = S_{\text{ground}} \cup S_{\text{disj}}, \) is a pair \( \Pi = (R, \tau), \) where \( R \) is a set of DHPP\textsuperscript{PA} rules with p-strategies from \( S, \) and \( \tau \) is a mapping \( \tau : B_E \rightarrow S_{\text{disj}}. \)

The mapping \( \tau \) in the DHPP\textsuperscript{PA} program definition associates to each atom, a, a disjunctive p-strategy that is used to combine the probability intervals obtained from different DHPP\textsuperscript{PA} rules with a appearing in their heads. For the simplicity of the presentation, hybrid basic formulae that appearing in DHPP\textsuperscript{PA} programs without probability annotations are assumed to be associated with the probability annotation \([1, 1].\) Nevertheless, probability annotated hybrid basic formulae of the form \( A : [P, P] \) are simply represented as \( A : P. \)

Example 3 The stochastic dietary problem described in Example 1 can be concisely and intuitively represented as DHPP\textsuperscript{PA} program, \( \Pi = (R, \tau), \) where \( \tau \) is any arbitrary assignments of disjunctive p-strategies and \( R \) consists of the following DHPP\textsuperscript{PA} rules in addition to the facts represented by units \((F, V, U, S) : P \) and food \((X) \) described in Example 1.

\[
\begin{align*}
pckg(F, 1, S) & \lor pckg(F, 2, S) & \leftarrow & \text{food}(F) \\
\text{nutr}(F, V, U \times N, S) : P & \leftarrow & \text{units}(F, V, U, S) : P & \text{pckg}(F, N, S)
\end{align*}
\]

\[
\begin{align*}
\Gamma & \leftarrow \neg \Gamma, \text{val}_E \{X : P \mid \text{nutr}(F, a, X, S) : P\} < 230 \\
\Gamma & \leftarrow \neg \Gamma, \text{val}_E \{X : P \mid \text{nutr}(F, b, X, S) : P\} < 75 \\
\Gamma & \leftarrow \neg \Gamma, \text{val}_E \{X : P \mid \text{nutr}(F, c, X, S) : P\} < 95
\end{align*}
\]

where the expected value is computed by the probability aggregate \text{val}_E. The last three DHPP\textsuperscript{PA} rules of the DHPP\textsuperscript{PA} program representation of the stochastic dietary problem described above guarantee that only probability answer sets that involve sufficient daily supply of each vitamin are generated.

Definition 4 The ground instantiation of a symbolic probability set \( S = \{F : [P_1, P_2] \mid C\} \) is the set of all ground pairs of the form \( (\theta(F) : [\theta(P_1), \theta(P_2)] \mid \theta(C)) \), where \( \theta \) is a substitution of every local variable appearing in \( S \) to a constant from \( U_L. \)

Definition 5 A ground instantiation of a DHPP\textsuperscript{PA} rule, \( r, \) is the replacement of each global variable appearing in \( r \) to a constant from \( U_L, \) then followed by the ground instantiation of every symbolic probability set, \( S, \) appearing in \( r. \)

The ground instantiation of a DHPP\textsuperscript{PA} program, \( \Pi, \) is the set of all possible ground instantiations of every DHPP\textsuperscript{PA} rule in \( \Pi. \)

Example 4 The ground instantiation of the DHPP\textsuperscript{PA} rule

\[
\Gamma \leftarrow \neg \Gamma, \text{val}_E \{X : P \mid \text{nutr}(F, a, X, S) : P\} < 230
\]

with respect to the DHPP\textsuperscript{PA} program, \( \Pi, \) in Example 3, is given by:

