Towards the Computation of Stable Probabilistic Model Semantics

Emad Saad

College of Computer Science and Information Technology
Abu Dhabi University
Abu Dhabi, U.A.E.
emad.saad@adu.ac.ae

Abstract. In [22], a stable model semantics extension of the language of hybrid probabilistic logic programs [21] with non-monotonic negation, normal hybrid probabilistic programs (NHPP), has been developed by introducing the notion of stable probabilistic model semantics. It has been shown in [22] that the stable probabilistic model semantics is a natural extension of the stable model semantics for normal logic programs and the language of normal logic programs is a subset of the language NHPP. This suggests that efficient algorithms and implementations for computing the stable probabilistic model semantics for NHPP can be developed by extending the efficient algorithms and implementation for computing the stable model semantics for normal logic programs, e.g., SMODELS [17]. In this paper, we explore an algorithm for computing the stable probabilistic model semantics for NHPP along with its auxiliary functions. The algorithm we develop is based on the SMODELS [17] algorithms. We show the soundness and completeness of the proposed algorithm. We provide the necessary conditions that these auxiliary functions have to satisfy to guarantee the soundness and completeness of the proposed algorithm. This algorithm is the first to develop for studying computational methods for computing the stable probabilistic models semantics for hybrid probabilistic logic programs with non-monotonic negation.

1 Introduction

Hybrid Probabilistic Programs (HPP) [5] is a probabilistic logic programming framework that enables the user to *explicitly* encode his/her knowledge about the type of dependencies existing between the probabilistic events being described by the programs. HPP generalizes the *probabilistic annotated logic programming framework*, originally proposed in [15] and further extended in [16]. In this paper we study the problem of automating the probabilistic reasoning under the stable probabilistic model (p-model) semantics proposed in [22]. Stable p-model semantics is the first formalism to study non-monotonic negation in hybrid probabilistic programs originally proposed in [5] and further modified and extended in [21]. Stable p-model semantics [22] generalizes both the stable model semantics for normal logic programs [10] and the semantics of hybrid probabilistic logic programs introduced in [21]. The idea in [21] comes upon observing that commonsense

C. Freksa, M. Kohlhase, and K. Schill (Eds.): KI 2006, LNAI 4314, pp. 143–158, 2007.

reasoning about probabilities relies on how *likely* (probable) are the various events to occur, rather than how precise our knowledge about these probabilities is [5]. Hybrid probabilistic programs (HPP) [21] is a probabilistic logic programming framework that enables the user to *explicitly* encode his/her knowledge about the type of dependencies existing between the probabilistic events being described by the programs. Moreover, it has the ability to encode the user's knowledge about how to combine the probabilities of the same event derived from different rules. HPP semantics [21] intuitively, captures the commonsense probabilistic reasoning according to how likely are the various events to occur. In addition, HPP subsumes Lakshmanan and Sadri's [11] probabilistic implication-based framework as well as it is a natural extension of classical logic programming.

In [22], we extended the language of HPP [21] to support non-monotonic negation. In addition, we defined two alternative semantics for the extended language; the stable probabilistic model semantics and the probabilistic well-founded semantics and studied their relationships. We showed that the stable probabilistic model semantics and the probabilistic well-founded semantics generalize the stable model semantics [10] and the well-founded semantics [9] for normal logic programs, and they reduce to the semantics of HPP [21] in the absence of non-monotonic negation. An important result is that the relationship between the stable probabilistic model semantics and the probabilistic well-founded semantics *preserves* the relationship between the stable model semantics and the well-founded semantics for normal logic programs [9].

Since the stable p-model semantics naturally generalize its classical counterpart, hence, this suggests that efficient algorithms and implementations can be developed by *extending* the existing efficient algorithms and implementations developed for computing the stable models for normal logic programs, such as SMODELS [17]. (In [21], we have presented an algorithm for computing the least fixpoint for HPP without non-monotonic negation, that extends the Dowling-Gallier algorithm for computing the satisfiability of a set of Horn formulae [7], which is the ground base for developing the various auxiliary functions in SMODELS.) In this paper, we provide an algorithm for computing the stable p-model semantics for normal hybrid probabilistic programs [22] based on the decision procedure of SMODELS [17] along with its auxiliary functions. We provide the necessary conditions that these auxiliary functions have to satisfy to guarantee the soundness and completeness of the proposed algorithm. We present formal definitions for these auxiliary functions and show that they satisfy the necessary conditions for the soundness and completeness of the proposed algorithm. In this paper, we focus on the the computation of the stable probabilistic model semantics. Motivating examples and extensive comparisons between stable probabilistic model semantics and other related work can be found in [22].

2 Normal Hybrid Probabilistic Programs

In the following subsections, we present the syntax and semantics of the Normal Hybrid Probabilistic Programs (NHPP) as presented in [22]. The notions

of probabilistic strategies, annotations, and hybrid basic formulae, which are defined below, were first introduced in [5].

