Stable Semantics for Probabilistic Deductive Databases

RAYMOND NG* and V. S. SUBRAHMANIAN†,

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

In this paper we study the semantics of non-monotonic negation in probabilistic deductive databases. Based on the stable semantics for classical logic programming, we examine three natural notions of stability: stable formula functions, stable families of probabilistic interpretations, and stable probabilistic models. We show that stable formula functions are minimal fixpoints of operators associated with probabilistic logic programs. We also prove that each member in a stable family of probabilistic interpretations is a probabilistic model of the program. Then we show that stable formula functions and stable families behave as duals of each other, tying together elegantly the fixpoint and model theories for probabilistic logic programs with negation. Furthermore, since a probabilistic logic program may not necessarily have a stable family of probabilistic interpretations, we provide a stable class semantics for such programs. Finally, we investigate the notion of stable probabilistic model. We show that this notion, though natural, is too weak in the probabilistic framework. © 1994 Academic Press, Inc.

1. INTRODUCTION

Many frameworks for multivalued logic programming have been proposed to handle uncertain information, such as the ones described in (3, 6, 7, 12, 13, 17–19, 23, 31). However, all these approaches are non-probabilistic in nature, as the way they interpret conjunctions and disjunctions (such as by taking minimum and maximum) is too restrictive for probabilistic data. Since probability theory is a well developed formalism for dealing with uncertainty, we believe that a probabilistic approach to quantitative deduction in logic programming is important. In (24, 25) we have proposed a framework for probabilistic deductive databases, i.e., logic programs without function symbols. We have shown that this framework is expressive, as it supports conditional probabilities, classical negation, propagation of probabilities, and Bayesian updates.

* Electronic mail: rng@cs.umd.edu.
† Electronic mail: vs@cs.umd.edu.
‡ To whom correspondence should be addressed.

0890-5401/94 $6.00
Copyright © 1994 by Academic Press, Inc.
All rights of reproduction in any form reserved.
However, one fundamental issue that remains unaddressed in the framework we proposed in (24, 25) is the representation and manipulation of non-monotonic modes of negation. In particular, the framework is incapable of default reasoning and drawing negative conclusions based on the absence of positive information. Thus, our focus in this paper is to study non-monotonic negation in probabilistic logic programming. The approach we take is based on the stable semantics of (classical) logic programming with negation (14).

In this paper, we introduce and investigate three natural notions of stable semantics for probabilistic logic programming. Though the notions are all based on different semantical structures of probabilistic logic programming, we show that two of the notions exhibit a one–one equivalence, thus tying together the fixpoint and model theories together. Finally, the third notion coincides with probabilistic model theory when the negations are interpreted classically, rather than non-monotonically.

This paper is organized as follows: Sections 2 and 3 present the syntax and semantics of negation-free probabilistic logic programs respectively. Section 4 presents the notion of stable formula functions and relates stable formula functions to fixpoints of operators associated with programs with negation. Section 5 introduces the notion of a stable family of probabilistic interpretations and ultimately shows that the notions of stable formula functions and stable families are essentially duals. As programs may not have stable formula functions, Section 6 extends the two notions of stability to provide semantics for such programs. Section 7 examines the notion of stable probabilistic models. It eventually shows that this notion is weaker than stable formula functions and stable families. Section 8 presents examples of default reasoning within our framework.

2. SYNTAX OF GENERAL PROBABILISTIC LOGIC PROGRAMS

In this section we present the syntax and several examples of probabilistic logic programs with negation. Let $L$ be a language generated by finitely many constant and predicate symbols and an infinite supply of ordinary variable symbols. For reasons explained later, $L$ does not contain function symbols. Thus, the concepts of terms, atoms, and literals are the usual ones (21). We denote the Herbrand Base of $L$ by $B_L$.

To manipulate probabilities, we enrich $L$ by adding certain new variable symbols called annotation variables and certain new pre-interpreted and computable function symbols called annotation functions. These new variables and function symbols can be used to annotate atoms and literals
of $L$. Thus, if $toss$ is a unary predicate in our language, $toss(head)$ can be annotated with $[0.49, 0.51]$, and written as

$$toss(head) : [0.49, 0.51]$$

to denote that the probability that a tossing a coin yields heads is in the interval $[0.49, 0.51]$. We continue to refer to the enriched language as $L$. We now proceed to define exactly what is allowed to appear in an annotation.