\[
\Gamma \leftarrow \neg \Gamma, \text{val}_E \{(60 : 0.7)\text{nutr}(\text{beef}, a, 60, s_1) : 0.7), (120 : 0.7)\text{nutr}(\text{beef}, a, 120, s_1) : 0.7), (50 : 0.3)\text{nutr}(\text{fish}, a, 50, s_2) : 0.3, (100 : 0.3)\text{nutr}(\text{beef}, a, 100, s_2) : 0.3, (22 : 0.2)\text{nutr}(\text{fish}, a, 22, s_2) : 0.2), (60 : 0.8)\text{nutr}(\text{turk}, a, 60, s_1) : 0.8, (120 : 0.8)\text{nutr}(\text{turk}, a, 120, s_1) : 0.8, (55 : 0.2)\text{nutr}(\text{turk}, a, 55, s_2) : 0.2), (110 : 0.2)\text{nutr}(\text{turk}, a, 110, s_2) : 0.2) \ldots < 230
\]

3 Probability Aggregates Semantics

We present two types of probability aggregates. The first type computes the expected value of a classical aggregate, e.g., the expected value of the minimum, denoted by \( f \in \{ \text{val}_E, \text{sum}_E, \text{times}_E, \text{min}_E, \text{max}_E, \text{count}_E \} , \) where \( \text{val}_E \) returns the expected value of a random variable and \( \text{sum}_E, \text{times}_E, \text{min}_E, \text{max}_E, \text{count}_E \) return the expected value of the the classical aggregates \( \text{sum}, \text{times}, \text{min}, \text{max}, \text{count} \) respectively. The second type of probability aggregates computes the probability of a classical aggregate, e.g, the probability of sum of values, denoted by \( g \in \{ \text{sum}_P, \text{times}_P, \text{min}_P, \text{max}_P, \text{count}_P \}, \) where \( \text{sum}_P, \text{times}_P, \text{min}_P, \text{max}_P, \text{count}_P \) return the probability of the the classical aggregates \( \text{sum}, \text{times}, \text{min}, \text{max}, \text{count} \) respectively. Any probability aggregate is applied to a probability set that represents a random variable with all its possible values and their associated probability intervals.

3.1 Probability Aggregates Mappings

Let \( X \) be a set of objects. Then, we use \( 2^X \) to denote the set of all multisets over elements in \( X. \) Let \( R \) denotes the set of all real numbers and \( N \) denotes the set of all natural numbers, and \( U_L \) denotes the Herbrand universe. Let \( \perp \) be a symbol that does not occur in \( L. \) Therefore,

- The mappings for the expected value probability aggregates are:
  - \( \text{val}_E : 2^R \times C[0, 1] \rightarrow [R, R]. \)
  - \( \text{sum}_E : 2^R \times C[0, 1] \rightarrow [R, R]. \)
Let \( C \) be a DHPP\(^{P,A} \) rule and \( head(r) = a_1 : \mu_1 \lor \ldots \lor a_m : \mu_m \) and \( body(r) = A_{k+1} : \mu_{k+1}, \ldots, A_m : \mu_m, not A_{m+1} : \mu_{m+1}, \ldots, not A_n : \mu_n \). We consider that probability annotated probability aggregate atoms that involve probability aggregates from \( \{valE, sumE, timesE, minE, maxE, countE\} \) are associated to the probability annotation \([1,1]\).

**Definition 9** Let \( \Pi = (R, \tau) \) be a ground DHPP\(^{P,A} \) program, \( r \) be a DHPP\(^{P,A} \) rule in \( R \), \( h \) be a p-interpretation for \( \Pi \), \( f \in \{valE, sumE, timesE, minE, maxE, countE\} \), and \( g \in \{sum_P, times_P, min_P, max_P, count_P\} \). Then,