2.1 Probabilistic Strategies

Let $C[0,1]$ denote the set of all closed intervals in $[0,1]$. In the context of HPP, probabilities are assigned to primitive events (atoms) and compound events (conjunctions or disjunctions of atoms) as intervals in $C[0,1]$. Let $[a_1,b_1],[a_2,b_2] \in C[0,1]$. Then the *truth order* asserts that $[a_1,b_1] \leq_t [a_2,b_2]$ iff $a_1 \leq a_2$ and $b_1 \leq b_2$. The set $C[0,1]$ and the relation $\leq_t$ form a complete lattice. In particular, the join ($\oplus_t$) operation is defined as $[a_1,b_1] \oplus_t [a_2,b_2] = [max\{a_1,a_2\}, max\{b_1,b_2\}]$ and the meet ($\otimes_t$) is defined as $[a_1,b_1] \otimes_t [a_2,b_2] = [min\{a_1,a_2\}, min\{b_1,b_2\}]$ w.r.t. $\leq_t$. The type of dependency among the primitive events within a compound event is described by *probabilistic strategies*, which are explicitly selected by the user. We call ρ, a pair of functions $\langle c, md\rangle$, a probabilistic strategy (p-strategy), where $c : C[0,1] \times C[0,1] \rightarrow C[0,1]$, the *probabilistic composition function*, which is *commutative*, *associative*, *monotonic* w.r.t. $\leq_t$, and meets the following *separation* criteria: there are two functions c_1, c_2 such that $c([a_1,b_1],[a_2,b_2]) = [c_1(a_1,a_2), c_2(b_1,b_2)]$. Whereas, $md : C[0,1] \rightarrow C[0,1]$ is the *maximal interval function*. The maximal interval function md of a certain p-strategy returns an estimate of the probability range of a primitive event, A, from the probability range of a compound event that contains A. The composition function c returns the probability range of a conjunction (disjunction) of two events given the ranges of its constituents. For convenience, given a multiset of probability intervals $M = \{\!\{[a_1,b_1],\ldots,[a_n,b_n]\}\!\}$, we use cM to denote $c([a_1,b_1], c([a_2,b_2],\ldots,c([a_{n-1},b_{n-1}],[a_n,b_n]))\ldots)$. According to the type of combination among events, p-strategies are classified into *conjunctive* p-strategies and *disjunctive* p-strategies. Conjunctive (disjunctive) p-strategies are employed to compose events belonging to a conjunctive (disjunctive) formula (please see [5,21] for the formal definitions).

2.2 Language Syntax

In this subsection, we describe the syntax of NHPP. Let L be an arbitrary first-order language with finitely many predicate symbols, constants, and infinitely many variables. Function symbols are disallowed. In addition, let $S = S_{conj} \cup S_{disj}$ be an arbitrary set of p-strategies, where S_{conj} (S_{disj}) is the set of all conjunctive (disjunctive) p-strategies in S. The Herbrand base of L is denoted by B_L. An *annotation* denotes a probability interval and it is represented by $[\alpha_1,\alpha_2]$, where α_1,α_2 are called annotation items. An *annotation item* is either a constant in $[0,1]$, a variable (*annotation variable*) ranging over $[0,1]$, or $f(\alpha_1,\ldots,\alpha_n)$ (called *annotation function*) where f is a representation of a computable total function $f : ([0,1])^n \rightarrow [0,1]$ and $\alpha_1,\ldots,\alpha_n$ are annotation items. The building blocks of the language of NHPP are *hybrid basic formulae*. Let us consider a collection of atoms $A_1,\ldots,A_n$, a conjunctive p-strategy ρ, and a disjunctive p-strategy ρ'. Then $A_1 \wedge_\rho \ldots \wedge_\rho A_n$ and $A_1 \vee_{\rho'} \ldots \vee_{\rho'} A_n$ are

called *hybrid basic formulae*, and $bf_S(B_L)$ is the set of all ground hybrid basic formulae formed using distinct atoms from B_L and p-strategies from S. An annotated hybrid basic formula is an expression of the form $F : \mu$ where F is a hybrid basic formula and μ is an annotation. A *hybrid literal* is an annotated hybrid basic formula $F : \mu$ (positive annotated hybrid basic formula or positive hybrid literal) or the negation of an annotated hybrid basic formula *not* $(F : \mu)$ (negative annotated hybrid basic formula or negative hybrid literal).

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

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

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

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

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

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

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

Example 1 ([22]). Consider an insurance company which determines the premium categories, by calculating the risk factor according to a genetic test for cancer and the family history for this disease. Assume that customers who have a family history of the disease have a probability of developing cancer with at least 92%. The insurance company will assign high premiums to the customers who have family history of the disease and tested positive as long as their risk conditions are unchanged. Risk conditions can be changed by taking specific medications. This situation can be represented by the following nh-rules:

$risk(X) : [0.9, 1] \leftarrow (test(X) \wedge_{pc} history(X)) : [0.60, 0.75],$ $not\ changeRisk(X)[0.8, 1]$

$risk(X) : [0, 0.1] \leftarrow (test(X) \wedge_{pc} history(X)) : [0.60, 0.75],$ $changeRisk(X) : [0.8, 1]$