**Definition 1.** An annotation function $f$ of arity $n$ is a total function of type $([0, 1])^n \rightarrow [0, 1]$.

We further assume that all annotation functions $f$ are computable in the sense that there is a fixed procedure $P_f$ such that if $f$ is $n$-ary, and if $\rho_1, \ldots, \rho_n$, all in $[0,1]$, are given as inputs to $P_f$, then $f(\rho_1, \ldots, \rho_n)$ is computed by $P_f$ in a finite amount of time.

In addition to the interpreted annotation function symbols, $L$ also contains infinitely many variable symbols. Furthermore, we assume that this set of variable symbols are partitioned into two infinite subsets. The first subset consists of normal variable symbols in first order logic; they can only appear in atoms. We refer to these variables as object variables. The other subset consists of annotation variable symbols. Annotation variables can only range between 0 and 1. Annotation variable symbols can only appear in annotation terms, a concept defined as follows.

**Definition 2.**

1. $\rho$ is called an annotation item if it is one of the following:
   
   (i) a constant in $[0, 1]$, or
   
   (ii) an annotation variable in $L$, or
   
   (iii) of the form $f(\delta_1, \ldots, \delta_n)$, where $f$ is an annotation function of arity $n$ and $\delta_1, \ldots, \delta_n$ are annotation items.

2. For real numbers $c, d$ such that $0 \leq c, d \leq 1$, let the closed interval $[c, d]$ be the set $\{x | c \leq x \leq d\}$.

3. $[\rho_1, \rho_2]$ is called an annotation (term) if $\rho_i$ ($i = 1, 2$) is an annotation item.

If an annotation does not contain any annotation variables, the annotation is called a $c$-annotation. Otherwise, it is called a $v$-annotation.

**Example 1.** $[0, 5, 0.7]$ and $[0.7, 0.3]$ (3 $\emptyset$) are $c$-annotations, while $[0, V_1]$, $[0.5 * V_1, 0.7]$, and $[V_1, V_2]$ are all $v$-annotations, where $V_1$ and $V_2$ are annotation variables, and $(0.5 * V_1)$ is an annotation item written in infix notation.
In the rest of this paper, we refer to an annotation term simply as an annotation. We also adopt the convention of using $\mu$'s to represent annotations in general and $\alpha$'s and $\beta$'s to denote e-annotations. In situations when we need to refer to the upper and lower bounds of annotations (i.e., annotation items), we use lower case Greek letters such as $\rho$ and $\delta$. If the upper or lower bounds are constants in $[0, 1]$, we use $e$'s and $d$'s instead. In addition, we use $V$'s to represent annotation variables.

**Definition 3.** (1) A basic formula, not necessarily ground, is either a conjunction of atoms or a disjunction of atoms. Note that disjunction and conjunction cannot occur simultaneously in one basic formula.

(2) Let $\text{conj}(B_L)$ denote the set of all conjunctions of ground atoms in $B_L$, i.e., $\text{conj}(B_L) = \{A_1 \land \cdots \land A_n \mid n \geq 1 \text{ is an integer and } A_1, \ldots, A_n \in B_L \text{ and } \forall 1 \leq i, j \leq n, i \neq j \Rightarrow A_i \neq A_j\}.$

(3) Similarly, let $\text{disj}(B_L)$ denote the set of all disjunctions of ground atoms in $B_L$; i.e., $\text{disj}(B_L) = \{A_1 \lor \cdots \lor A_n \mid n \geq 1 \text{ is an integer and } A_1, \ldots, A_n \in B_L \text{ and } \forall 1 \leq i, j \leq n, i \neq j \Rightarrow A_i \neq A_j\}.$

We use $bf(B_L)$ to denote $\text{conj}(B_L) \cup \text{disj}(B_L)$; i.e., $bf(B_L)$ is the set of all ground basic formulas in our language.

Since $L$ does not contain any function symbols, $bf(B_L)$, $\text{conj}(B_L)$, and $\text{disj}(B_L)$ are all finite.

**Definition 4.** If $C$ is a conjunction of atoms, not necessarily ground, and $\mu$ is an annotation, then $C : \mu$ is called an annotated conjunction. Similarly, if $D$ is a disjunction, not necessarily ground, then $D : \mu$ is called an annotated disjunction.