1. \( h \) satisfies \( a_i : \mu_i \) in \( head(r) \) iff \( \mu_i \subseteq h(a_i) \).
2. \( h \) satisfies \( f(S) \prec T : [1,1] \) in \( body(r) \) iff \( f(S_h) \neq \bot \) and \( f(S_h) \prec T \).
3. \( h \) satisfies not \( f(S) \prec T : [1,1] \) in \( body(r) \) iff \( f(S_h) = \bot \) or \( f(S_h) \neq \bot \) and \( f(S_h) \prec T \).
4. \( h \) satisfies \( g(S) \prec T : [1,1] \) in \( body(r) \) iff \( g(S_h) = (x,\nu) \neq \bot \) and \( x \prec T \) and \( \mu \subseteq \nu \).
5. \( h \) satisfies not \( g(S) \prec T : [1,1] \) in \( body(r) \) iff \( g(S_h) = \bot \) or \( g(S_h) = (x,\nu) \neq \bot \) and \( x \nprec T \) or \( \mu \not\subseteq \nu \).
6. \( h \) satisfies \( A_1 : \mu_1 \) in \( body(r) \) iff \( \mu_1 \subseteq h(A_1) \).
7. \( h \) satisfies not \( A_j : \mu_j \) in \( body(r) \) iff \( \mu_j \not\subseteq h(A_j) \).
8. \( h \) satisfies \( body(r) \) iff \( \forall(k + 1 \leq i \leq m), h \) satisfies \( A_i : \mu_i \) and \( \forall(m + 1 \leq j \leq n), h \) satisfies not \( A_j : \mu_j \).
9. \( h \) satisfies \( head(r) \) iff \( \exists i (1 \leq i \leq k) \) such that \( h \) satisfies \( a_i : \mu_i \).
10. \( h \) satisfies \( r \) iff \( h \) satisfies \( head(r) \) whenever \( h \) satisfies \( body(r) \) or \( h \) does not satisfy \( body(r) \).
11. \( h \) satisfies \( \Pi \) iff \( h \) satisfies every DHPP\(^{P,A} \) rule in \( R \) and

\[
\begin{align*}
& c_{r(a_1)} \left( \mu_1 \mid head(r) \right. 
\left. \left. \longleftarrow body(r) \in R \right) \leq t \ h(a_i) \right)
&\text{such that} \ h \text{ satisfies} \ body(r) \text{ and} \ h \text{ satisfies} \ a_1 : \mu_1 \text{ in the head(r).}
& c_{r(h(a_1) \ldots h(a_n))} \leq t \ h(A) \text{ such that} \ a_1, \ldots, a_n \text{ are atoms in} \ B_E \text{ and} \\
& A = a_1 *_p \ldots *_p a_n \text{ is hybrid basic formula in} \ b_f S(B_E) \text{ and} \ * \in \{\land, \lor\}.
\end{align*}
\]
Example 5 Let $\Pi = \langle R, \tau \rangle$ be a DHPP$^A$ program, where $\tau$ is any arbitrary assignments of disjunctive p-strategies and $R$ consists of the DHPP$^A$ rules:

$$a(1, 1) : 0.5 \lor a(2, 1) : 0.5 \leftarrow \sum_P \{X : P \mid (a(X, Y) : P) \geq 3 : 0.3\}$$

The $r$ ground instantiation of $r$ is given by:

$$r' : \Gamma \leftarrow \text{not } \Gamma,$$

$$\sum_P \{X : P \mid (a(X, Y) : P) \geq 3 : 0.3\}$$

Let $h$ be a $p$-interpretation for $\Pi$ that assigns $0.7$ to $a(1, 2)$, $0.5$ to $a(2, 1)$, and $0$ to the remaining hybrid basic formulae in $S_R$. Thus the evaluation of the probability aggregate atom, $\sum_P(S) \geq 3$ in $r'$ w.r.t. to $h$ is given as follows, where

$$S = \{1 : 0.5 \mid (a(1, 1) : 0.5), (2 : 0.5 \mid a(2, 1) : 0.5), (1 : 0.7 \mid a(1, 2) : 0.7), (2 : 0.3 \mid a(2, 2) : 0.3)\}$$

and $S_h = \{1 : 0.7, 2 : 0.5\}$. Therefore, $\sum_P(S) \geq 3 : 0.3$ is satisfied by $h$. This is because $\sum_P(S) \geq (3, 0.35)$, and consequently, the probability annotated probability aggregate atom $\sum_P(S) \geq 3 : 0.3$ is satisfied by $h$. This is because $\sum_P(S) \geq (3, 0.35) \neq \perp$ and $3 \geq 3$ and $0.3 \leq 0.35$