$changeRisk(X) : [0.9, 1] \leftarrow medicine(X, Med) : [0.65, 1]$

$highPremium(X) : [1, 1] \leftarrow risk(X) : [0.9, 1]$

$lowPremium(X) : [1, 1] \leftarrow risk(X) : [0, 0.1]$

$test(sam) : [0.92, 1] \leftarrow$

$history(sam) : [0.95, 1] \leftarrow$

$medicine(sam, medication) : [0.98, 1] \leftarrow$

and the mapping τ assigns *ncd* to $risk(sam)$ and an arbitrary disjunctive p-strategy [5,21] to the other hybrid basic formulae. The *ncd* denotes the disjunctive negative correlation p-strategy, which is defined as: $c_{ncd}([a_1, b_1], [a_2, b_2]) = [\min(1, a_1 + a_2), \min(1, b_1 + b_2)]$. The first nh-rule asserts that the risk factor is at least 90% whenever the cancer genetic test for a customer is positive and that customer has a family history of cancer with probability between 60% and 75%, and it is not provable that his risk conditions have changed with probability at least 80%. Observe that *test* and *history* events are conjoined according to the *positive correlation* p-strategy (denoted by $\wedge_{pc}$) where $c_{pcc}([a_1, b_1], [a_2, b_2]) = [\min(a_1, a_2), \min(b_1, b_2)]$. The second rule says that the risk factor is at most 10% whenever the customer risk conditions are changed, even though the person tested positive and have a family history of the disease with probability between 60% and 75%. The third nh-rule describes the change of the risk conditions of a customer with probability at least 90% if a medication for the disease becomes available with probability at least 65%. The fourth and fifth nh-rules assert that definite high premium and low premium are considered whenever the probability of risk factors are at least 90% and at most 10% respectively. The last three nh-rules represent the facts available about a specific customer named sam.

2.3 Satisfaction and Models

In this subsection, we review the declarative semantics of nh-programs [22]. The notion of a probabilistic model (p-model) is based on *hybrid formula functions* defined below.

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

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

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

Definition 4 (Probabilistic Satisfaction). *Let* $P = \langle R, \tau \rangle$ *be a ground nh-program,* h *be a p-interpretation, and*
$r \equiv A : \mu \leftarrow F_1 : \mu_1, \ldots, F_n : \mu_n, not\ (G_1 : \beta_1), \ldots, not\ (G_m : \beta_m) \in R$. *Then*

- h *satisfies* $F_i : \mu_i$ *(denoted by* $h \models F_i : \mu_i$*) iff* $\mu_i \leq_t h(F_i)$.
- h *satisfies not* $(G_j : \beta_j)$ *(denoted by* $h \models not\ (G_j : \beta_j)$*) iff* $\beta_j \not\leq_t h(G_j)$.

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

Definition 5 (Models). *Let P be an nh-program. A probabilistic model (*p-model*) of P is a probabilistic interpretation of P that satisfies P.*

Proposition 1. *Let P be an* h-program. $h_P = \otimes_t \{h | h \text{ is a p-model of } P\}$ *is the least p-model of P.*

Associated with each h-program P, is an operator, T_P, called the *fixpoint operator*, which maps probabilistic interpretations to probabilistic interpretations.

Definition 6. *Let* $P = \langle R, \tau \rangle$ *be a ground h-program and h be a total probabilistic interpretation. The fixpoint operator* T_P *is a mapping* $T_P : HFF \rightarrow HFF$ *which is defined as follows:*

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

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

3 Stable Probabilistic Model Semantics

In this section we introduce the notion of *stable probabilistic models (sp-models)*, which extends the notion of stable models for classical logic programming [10]. The semantics is defined in two steps. First, we guess a p-model h for a certain nh-program P, then we define the notion of the probabilistic reduct of P with respect to h—which is an h-program. Second, we determine whether h is a stable p-model for P or not by employing the fixpoint operator of the probabilistic reduct to verify whether h is its least p-model. It must be noted that every h-program has a unique least p-model [21].

Definition 7 (Probabilistic Reduct). *Let* $P = \langle R, \tau \rangle$ *be a ground nh-program and h be a probabilistic interpretation. The probabilistic reduct* P^h *of P w.r.t. h is* $P^h = \langle R^h, \tau \rangle$ *where:*

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

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

Definition 8 (Stable Probabilistic Model). *A probabilistic interpretation h is a stable p-model of an nh-program P if h is the least p-model of P^h.*

Example 2. It is easy to verify that the only stable p-model of the program in Example 1 is given by:

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

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

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

and $\tau(a) = \tau(b) = \tau(c) = \tau(d) = \pi$ where π is any arbitrary disjunctive p-strategy. This nh-program has two stable p-models h_1 and h_2 where $h_1(a) = [0.89, 0.91], h_1(b) = [0, 0], h_1(c) = [0.2, 0.3], h_1(d) = [0.1, 0.2], h_1(e) = [0, 0]$ and $h_2(a) = [0, 0], h_2(b) = [0.55, 0.60], h_2(c) = [0.2, 0.3], h_2(d) = [0.1, 0.2], h_2(e) = [0, 0]$. Since, for example, h_1 can be verified as a stable p-model because the probabilistic reduct of P w.r.t. h_1 contains the h-rules:

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

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

Theorem 1. *Every h-program P has a unique stable p-model h iff h is the least p-model of P.*

Let us show that the stable probabilistic model semantics generalizes the stable model semantics of normal logic programs [10]. A normal logic program P can be represented as an nh-program $P' = \langle R, \tau \rangle$ where each normal rule

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

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

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

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

Proposition 3. *Let P be a normal logic program. Then S' is a stable model of P iff h is a stable p-model of $P' \in NHPP_1$ that corresponds to P where $h(a) = [1,1]$ iff $a \in S'$ and $h(b) = [0,0]$ iff $b \in B_L \setminus S'$.*