**Definition 5.** (1) Let $F_0, \ldots, F_n, G_1, \ldots, G_m$ be basic formulas. Also let $\mu_0, \ldots, \mu_{n+m}$ be annotations such that every annotation variable occurring in $\mu_0$, if any, also appears in one of $\mu_1, \ldots, \mu_{n+m}$. Then the clause

$$F_0 : \mu_0 \leftarrow F_1 : \mu_1 \land \cdots \land F_n : \mu_n \land \neg (G_1 : \mu_{n+1}) \land \cdots \land \neg (G_m : \mu_{n+m})$$

is called a general probabilistic clause (gp-clause for short). $F_0 : \mu_0$ is called the head of this clause, and $(F_1 : \mu_1 \land \cdots \land F_n : \mu_n \land \neg (G_1 : \mu_{n+1}) \land \cdots \land \neg (G_m : \mu_{n+m}))$ is called its body.

(2) A pff-clause is a gp-clause without negated annotated basic formulas (25), i.e., when $m = 0$.

**Definition 6.** (1) A general probabilistic (gp-) program is a finite set of gp-clauses.
(2) A pf-program is a finite set of pf-clauses (25).

If the annotation $\mu$ is a c-annotation $[c_1, c_2]$, the annotated basic formula $F : \mu$ intuitively means: "The probability of $F$ must lie in the interval $[c_1, c_2]". Suppose $\mu_0, \ldots, \mu_n$ are annotations. Then the clause

$$ F_0 : \mu_0 \leftarrow F_1 : \mu_1 \land \cdots \land F_n : \mu_n $$

is read as follows: "If the probability of $F_1$ is in the interval $\mu_1$ and \cdots the probability of $F_n$ is in the interval $\mu_n$, then the probability of $F_0$ is in the interval $\mu_0." Similarly, the negation of the annotated formula, $\neg(G : \mu)$, is to be read as follows: "It is not provable that the probability of $G$ must lie in the interval $\mu." Hence, the negation $\neg$ considered here is non-monotonic. In (25) we demonstrate the ability of our framework to express conditional probabilities, propagation of probabilities, Bayesian updates, and quantitative rule processing in van Emden's framework (31). Here we concentrate on the role of negation. Consider the following examples.

**Example 2.** Suppose we intend to say that the conditional probability of $A$ given $B$ is $p$. This is equivalent to saying that: $\text{Prob}(A \land B) = p \cdot \text{Prob}(B)$. This relation can be expressed by the clause

$$ (A \land B) : [p \cdot V, p \cdot V] \leftarrow B : [V, V]. $$

Similarly, if we wish to state the conditional probability of $A$ given $B$, we can denote this by a new proposition, called $AB$, say, and use the clause

$$ (AB) : [V_2/V_1, V_2/V_1] \leftarrow (A \land B) : [V_2, V_2] \land B : [V_1, V_1], $$

assuming that the probability of $B$ is non-zero. Note that the annotation variables (i.e, $V, V_1, V_2$) are required to express these conditional probability statements within our framework.

**Example 3.** Suppose we believe that if it is not provable that the probability of a coin $C$ showing heads is within the range $[0.49, 0.51]$, then there is over 95% chance that the coin is unfair. The gp-clause

$$ \text{unfair}(C) : [0.95, 1] \leftarrow \neg(\text{head}(C) : [0.49, 0.51]) $$

represents our belief.

**Example 4.** Suppose we know that there is over 95% chance that a dog can bark, unless the dog is abnormal. We also know that Benjy and Fido are dogs. However, Benjy is unable to bark (his vocal cords were injured at some point). This can be represented as
\[ \text{bark}(X) : [0.95, 1] \leftarrow \text{dog}(X) : [1, 1] \land \neg (\text{abnormal}(X) : [1, 1]) \]

\[ \text{dog}(\text{fido}) : [1, 1] \leftarrow \]

\[ \text{dog}(\text{benjy}) : [1, 1] \leftarrow \]

\[ \text{bark}(\text{benjy}) : [0, 0] \leftarrow \]

\[ \text{abnormal}(X) : [1, 1] \leftarrow \text{bark}(X) : [0, 0]. \]

The last clause says that a dog is certainly abnormal if it definitely cannot bark. As we shall see later on (cf. Example 10), we can deduce from these clauses the fact that Fido can bark, but Benjy cannot.