Let $L$ denotes a probability annotated hybrid basic formula, $A : \mu$ or the negation of $A : \mu$, denoted by not $A : \mu$. Let $h_1, h_2$ be two $p$-interpretations. Then, we say that $L$ is monotone if $\forall(h_1, h_2)$ such that $h_1 \leq_t h_2$, it is the case that if $h_1$ satisfies $L$ then $h_2$ also satisfies $L$. However, $L$ is antimonotone if $\forall(h_1, h_2)$ such that $h_1 \leq_t h_2$ it is the case that if $h_2$ satisfies $L$ then $h_1$ also satisfies $L$. But, if $L$ is not monotone or not antimonotone, then we say $L$ is nonmonotone. A probability annotated atom or a probability annotated probability aggregate atom, $a : \mu$, or the negation of a probability annotated atom or the negation of a probability annotated probability aggregate atom, not $a : \mu$, can be monotone, antimonotone or nonmonotone, since their probability annotations are allowed to be arbitrary functions. Moreover, probability aggregate atoms by themselves can be monotone, antimonotone or nonmonotone. This also carry over to probability annotated hybrid basic formulae.

Definition 10 A probability model, $p$-model, for a DHPP$^A$ program, $\Pi$, is a $p$-interpretation for $\Pi$ that satisfies $\Pi$. A $p$-model $h$ for $\Pi$ is $\leq_t$-minimal if there does not exist a $p$-model $h'$ for $\Pi$ such that $h' <_t h$.

Example 6 It can easily verified that the $p$-interpretation, $h$, for DHPP$^A$ program, $\Pi$, described in Example 5, is not a $p$-model for $\Pi$. However, by considering only the relevant hybrid basic formulae, the following $p$-interpretation, $h'$, is a $p$-model for $\Pi$, where $h' = \{a(1, 1) : 0.5, a(1, 2) : 0.7, \ldots\}$.

Definition 11 Let $\Pi = \langle R, \tau \rangle$ be a ground DHPP$^A$ program, $r$ be a DHPP$^A$ rule in $R$, and $h$ be a $p$-interpretation for $\Pi$. Let $h \models \text{body}(r)$ denotes $h$ satisfies body$(r)$. Then, the probability reduct, $\Pi^h$, of $\Pi$ w.r.t. $h$ is a ground DHPP$^A$ program $\Pi^h = \langle R^h, \tau \rangle$ where

$$R^h = \{\text{head}(r) \leftarrow \text{body}(r) \mid r \in R \land h \models \text{body}(r)\}$$

Definition 12 A $p$-interpretation, $h$, of a ground DHPP$^A$ program, $\Pi$, is a probability answer set for $\Pi$ if $h$ is $\leq_t$-minimal $p$-model for $\Pi^h$.

Observe that the definitions of the probability reduct and the probability answer sets for DHPP$^A$ programs are generalizations of the probability reduct and the probability answer sets of the original DHPP programs described in [Saad, 2007a].