4 An Algorithm for Computing Stable P-Models

In this section, we develop an algorithm for computing the stable p-models for an nh-program, which is based on SMODELS algorithm [17] for computing the stable model semantics for normal logic programs. The algorithm we develop constructs a stable p-model *incrementally*. It takes a ground nh-program P and a partial hybrid formula function (partial p-interpretation) h as inputs and returns true if h can be extended to a stable p-model (which is a total hybrid formula function) for P. Otherwise, it returns false. In the following we provide definitions and notions that we use throughout the rest of the paper. Let h be a probabilistic interpretation, then $dom(h) \subseteq bf_S(B_L)$ denotes the domain of h ($dom(h) \subsetneqq bf_S(B_L)$ if h is a partial probabilistic interpretation). We use $negdom(h)$ to denote the set $\{F \mid F \in dom(h), h(F) = [0,0]\}$. We also define $posdom(h) = dom(h) \backslash negdom(h)$. Given a hybrid formula function h, $Pos(h)$ and $Neg(h)$ denote the following mappings:

- $Pos(h)(F) = h(F)\ \forall F \in dom(h)$ such that $h(F) \neq [0,0]$.
- $Neg(h)(F) = h(F)\ \forall F \in dom(h)$ such that $h(F) = [0,0]$.

We will describe each hybrid formula function using its graph. In other words, if h is a hybrid formula function, then h can be represented as the set $\{(A,\mu) | A \in dom(h)$ and $\mu = h(A)\}$. More conveniently, we use $A : \mu$ to denote (A,μ). Thus, we will frequently refer to h as a *set of annotated hybrid basic formulae*. Furthermore, if P is a ground nh-program, we consider $bf_S(B_L)$ as the set of all distinct ground hybrid basic formulae that appear in P, denoted by $Formulae(P)$. If h is a partial or total hybrid formula function, then $dom(h) = posdom(h) \cup negdom(h)$ and $h = Pos(h) \cup Neg(h)$ viewing h as a set of annotated hybrid basic formulae.

Definition 9. *A set of annotated hybrid basic formulae h (a hybrid formula function) is said to* cover *a set of annotated hybrid basic formulae g (a hybrid formula function) if $dom(g) \subseteq dom(h)$ and $\forall\ F \in dom(g), g(F) \leq_t h(F)$.*

Definition 10. *A set of annotated hybrid basic formulae g (a hybrid formula function) is said to* agree *with a set of annotated hybrid basic formulae h (a hybrid formula function) if the following conditions hold:*

- $posdom(h) \subseteq posdom(g)$,
- $negdom(h) \subseteq negdom(g)$, *and*
- $\forall\ F \in dom(h), h(F) \leq_t g(F)$.

Definition 10 is closely related to the definition of the well-founded order defined in [22].

Definition 11 ([22]). *Let P be an nh-program, H_P be the set of all partial hybrid formula functions of P, and $h_1, h_2 \in H_P$. We define the following partial order ($\leq_w$) on H_P: $h_1 \leq_w h_2$ iff $posdom(h_1) \subseteq posdom(h_2)$, $negdom(h_1) \subseteq negdom(h_2)$, and $\forall\, F \in dom(h_1), h_1(F) \leq_t h_2(F)$.*

The notion of *cover* can also be applied to sets of hybrid basic formulae. If h and g are two hybrid formula functions, then $dom(h)$ covers $dom(g)$ if $dom(g) \subseteq dom(h)$. Moreover, a hybrid basic formula F is said to be covered by $dom(h)$ if $F \in dom(h)$.

Definition 12. *Let $P = \langle R, \tau \rangle$ be a ground nh-program and h be a total hybrid formula function, then we define the operator F_P as a mapping $F_P : HFF \rightarrow HFF$ where $F_P(h) = lfp(T_{P^h})$.*

Lemma 1. *Let $P = \langle R, \tau \rangle$ be a ground nh-program and h be a total hybrid formula function, then h is a stable p-model of P iff $F_P(h) = h$.*

Lemma 2. *The function F_P is anti-monotone with respect to $\leq_t$.*

Now, we describe an algorithm for computing the stable p-model semantics for an nh-program along with the auxiliary functions. In addition, we present the necessary conditions that these auxiliary functions have to satisfy to guarantee the soundness and completeness of stable p-model semantics computation algorithm. Figure 1 describes a decision procedure for determining whether an nh-program P has a stable p-model or not. The function $spmodels(P, h)$ computes one stable p-model for P, however, it can be modified to compute all the stable p-models of P. It returns true if there is a stable p-model for P agreeing with the set of annotated hybrid basic formulae h (a hybrid formula function), otherwise it returns false. It takes a ground nh-program P and a partial hybrid formula function (a set of annotated hybrid basic formulae) h as an input. The set h represents the partially computed stable p-model.

The function $spmodels(P, h)$ calls two functions: $pexpand(P, h)$ and $pconflict(P, h)$. The function $pexpand(P, h)$ expands the set of annotated hybrid basic formulae h by the functions $PAtleast(P, h)$ and $PAtmost(P, h)$, whereas the function $pconflict(P, h)$ discovers the conflicts. The function $pconflict(P, h)$ determines whether h is a hybrid formula function (partial or total) that could be extended to a stable p-model of P, by checking that each hybrid basic formula defined in h is assigned exactly one probability interval and h satisfies P. To guarantee the soundness and completeness of $spmodels(P, h)$ we present the conditions E1-E2 and C1-C2 required for designing $pexpand(P, h)$ and $pconflict(P, h)$. Let $h' = pexpand(P, h)$ we assume that:

E1: $posdom(h) \subseteq posdom(h'), negdom(h) \subseteq negdom(h')$, and for all $F \in dom(h)$, $h(F) \leq_t h'(F)$, and
E2: every stable p-model of P that agrees with h agrees also with h'.