**Example 5.** Consider a computer system in which resources are shared by many users. In particular, a server is shared by two users—a superuser \(a\) and a normal user \(b\). Suppose that a statistical survey of this system discovers that the probability of approval of a request by \(a\) to use the server is over 0.9 when certain conditions are met; these conditions may be probabilistic as well. For a similar request from the normal user \(b\), the following restrictions apply. If there is an over 95\% chance that \(b\) will use up his computer budget, the request is simply not honoured. Otherwise, the probability of approval of the request is conditional upon \(b\)'s chances of obtaining all the other resources he needs. Suppose this conditional probability is found to be 0.8. Furthermore, due to the mutual exclusion requirement of the server, at most one user can be handled by the server at any point in time. The situation can be represented by the following gp-clauses:

\[ \text{getServer}(a) : [0.9, 1] \leftarrow \ldots (\text{some conditions}) \quad (1) \]

\[ (\text{getServer}(b) \land \text{getOthers}(b)) : [0.8 \ast V, 0.8 \ast V] \leftarrow \text{getOthers}(b) : [V, V] \land \neg (\text{overBudget}(b) : [0.95, 1]) \quad (2) \]

\[ \text{getServer}(b) : [0, 0] \leftarrow \text{overBudget}(b) : [0.95, 1] \quad (3) \]

\[ (\text{getServer}(a) \land \text{getServer}(b)) : [0, 0] \leftarrow . \quad (4) \]

Clause (1) says that whenever \(a\) meets certain conditions, \(a\)'s chance of getting the server is between 0.9 and 1. Clause (2) represents the situation where it is not provable that there is an over 95\% chance that \(b\) will use up his budget.\(^1\) It states that if \(b\)'s probability of obtaining the other

\(^1\) If we consider a more general situation in which there are many normal users, the \(b\) in the clause can be replaced by a variable \(X\). But to simplify our presentation, we consider only one normal user \(b\).
resources is $V$, then the combined probability for $b$ to obtain the server and
the other resources is $0.8 \cdot V$, where $V$ is an annotation variable that
ranges from 0 to 1. This clause maintains the conditional probability
relationship stated above because of the identity in probability theory
$\text{Prob}(A \land B) = \text{Prob}(A \mid B) \cdot \text{Prob}(B)$, for any events $A$ and $B$, where
$\text{Prob}(A \mid B)$ denotes the conditional probability of $A$ given $B$. In (25) we
show that this is a valid way of representing conditional probabilities.
Clause (3) considers the case when there is an over 95\% chance that $b$ will
use up his budget. The last clause stresses that it is impossible for $a$ and $b$
to get the server simultaneously. With these clauses, the behaviour of this
system can now be studied.

In the context of probabilistic deduction with annotated basic formulas,
there are two major kinds of negation. The first is classical negation. Given
that the probability of a (basic) formula $F$ is within the range $[\rho, \delta]$, it is
easy to verify that the probability of $\overline{F}$, the classical negation of $F$, must lie
in the range $[1 - \delta, 1 - \rho]$. Hence, pf-clauses (cf. part 2 of Definition 5)
are expressive enough for representing classical negation. For instance,
in Example 3, $\text{unfair}(C) : [0.95, 1]$ may very well be represented as
$\text{fair}(C) : [0, 0.05]$.

The second kind of negation is the one considered here. Again,
$\neg(F : [\rho, \delta])$ is to be read as: “It is not provable that the probability of
$F$ must lie in the interval $[\rho, \delta]$.” For instance, in Example 5 above, if the
probability of $\text{overBudget}(b)$ lies in the range $[0, 0.98]$, then it is not
provably that the probability of $\text{overBudget}(b)$ must lie in $[0.95, 1]$. Thus,
in order to conclude $\neg(F : [\rho, \delta])$, one need only demonstrate that
$(F : [\rho, \delta])$ is not provable. It is important to note that showing this is not
the same as showing that $F$'s probability lies in some other interval. In fact,
it might be possible (e.g., true in some, but not all models) that $F$'s
probability lies in the range $[\rho, \delta]$ without it being provable that $F$'s
probability lies in the range $[\rho, \delta]$. In this case, we could non-monotoni-
cally conclude $\neg(F : [\rho, \delta])$.

3. BACKGROUND: FIXPOINT AND PROBABILISTIC MODEL THEORIES
FOR pf-PROGRAMS

In this section we informally summarize the essential notions and results
of the fixpoint and model theories we developed for probabilistic logic
programs without negation (cf. (24, 25)).

3.1 Probabilistic Interpretations and Formula Functions

A world $w$ is any set of ground atoms, i.e. $w \subseteq B_L$. We define three
semantical structures for probabilistic logic programming.
The first semantical structure we define is a world probability density function.