Example 7 It can be easily verified that the DHPP$^A$ program presented in Example 5 has three probability answer sets, which by considering relevant hybrid basic formulae are:

$$h_1 = \{a(1, 1) : 0.5, a(1, 2) : 0.7, \ldots\}, h_2 = \{a(1, 1) : 0.5, a(2, 2) : 0.3, \ldots\}, h_3 = \{a(2, 1) : 0.5, a(2, 2) : 0.3, \ldots\}$$

Example 8 The stochastic dietary problem representation by the DHPP$^A$ program described in Example 3 has four probability answer sets, which are:

$$h_1 = \{\text{pckg}(\text{beef}, 2, s_1), \text{pckg}(\text{fish}, 2, s_1), \text{pckg}(\text{turk}, 2, s_1), \text{pckg}(\text{beef}, 1, s_2), \text{pckg}(\text{fish}, 2, s_2), \text{pckg}(\text{turk}, 2, s_2), \text{nutr}(\text{beef}, a, 120, s_1) : 0.7, \text{nutr}(\text{fish}, a, 16, s_1) : 0.8, \text{nutr}(\text{turk}, a, 120, s_1) : 0.8, \text{nutr}(\text{beef}, a, 50, s_2) : 0.3, \text{nutr}(\text{fish}, a, 22, s_2) : 0.2, \text{nutr}(\text{turk}, a, 110, s_2) : 0.2, \text{nutr}(\text{beef}, b, 30, s_1) : 0.7, \text{nutr}(\text{fish}, b, 30, s_1) : 0.5, \text{nutr}(\text{beef}, b, 20, s_1) : 0.6, \text{nutr}(\text{turk}, b, 40, s_2) : 0.3, \text{nutr}(\text{fish}, b, 36, s_2) : 0.5, \text{nutr}(\text{beef}, b, 8, s_2) : 0.4, \text{nutr}(\text{beef}, 40, s_1) : 0.8, \text{nutr}(\text{fish}, c, 26, s_2) : 0.6, \text{nutr}(\text{turk}, c, 50, s_2) : 0.1, \text{nutr}(\text{beef}, c, 15, s_2) : 0.2, \text{nutr}(\text{turk}, c, 40, s_1) : 0.9, \text{nutr}(\text{fish}, c, 20, s_1) : 0.4, \ldots\}$$

$$h_2 = \{\text{pckg}(\text{beef}, 2, s_1), \text{pckg}(\text{fish}, 2, s_1), \text{pckg}(\text{turk}, 2, s_1), \text{pckg}(\text{beef}, 2, s_2), \text{pckg}(\text{fish}, 2, s_2), \text{pckg}(\text{turk}, 2, s_2), \text{nutr}(\text{beef}, a, 120, s_1) : 0.7, \text{nutr}(\text{fish}, a, 16, s_1) : 0.8, \text{nutr}(\text{turk}, a, 120, s_1) : 0.8, \text{nutr}(\text{beef}, a, 100, s_2) : 0.3, \text{nutr}(\text{fish}, a, 22, s_2) : 0.2, \text{nutr}(\text{turk}, a, 110, s_2) : 0.2, \text{nutr}(\text{beef}, b, 20, s_1) : 0.6, \text{nutr}(\text{fish}, b, 30, s_1) : 0.5, \text{nutr}(\text{beef}, b, 30, s_1) : 0.5, \text{nutr}(\text{beef}, b, 20, s_1) : 0.6, \text{nutr}(\text{turk}, b, 40, s_2) : 0.3, \text{nutr}(\text{fish}, b, 36, s_2) : 0.5, \text{nutr}(\text{beef}, b, 8, s_2) : 0.4, \text{nutr}(\text{beef}, c, 40, s_1) : 0.8, \text{nutr}(\text{fish}, c, 26, s_2) : 0.6, \text{nutr}(\text{turk}, c, 50, s_2) : 0.1, \text{nutr}(\text{beef}, c, 15, s_2) : 0.2, \text{nutr}(\text{turk}, c, 40, s_1) : 0.9, \text{nutr}(\text{fish}, c, 20, s_1) : 0.4, \ldots\}$$
h_3 = \{ \text{pckg(beef,2,s1),pckg(fish,2,s1),} \\
\text{pckg(turk,2,s1),pckg(beef,2,s2),pckg(fish,2,s2),} \\
\text{pckg(turk,1,s2),nutr(beef,a,120,s1) : 0.7,} \\
\text{nutr(fish,a,16,s1) : 0.8, nutr(turk,a,120,s1) : 0.8,} \\
\text{nutr(beef,a,100,s2) : 0.3, nutr(fish,a,22,s2) : 0.2,} \\
\text{nutr(turk,a,55,s2) : 0.2, nutr(beef,b,20,s1) : 0.6,} \\
\text{nutr(fish,b,30,s1) : 0.5, nutr(turk,b,30,s1) : 0.7,} \\
\text{nutr(beef,b,16,s2) : 0.4, nutr(fish,b,36,s2) : 0.5,} \\
\text{nutr(turk,b,20,s2) : 0.3, nutr(beef,c,40,s1) : 0.8,} \\
\text{nutr(fish,c,20,s1) : 0.4, nutr(turk,c,40,s1) : 0.9,} \\
\text{nutr(beef,c,30,s2) : 0.2, nutr(fish,c,26,s2) : 0.6,} \\
\text{nutr(turk,c,25,s2) : 0.1, . . . } \}