In addition, we assume that $pconflict(P, h)$ satisfies the following conditions

C1: if $dom(h)$ covers $Formulae(P)$ and there is no stable p-model that agrees with h, then $pconflict(P, h)$ returns true, and
C2: if $pconflict(P, h)$ returns true, then there is no stable p-model of P that agrees with h.

The function $spmodels(P, h)$ starts by expanding the partially computed stable p-model h (line 2). Condition E1 ensures that h is really extended and E2 guarantees that no stable p-model is lost. Then a test for checking a conflict is performed. Condition C1 ensures that if $Formulae(P)$ is covered and there is a conflict, the conflict is detected (lines 3 and 4). Condition C2 guarantee that if there is a conflict, then there is no stable p-model agreeing with h'. If there is no conflict (lines 5 and 6) and $dom(h')$ covers $Formulae(P)$, then $spmodels(P, h)$ returns true with h' is a stable p-model of P. If there is $x \in Formulae(P)$

```
 1: function spmodels(P, h)
 2: h' := pexpand(P, h)
 3: if pconflict(P, h') then
 4:    return false
 5: else if dom(h') covers Formulae(P) then
 6:    return true
 7: else
 8:    take some x ∈ Formulae(P) not covered by dom(h')
 9:    if spmodels(P, h' ∪ {x : [0, 0]}) then
10:       return true
11:    else
12:       take x : [a, b] ∈ lfp(F_P^2) or gfp(F_P^2)
13:       return spmodels(P, h' ∪ {x : [a, b]})
14:    end if
15: end if
```

```
 1: function pexpand(P, h)
 2: repeat
 3:    h' := h
 4:    h := PAtleast(P, h)
 5:    h := h ∪ {F : [0, 0] | F ∈ Formulae(P) and F : [0, 0] ∈ PAtmost(P, h)}
 6: until h' = h
 7: return h
```

```
 1: function pconflict(P, h)
 2: { Precondition: h = expand(P, h)}
 3: if posdom(h) ∩ negdom(h) ≠ ∅ then
 4:    return true
 5: else if dom(h) covers Formulae(P) and h does not satisfy P then
 6:    return true
 7: else
 8:    return false
 9: end if
```

Fig. 1. A decision procedure for the stable p-model semantics

not covered by $dom(h')$ (line 8), then either $x : [0,0]$ belongs to the partially computed stable p-model (line 9) or there is some constant annotation $[a,b]$ such that $x : [a,b]$, with $[a,b] \neq [0,0]$, belongs to the partially computed stable p-model (line 12 and 13). The two cases are handled by backtracking. In the first case we extend h' by $\{x : [0,0]\}$, but if $spmodels(P, h' \cup \{x : [0,0]\})$ returns false, then $x : [0,0]$ does not belong to the computed stable p-model. Hence, $spmodels(P,h)$ returns what $spmodels(P, h' \cup \{x : [a,b]\})$ returns, where $x : [a,b] \in lfp(F_P^2)$ *or* $gfp(F_P^2)$. Since F_P is antimonotone, F_P^2 is monotone and its least fixpoint and greatest fixpoint limit the fixpoints of F_P [12]. Therefore, because of the possibility of having multiple nh-rules in P with x in their heads with different annotations, we select $[a,b]$ that is guaranteed to be in the computed stable p-model. This is achieved by selecting $x : [a,b]$ such that $x : [a,b] \in lfp(F_P^2)$ or $x : [a,b] \in gfp(F_P^2)$. This is because any stable p-model is a fixpoint of the operator F_P. The following theorem proves the correctness of decision procedure described in Figure 1.

Theorem 2. *Let P be an nh-program and h be a hybrid formula function. Then, there is a stable p-model of P agreeing with h if and only if $spmodels(P,h)$ returns true.*

Proof. The proof of this theorem is similar to the proof of a corresponding result presented in [17]. The proof proceeds as follows. Let $NC(P,h) = Formulae(P) \backslash dom(h)$ be the set of hybrid basic formulae that is in $Formulae(P)$ but not covered by $dom(h)$. We prove the theorem by induction on $NC(P,h)$. Assume that $NC(P,h) = \emptyset$ which implies that $dom(h)$ covers $Formulae(P)$. Then, $h' = pexpand(P,h)$ and by E1 $dom(h')$ covers $Formulae(P)$ as well and $spmodels(P,h)$ returns true if and only if $pconflict(P,h')$ returns false. By E2, C1, and C2 $pconflict(P,h')$ returns false exactly when there is a stable p-model agreeing with h.

Assume that $NC(P,h) \neq \emptyset$. If $pconflict(P,h')$ returns true, then $spmodels(P,h)$ returns false. Hence, there is no stable p-model agreeing with h by the conditions E2 and C2. However, if $pconflict(P,h')$ returns false and $dom(h')$ covers $Formulae(P)$, then $spmodels(P,h)$ returns true. Therefore, there is a stable p-model that agrees with h due to the conditions E2 and C1. Otherwise, since $spmodels(P,h)$ returns true and $dom(h')$ still not covers $Formulae(P)$ and since the size of $NC(P, h' \cup \{x.[0,0]\}) = NC(P, h' \cup \{x.[a,b]\}) \subset NC(P,h')$ then by inductive hypothesis together with E1 and E2 we have that that either $spmodels(P, h' \cup \{x.[0,0]\})$ or $spmodels(P, h' \cup \{x.[a,b]\})$ returns true if and only if there is a stable p-model agreeing with h. ■

5 PAtleast(P,h) and PAtmost(P,h)

In this subsection we provide foundations for computing the functions $PAtleast(P,h)$ and $PAtmost(P,h)$. The function $PAtleast(P,h)$ enlarges the partially computed stable p-model h by adding annotated hybrid basic formulae and/or monotonically increasing the annotations associated to the hybrid basic

formulae that already exist in the partially computed stable p-model h. The function $PAtleast(P, h)$ computes the least fixpoint of the operator D_P, which is a variation of probabilistic well-founded operator W_P defined in [22]. We say that an nh-program globally satisfies $F : \nu$ (*not* $(G : \beta)$) if the nh-program as a whole provides evidence for satisfying $F : \nu$ (*not* $(G : \beta)$).