**Definition 7.** A world probability density function $WP: 2^{B_l} \rightarrow [0, 1]$ assigns to each world $W_j \in 2^{B_l}$ a probability $WP(W_j)$ such that for all $W_j \in 2^{B_l}$, $WP(W_j) \geq 0$ and $\sum_{W_j \in 2^{B_l}} WP(W_j) = 1$.

To simplify our notation, hereafter we use $p_j$ to denote $WP(W_j)$ for $W_j \in 2^{B_l}$.

In the context of probabilistic deduction, we assume that the "real" world is definite; i.e., there are some propositions that are true, and some that are false. However, we are uncertain about which of the various "possible worlds" is the right one. Thus, we use a world probability density function to define probability densities (20) on the set of all possible worlds. In other words, a world probability density function assigns a probability (i.e., a non-negative number) to each world such that the sum of all probabilities is 1. Our notions of worlds and world probability density functions are similar in essence to the "possible worlds" approach suggested by Hailperin (15) and Fagin et al. [10].

Every world probability density function $WP$ provides a way of assigning point probabilities to formulas in $hf(B_l)$. Suppose $F$ is one such formula. Let $Models(F)$ be the set of worlds in which $F$ is true. The probability of $F$ according to $WP$ is simply obtained by summing up the probabilities of the worlds in $Models(F)$; i.e., the probability of $F$ is simply $\sum_{w \in Models(F)} WP(w)$.

**Definition 8.** The function $I$ associated with $WP$ that assigns the value

$$I(F) = \sum_{w \in Models(F)} WP(w)$$

to the basic formula $F$ is called a probabilistic interpretation.

Probabilistic interpretations constitute the second semantic structure introduced in this section. Throughout the rest of this paper, we drop the word "probabilistic" whenever no confusion arises.

In (24) we show that this notion of (probabilistic) interpretation satisfies many general properties of probability, including Fenstad's criteria for defining probability functions on first-order languages [11]. Hereafter whenever no confusion arises, we may simply use $I$ to denote an interpretation, without referring to the world probability density function $WP$ with which $I$ is associated. The third semantic structure we find useful to define is a formula function which assigns closed intervals to basic formulas.
DEFINITION 9. (1) Let \( \mathcal{C}[0,1] \) denote the set of all closed subintervals of the unit interval \([0,1]\), i.e., the set of all (contiguous) closed intervals \([c,d]\) that are subsets of \([0,1]\).

(2) A formula function is a mapping \( hf(B_L) \rightarrow \mathcal{C}[0,1] \).

We use \( \mathcal{F} \mathcal{F} \) to denote the set of all formula functions associated with our language \( L \). The empty interval, denoted by \( \emptyset \), is a member of \( \mathcal{C}[0,1] \), because it may be represented as \([c_1,c_2]\) where \( c_2 < c_1 \). Intuitively, a formula function assigns a probability range to each ground basic formula. Thus, given a formula function, we can find world probability density functions that obey the ranges assigned by the formula function. This is achieved by setting up a set of linear constraints, as described in the following definition.

DEFINITION 10. (1) Let \( h \) be a formula function. A set of linear constraints, denoted by \( \mathcal{L} \mathcal{C}(h) \), is defined as follows. For all \( F_i \in hf(B_L) \), if \( h(F_i) = [c_i,d_i] \), then the inequality \( c_i \leqslant (\sum_{w_j \models F_i} p_j \in C^h, p_j) \leqslant d_i \) is in \( \mathcal{L} \mathcal{C}(h) \) (where \( p_j \)'s are used as specified in Definition 7). In addition, \( \mathcal{L} \mathcal{C}(h) \) contains the following two constraints: \( \sum_{w_j \in 2^{B^h, p_j} = 1} \) and \( (\forall w_j \in 2^{B^h}, p_j \geqslant 0) \).

(2) Let \( W \mathcal{P}(h) \) denote the solution set of \( \mathcal{L} \mathcal{C}(h) \).

It is easy to see that the solution set of \( \mathcal{L} \mathcal{C}(h) \) is a set of world probability density functions. Moreover, when any of these solutions, \( WP \), is extended to an interpretation \( I \), \( I \) assigns probabilistic truth values to basic formulas that satisfy the probability ranges assigned by the formula function \( h \). This completes the description of the process of finding probabilistic truth values based on the probability ranges specified in a pf-program.