h_4 = \{ \text{pckg(beef,2,s1),pckg(fish,1,s1),} \\
\text{pckg(turk,2,s1),pckg(beef,2,s2),pckg(fish,2,s2),} \\
\text{pckg(turk,2,s2),nutr(beef,a,120,s1) : 0.7,} \\
\text{nutr(fish,a,8,s1) : 0.8, nutr(turk,a,120,s1) : 0.8,} \\
\text{nutr(beef,a,100,s2) : 0.3, nutr(fish,a,22,s2) : 0.2,} \\
\text{nutr(turk,a,110,s2) : 0.2, nutr(beef,b,20,s1) : 0.6,} \\
\text{nutr(fish,b,15,s1) : 0.5, nutr(turk,b,30,s1) : 0.7,} \\
\text{nutr(beef,b,16,s2) : 0.4, nutr(fish,b,36,s2) : 0.5,} \\
\text{nutr(turk,b,10,s2) : 0.3, nutr(beef,c,40,s1) : 0.8,} \\
\text{nutr(fish,c,10,s1) : 0.4, nutr(turk,c,40,s1) : 0.9,} \\
\text{nutr(beef,c,30,s2) : 0.2, nutr(fish,c,26,s2) : 0.6,} \\
\text{nutr(turk,c,50,s2) : 0.1, 50, . . . } \}

5 DHPP^{PA} Semantics Properties

In this section we study the semantics properties of DHPP^{PA} programs and its relationship to the original classical answer set semantics of disjunctive hybrid probability logic programs, denoted by DHPP^{PA} [Saad, 2007a]; the classical answer set semantics of classical disjunctive logic programs with classical aggregates, denoted by DLP^{PA} [Faber et al., 2010]; and the classical original answer set semantics of classical disjunctive logic programs, denoted by DLP^{PA} [Gelfond and Lifschitz, 1991].

Theorem 1 Let II be a DHPP^{PA} program. The probability answer sets for II are \( \leq_1 \)-minimal p-models for II.

The following theorem shows that the probability answer set semantics of DHPP^{PA} programs subsumes and generalizes the probability answer set semantics of DHPP^{PA} [Saad, 2007a] programs, which are DHPP^{PA} programs without probability aggregate atoms and with only monotone probability annotation functions.

Theorem 2 Let II be a DHPP program and h be a p-interpretation. Then, h is a probability answer set for II iff h is a probability answer set for II w.r.t. the probability answer set semantics of DHPP^{PA} [Saad, 2007a].

In what follows we show that the probability answer set semantics of DHPP^{PA} programs naturally subsumes and generalizes the classical answer set semantics of the classical disjunctive logic programs with the classical aggregates, DLP^{PA} [Faber et al., 2010], which consequently naturally subsumes the classical answer set semantics of the original classical disjunctive logic programs, DLP^{PA} [Gelfond and Lifschitz, 1991].