Definition 13 (Global Satisfaction). *Let P be an nh-program and $F : \nu$ (not $(G : \beta)$) be a positive (negative) hybrid literal. We say that $F : \nu$ (not $(G : \beta)$) is* globally satisfied by *P if every minimal probabilistic interpretation that satisfies P also satisfies $F : \nu$ (not $(G : \beta)$).*

Let $P = \langle R, \tau \rangle$ be an nh-program and g be a stable p-model of P agreeing with the set of annotated hybrid basic formulae h and H_P is the set of all partial p-interpretations of P. Then we define $PAtleast(P, h)$ to be the least fixpoint of the operator $D_P : H_P \rightarrow H_P$ defined as follows:

1. For each atom A we have that $D_P(h)(A) = c_{\tau(A)}\ M_A$, where $M_A \neq \emptyset$ contains the probability intervals μ obtained from the nh-rules $A : \mu \leftarrow Body \in R$, such that h satisfies $Body$, and for each negative hybrid literal *not* $(G_j : \beta_j)$ in $Body$ we have that P globally satisfies *not* $(G_j : \beta_j)$.
2. For each atom A we have that $D_P(h)(A) = [0, 0]$ if for each nh-rule $r \in R$ such that A appears in its head, h does not satisfy some hybrid literal $F : \nu$ or *not* $(G : \beta)$ in the body of r and P does not globally satisfy $F : \nu$.
3. $D_P(h)(G_1 \wedge_\rho G_2) = c_\rho(D_P(h)(G_1), D_P(h)(G_2))$ where $(G_1 \wedge_\rho G_2) \in bf_S(B_L)$ and for each atom A in $(G_1 \wedge_\rho G_2)$, A is defined in $D_P(h)$.
4. $D_P(h)(G_1 \vee_{\rho'} G_2) = c_{\rho'}(D_P(h)(G_1), D_P(h)(G_2))$ where $(G_1 \vee_{\rho'} G_2) \in bf_S(B_L)$ and for each atom A in $(G_1 \vee_{\rho'} G_2)$, A is defined in $D_P(h)$.

Example 4. Consider the following nh-program P

$$\begin{array}{ll} a : [0.9, 1] & \leftarrow b : [0.7, 0.8], \mathit{not}\ (c : [0.5, 0.55]) \\ d : [0.9, 1] & \leftarrow \mathit{not}\ (a : [0.9, 1]) \\ e : [0.2, 0.35] & \leftarrow \mathit{not}\ (b : [0.7, 0.8]) \end{array}$$

We will compute $h = PAtleast(P, \emptyset)$. Since b and c do not appear in heads of any nh-rules in P, $b : [0, 0] \in h$ and $c : [0, 0] \in h$ by 2 in the above definition. In addition, $a : [0, 0] \in h$ by 2 as well since the first nh-rule is not satisfied due to $b : [0.7, 0.8]$ in the nh-rule because $[0.7, 0.8] \not\leq_t [0, 0]$. Obviously, $d : [0.9, 1]$ and $e : [0.2, 0.35]$ are in h by 1 in the above definition. Hence, $PAtleast(P, \emptyset) = \{a : [0, 0], b : [0, 0] c : [0, 0], d : [0.9, 1], e : [0.2, 0.35]\}$ which is the least fixpoint of D_P.

Lemma 3. *The function $PAtleast(P, h)$ is monotonic with respect to $\leq_w$ in its second argument.*

Note that $D_P(h)$ is a variation of the probabilistic well-founded operator W_P defined in [22]. This implies that $PAtleast(P, h) = lfp(D_P(h)) = lfp(W_P)$. Therefore, according to Theorem 4 of [22], g is a stable p-model of P if g is a fixpoint of W_P, and hence a fixpoint of D_P which in turn a fixpoint of $PAtleast(P, h)$. This implies that g is a stable p-model of P iff $g = W_P(g) = D_P(g) = PAtleast(P, g)$.

Proposition 4. *If g is a stable p-model of an nh-program P that agrees with the partial hybrid formula function h, then g agrees with $PAtleast(P, h)$.*

Furthermore, we can bound a stable p-model from above by defining the function $PAtmost(P, h)$. The function $PAtmost(P, h)$ computes the least fixpoint of P^h, the probabilistic reduct of P with respect to h (defined below). The idea is to extend the set of annotated hybrid basic formulae h which corresponds to the partially computed stable p-model by adding annotated hybrid basic formulae of the form $x : [0, 0]$ without violating condition E2. We can add $x : [0, 0]$ to the set h if $x : [0, 0] \in PAtmost(P, h) = lfp(T_{P^h})$. However, a different notion of probabilistic reduct from the one defined in Definition 7 is needed in this context as defined below.

Definition 14. *Let $P = \langle R, \tau \rangle$ be a ground nh-program, h be a partial hybrid formula function, and*

$$r \equiv A : \mu \leftarrow F_1 : \mu_1, \ldots, F_n : \mu_n, not\ (G_1 : \beta_1), \ldots, not\ (G_m : \beta_m) \in R.$$

Then the probabilistic reduct P^h of P with respect to h is $P^h = \langle R^h, \tau \rangle$ where R^h is the set of h-rules obtained from R by:

- *deleting every nh-rule r in R where there is a not $(G_j : \beta_j)$ in the body of r such that $\beta_j \leq_t h(G_j)$,*
- *deleting every not $(G_k : \beta_k)$ from the body of the remaining nh-rules.*

The notion of reduct in the above definition is a generalization of the notion of reduct in Definition 7, to cope with partial hybrid formula functions. For total hybrid formula functions both notions of reduct coincides. In addition, $PAtmost(P, h)$ is a total hybrid formula function. Consequently, if g is a stable p-model of an nh-program P, then $g = PAtmost(P, g)$. It is worth noting that, from the definition of the probabilistic reduct with respect to partial hybrid formula function h, h can be extended to a total hybrid formula function and we still get the same probabilistic reduct. This is achieved by adding to h the annotated hybrid basic formulae $F : [0, 0]$ such that $F \in bf_S(B_L) \setminus dom(h)$. This means, if $h_1 \leq_w h_2$, then $h_1 \leq_t h_2$ as well, after extending both h_1 and h_2 to total hybrid formula functions by adding $F : [0, 0]$ such that $F \in bf_S(B_L) \setminus dom(h_1)$ to h_1 and $F : [0, 0]$ such that $F \in bf_S(B_L) \setminus dom(h_2)$ to h_2 respectively.

Lemma 4. *The function $PAtmost(P, h)$ is anti-monotone in its second argument.*

The above lemma shows that the function $PAtmost(P, h)$ is anti-monotone with respect to $\leq_t$. This is because given $h_1 \leq_w h_2$, then we also get $h_1 \leq_t h_2$, which implies that $PAtmost(P, h_2) \leq_t PAtmost(P, h_1)$.

Proposition 5. *Let g be a stable p-model of P that agrees with h. Then $g \leq_t PAtmost(P, h)$.*

Corollary 1. *The function $pexpand(P, h)$ satisfies conditions E1 and E2*

Corollary 2. *The function* $conflict(P, h)$ *satisfies conditions C2*

It follows that $pexpand(P, h)$ satisfies conditions E1 and E2. The function $conflict(P, h)$ obviously fulfills C2, and the next proposition shows that C1 also holds.

Proposition 6. *If* $h = pexpand(P, h)$, $dom(h)$ *covers* $Formulae(P)$, *and* $posdom(h) \cap negdom(h) = \emptyset$ *and* h *satisfies* P, *then* h *is a stable p-model of* P.

Example 5. Consider the following nh-program P

$$\begin{array}{ll} a : [0.45, 0.55] & \leftarrow c : [1, 1], not\ (b : [0.7, 0.95]) \\ b : [0.7, 0.95] & \leftarrow c : [1, 1], not\ (a : [0.45, 0.55]) \\ c : [1, 1] & \leftarrow not\ (a : [0.45, 0.55]) \end{array}$$

We use the decision procedure *spmodels* to determine whether P has a stable p-model or not and return it if exist. The $lfp(F_P^2)$ is the empty set and $gfp(F_P^2)$ is

$$\{a : [0.45, 0.55], b : [0.7.0.95], c : [1, 1]\}.$$

First$pexpand(P, \emptyset)$ returns $\emptyset$ and $pconflict(P, \emptyset)$ returns false. Since $Formulae(P) = \{a, b, c\}$ is not covered by $\emptyset$, we choose either a, b, or c in order to proceed. Let us take b, then $spmodels(P, \{b : [0, 0]\})$ is executed. $pexpand(P, \{b : [0, 0]\})$ returns $\{b : [0, 0]\}$. Then $pconflict(P, \{b : [0, 0]\})$ returns false. Since $Formulae(P)$ is not covered by $\{b\}$, we choose either a or c in order to proceed. Let us take a, then

$$spmodels(P, \{a : [0, 0], b : [0, 0]\})$$

is executed. $pexpand(P, \{a : [0, 0], b : [0, 0]\})$ returns

$$\{a : [0, 0], b : [0, 0], c : [1, 1], b : [0.7, 0.95], a : [0.45, 0.55]\}.$$

Then $pconflict(P, \{a : [0, 0], b : [0, 0], c : [1, 1], b : [0.7, 0.95], a : [0.45, 0.55]\})$ returns true. Then we backtrack and execute $spmodels(P, \{a : [0.45, 0.55], b : [0, 0]\})$.

$pexpand(P, \{a : [0.45, 0.55], b : [0, 0]\})$ *returns* $\{a : [0.45, 0.55], b : [0, 0], a : [0, 0], c.[0, 0]\}$.