Any DLP^{PA} program, II, is represented as a DHPP^{PA} program, II' = (R, \tau), where each DLP^{PA} rule in II of the form

\[ a_1 \lor \ldots \lor a_k \leftarrow a_{k+1}, \ldots, a_m, \neg a_{m+1}, \ldots, \neg a_n \]

is represented, in R, as a DHPP^{PA} rule of the form

\[ a_1 : [1,1] \lor \ldots \lor a_k : [1,1] \leftarrow a_{k+1} : [1,1], \ldots, a_m : [1,1], \neg a_{m+1} : [1,1], \ldots, \neg a_n : [1,1] \]

where \( a_1, \ldots, a_k \) are atoms and \( a_{k+1}, \ldots, a_m \) are atoms or probability aggregate atoms whose probability aggregates contain probability sets that involve conjunctions of probability annotated atoms with probability annotation \([1,1]\), where \([1,1]\) represents the truth value true. In addition, \( \tau \) is any arbitrary assignments of disjunctive p-strategies. We call this class of DHPP^{PA} programs as DHPP^{PA} programs. Any DLP program is represented as a DHPP^{PA} program by the same way as DLP^{PA} except that DLP disallows classical aggregate atoms. The following results show that DHPP^{PA} programs subsume both DLP^{PA} and DLP programs.

Theorem 3 Let II' be the DHPP^{PA} program equivalent to a DLP^{PA} program II. Then, h is a probability answer set for II' iff I is a classical answer set for II, where h(a) = [1,1] iff a \in I and h(b) = [0,0] iff b \in B_2 - I.

Proposition 1 Let II' be the DHPP^{PA} program equivalent to a DLP program II. Then, h is a probability answer set for II' iff I is a classical answer set for II, where h(a) = [1,1] iff a \in I and h(b) = [0,0] iff b \in B_2 - I.

6 Conclusions and Related Work

We presented DHPP^{PA} that extends the original language of DHPP with arbitrary probability annotations functions and arbitrary probability aggregate functions that determine the expected value of the classical aggregate functions and the probability of a classical aggregate functions. We introduced the probability answer set semantics of DHPP^{PA} with arbitrary probability aggregates including monotone, antimonotone, and nonmonotone probability aggregates. We have shown that the DHPP^{PA} probability answer set semantics generalize DHPP original probability answer set semantics [Saad, 2007a]. In addition, we proved that the probability answer sets of DHPP^{PA} are minimal probability models and consequently incomparable, which is an important property for nonmonotonic probability reasoning.

To the best of our knowledge, this development is the first in probability logic programming literature to consider probability aggregates in probability logic programming in general and probability answer set programming in particular. Nevertheless, classical aggregates were extensively investigated in classical answer set programming [Faber et al., 2010; Niemela and Simons, 2000; Pelov et al., 2007; Pelov and Truszczynski, 2004].
Ferraris and Lifschitz, 2005; Ferraris and Lifschitz, 2010

Among these investigations, Faber et al., 2010 is the most general intuitive semantics for classical aggregates in DLP, since it is declarative classical answer semantics for classical disjunctive logic program with arbitrary classical aggregates (DLP^A), including monotone, antimonotone, and nonmonotone aggregates, and a natural generalization of the classical answer set semantics of aggregate-free DLP. Gelfond and Lifschitz, 1991. We have shown that the probability answer set semantics of DHPP^P^A subsumes both DLP^A and DLP classical answer set semantics. Extensive comparisons between DLP^A and the existing approaches to classical aggregates can be found in Faber et al., 2010. Among these approaches, Niemela and Simons, 2000 that allows only classical aggregates of the form \textit{sum} and \textit{count}, however, they do not behave intuitively with negative values Ferraris and Lifschitz, 2005. In addition, Ferraris and Lifschitz, 2010 presented classical aggregates for first-order formulae.