Then $pconflict(P, \{a : [0.45, 0.55], b : [0, 0], a : [0, 0], c.[0, 0]\})$ returns true. Finally, we backtrack and execute $spmodels(P, \{b : [0.7, 0.95]\})$. $pexpand(P, \{b : [0.7, 0.95]\})$ returns $\{b : [0.7, 0.95]\}$. Then $pconflict(P, \{b : [0.7, 0.95]\})$ returns false. Since $Formulae(P)$ is not covered by $\{b\}$, we choose either a or c in order to proceed. Let us take a, then $spmodels(P, \{a : [0, 0], b : [0.7, 0.95]\})$ is executed. $pexpand(P, \{a : [0, 0], b : [0.7, 0.95]\})$ returns $\{a : [0, 0], b : [0.7, 0.95], c : [1, 1]\}$. Then

$$pconflict(P, \{a : [0, 0], b : [0.7, 0.95], c : [1, 1]\})$$

returns false and $spmodels(P, \{a : [0, 0], b : [0.7, 0.95]\})$ returns true as well as $spmodels(P, \{b : [0.7, 0.95]\})$ and $spmodels(P, \emptyset)$ having $\{a : [0, 0], b : [0.7, 0.95], c : [1, 1]\}$ as a stable p-model of P.

6 Conclusions

In this work, we have proposed an algorithm for computing the stable probabilistic model semantics [22]. The proposed algorithm is a modification of the decision procedure of SMODELS [17], a state-of-the-art system for computing the stable model semantics of normal logic programs. We have described the modified algorithm, along with its auxiliary functions, and we have provided the necessary conditions that these auxiliary functions have to satisfy to guarantee the soundness and completeness of the proposed algorithm. We have presented formal definitions and algorithms for these auxiliary functions and shown that they satisfy the necessary conditions for the soundness and completeness of the proposed algorithm.

As future work, we plan to provide an implementation of these algorithms, and investigate applications of the resulting framework in the context of knowledge representation and reasoning in presence of uncertainty (e.g., probabilistic planning).

References

1. C. Baral et al. Probabilistic reasoning with answer sets. *In 7th International Conference on Logic Programming and Nonmonotonic Reasoning*, Springer Verlag, 2004.
2. C. Bell, A. Nerode, R. Ng, V. S. Subrahmanian. Mixed integer programming methods for computing Nonmonotonic Deductive Databases. *Journal of ACM*, 41(6): 1178-1215, 1994.
3. W. D. Chen, D. S. Warren. Computation of stable models and its integration with logical query processing. *IEEE Transaction on Knowledge and Data Engineering*, 8(5): 742-757, 1996.
4. P. Cholewinski, V. Marek, M. Truszczynski, A. Mikitiuk. Computing with default logic. *Artificial Intelligence*, 112(1-2): 105-146, 1999.
5. A. Dekhtyar and V. S. Subrahmanian. Hybrid probabilistic program. *Journal of Logic Programming*, 43(3): 187-250, 2000.
6. M. Dekhtyar, A. Dekhtyar, and V. S. Subrahmanian. Hybrid probabilistic programs: algorithms and complexity. *In Uncertainty in Artificial Intelligence Conference*, pages 160-169, 1999.
7. W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 1(3): 267-284, 1984.
8. A. Van. Gelder. The alternating fixpoint of logic programs with negation. *Journal of Computer and System Sciences*, 47(1):185-221, 1993.
9. A. Van. Gelder, K. A. Ross, and J. S. Schlipf. The Well-founded semantics for general logic programs. *Journal of ACM*, 38(3):620-650, 1991.
10. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. Kowalski and K. Bowen, editors, *In Fifth International Conference and Symposium on Logic Programming*, 1070-1080, 1988.
11. L. V.S. Lakshmanan and F. Sadri. On a theory of probabilistic deductive databases. *Journal of Theory and Practice of Logic Programming*, 1(1):5-42, January 2001.
12. V. Lifschitz. Foundations of logic programming. *In Principles of Knowledge Representation*, 69-127, CSLI Publications, 1996.

13. J. J. Lu and S. M. Leach. Computing annotated logic programs. *In International Conference on Logic Programming*, Pascal van Hentenryck, editor, MIT press, Cambridge, MA, 1994.
14. W. Marek and M. Truszczynski. Autoepistemic logic. *Journal of ACM*, 38(3):588–619, 1991.
15. R. T. Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150-201, December 1992.
16. R. T. Ng and V. S. Subrahmanian. Stable semantics for probabilistic deductive databases. *Information and Computation*, 110(1):42-83, 1994.
17. I. Niemela and P. Simons. Efficient implementation of the well-founded and stable model semantics. *In Joint International Conference and Symposium on Logic Programming*, 289-303, 1996.
18. I. Niemela, P. Simons, T. Soininen. Stable model semantics of weight constraint rules. *In Fifth International Conference on Logic Programming and Nonmonotonic Reasoning*, 317-331, 1999.
19. R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81-132, 1980.
20. E. Saad. *Hybrid probabilistic programs with non-monotonic negation: semantics and algorithms*. Ph.D. thesis, New Mexico State University, May 2005.
21. E. Saad and E. Pontelli. Towards a more practical hybrid probabilistic logic programming framework. In *Practical Aspects of Declarative Languages*. Springer Verlag, 2005.
22. E. Saad and E. Pontelli. Hybrid probabilistic logic programming with non-monotoic negation. In *Twenty First International Conference on Logic Programming*. Springer Verlag, 2005.
23. V. S. Subrahmanian, D. S. Nau, C. Vago. wfs + branch and bound = stable models. *IEEE Transaction on Knowledge and Data Engineering*, 7(3): 362-377, 1995.
24. J. Vennekens, S. Verbaeten, and M. Bruynooghe. Logic programs with annotated disjunctions. *In International Workshop on Nonmonotonic Reasoning*, 2004.