On the other hand probability aggregates are studied in probability databases in the context of query evaluation over probability data Jayram et al., 2007; Burdick et al., 2007; Re and Suciu, 2009. Two main approaches are available for defining the semantics of probability aggregate queries in probability databases. The first approach adopted in Jayram et al., 2007; Burdick et al., 2007; Re and Suciu, 2009, applied to OLAP applications, defines the semantics of the probability aggregates queries as the expected value of the aggregate queries over the possible worlds of the probability database. However, the second approach Re and Suciu, 2009, defines the semantics of the probability aggregates queries as the probability of the aggregate queries over the possible worlds of the probability database. The possible world semantics is adopted in defining the semantics of probability aggregate queries in both approaches in Jayram et al., 2007; Burdick et al., 2007; Re and Suciu, 2009. In DHPP^P^A, we considered the two approaches, where probability aggregates are evaluated with respect to a probability answer set, which is considered evaluation over a possible world.

References

Burdick et al., 2007 D. Burdick, P. Deshpande, T.S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. Olap over uncertain and imprecise data. VLDB, 16(1):123–144, 2007.

Faber et al., 2010 W. Faber, N. Leone, and G. Pfeifer. Semantics and complexity of recursive aggregates in answer set programming. Artificial Intelligence, 2010.

Ferraris and Lifschitz, 2005 P. Ferraris and V. Lifschitz. Weight constraints as nested expressions. TPLP, 5:45–74, 2005.

Ferraris and Lifschitz, 2010 P. Ferraris and V. Lifschitz. On the stable model semantics of first-order formulas with aggregates. In Nonmonotonic Reasoning, 2010.

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

Jayram et al., 2007 T.S. Jayram, S. Kale, and E. Vee. Efficient aggregation algorithms for probabilistic data. In Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2007.

Niemela and Simons, 2000 I. Niemela and P. Simons. Extending the smodels system with cardinality and weight constraints. In In Jack Minker, editor, Logic-Based AI, pages 491–521, 2000.

Pelov and Truszczynski, 2004 N. Pelov and M. Truszczynski. Semantics of disjunctive programs with monotone aggregates - an operator-based approach. In NMR, 2004.

Pelov et al., 2007 N. Pelov, M. Denecker, and M. Bruynooghe. Well-founded and stable semantics of logic programs with aggregates. TPLP, 7:355 – 375, 2007.

Pelov, 2004 N. Pelov. Semantics of logic programs with aggregates. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 2004.

Re and Suciu, 2009 C. Re and D. Suciu. The trichotomy of having queries on a probabilistic database. VLDB, 2009.

Saad and Pontelli, 2006 E. Saad and E. Pontelli. A new approach to hybrid probabilistic logic programs. Annals of Mathematics and Artificial Intelligence, 48(3-4):187–243, 2006.

Saad, 2006 E. Saad. Incomplete knowlege in hybrid probability logic programs. In 10th European Conference on Logics in Artificial Intelligence, 2006.

Saad, 2007a E. Saad. A logical approach to qualitative and quantitative reasoning. In 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, 2007.

Saad, 2007b E. Saad. Probability planning in hybrid probability logic programs. In 1st International Conference on Scalable Uncertainty Management, 2007.

Saad, 2008a E. Saad. A logical framework to reinforcement learning using hybrid probability logic programs. In 2nd International Conference on Scalable Uncertainty Management, 2008.

Saad, 2008b E. Saad. On the relationship between hybrid probability logic programs and stochastic satisfiability. In 2nd International Conference on Scalable Uncertainty Management, 2008.

Saad, 2009 E. Saad. Probability planning with imperfect sensing actions using hybrid probability logic programs. In 3rd SUM, 2009.
[Saad, 2011] E. Saad. Learning to act optimally in partially observable markov decision processes using hybrid probability logic programs. In Fifth International Conference on Scalable Uncertainty Management, 2011.