In the set of linear constraints defined above, each basic formula in our language generates an inequality. If function symbols are allowed in our language, then \( hf(B_L) \) is infinite and the linear program has infinitely many constraints and infinitely many \( p_j \)'s. To our knowledge, the current theory of linear programming in infinite dimensional spaces (1) can only deal with semi-infinite linear programs which either have infinitely many variables (\( p_j \)'s in our case) or infinitely many constraints, but not both. This is the technical reason why function symbols are not supported in our framework.

Thus far, we have defined three semantical structures for probabilistic logic programs, viz., world probability density functions, probabilistic interpretations, and formula functions. The key difference between these three semantic structures is that world probability density functions assign probabilities to worlds, probabilistic interpretations assign probabilities
to *basic formulas*, and formula functions assign closed intervals to basic formulas. Formula functions may be used to generate *families* of world probability density functions. The lemma presented below exhibits a strong correspondence between interpretations and world probability density functions.

**Definition 11.** (1) Let \( WP_1 \) and \( WP_2 \) be two world probability density functions. We say that \( WP_1 = WP_2 \) if and only if \( WP_1(w) = WP_2(w) \) for all worlds \( w \in 2^{B_L} \).

(2) Let \( I_1 \) and \( I_2 \) be two probabilistic interpretations. We say that \( I_1 = I_2 \) if and only if \( I_1(F) = I_2(F) \) for all \( F \in bf(B_L) \).

Given two world probability density functions \( WP_1 \) and \( WP_2 \), suppose \( I_1 \) and \( I_2 \) are the interpretations associated with \( WP_1 \) and \( WP_2 \), respectively. Then it immediately follows that if \( WP_1 = WP_2 \), then \( I_1 = I_2 \). The following lemma shows the converse: that two distinct world probability density functions correspond to two different interpretations. In other words, a particular probabilistic interpretation \( I \) cannot be associated with two different world probability density functions.

**Lemma 1.** Let \( WP_1 \) and \( WP_2 \) be two world probability density functions, and let \( I_1 \) and \( I_2 \) be their corresponding interpretations respectively. If \( WP_1 \neq WP_2 \), then \( I_1 \neq I_2 \).

**Proof.** To prove the lemma, it suffices to prove the contrapositive that \( I_1 = I_2 \) implies \( WP_1 = WP_2 \). Let \( \{A_1, ..., A_n\} \) denote the Herbrand base \( B_L \) for some integer \( n \) (since \( B_L \) is finite). Then recall from Definition 7 that world probability density functions \( WP_1 \) and \( WP_2 \) assign a probability (a real number in \([0, 1]\)) to each world \( w \) in \( 2^{B_L} \) such that all those probabilities add up to 1.

**Claim.** For all \( 0 \leq m \leq n \), \( WP_1(w) = WP_2(w) \) for all worlds \( w \) such that the cardinality of \( w \) is not greater than \( m \); i.e., \( |w| \leq m \).

Proceed by induction on \( m \).

(i) Base case: \( m = 0 \).

There is only one world \( w \) such that \( |w| = 0 \); this is the world \( w = \emptyset \). Consider the probabilistic truth value of the disjunction \((A_1 \lor \cdots \lor A_n) \in bf(B_L)\). Thus, \( I_1(A_1 \lor \cdots \lor A_n) = \sum_{v \models (A_1 \lor \cdots \lor A_n) \text{ and } v \in 2^{B_L}} WP_1(v) \). But \( w = \emptyset \) is the only world such that \( w \not\models (A_1 \lor \cdots \lor A_n) \). Therefore, it follows that \( I_1(A_1 \lor \cdots \lor A_n) = 1 - WP_1(\emptyset) \). Similarly, it is also the case that \( I_2(A_1 \lor \cdots \lor A_n) = 1 - WP_2(\emptyset) \). Since \( I_1 = I_2 \), it follows that \( WP_1(\emptyset) = WP_2(\emptyset) \).

(ii) Inductive case.
Pick any world $w$ of cardinality $m$. Without loss of generality, suppose the world $w$ is \{\(A_{n-m+1}, \ldots, A_n\)\}. Consider the probabilistic truth value of \((A_1 \lor \cdots \lor A_{n-m}) \in \text{bf}(B_L)\). Hence, \(I_1(A_1 \lor \cdots \lor A_{n-m}) = (\sum_{v \models (A_1 \lor \cdots \lor A_{n-m}) \text{ and } v \in \mathcal{W}_L} WP_1(v))\). Note that the set of all worlds can be partitioned into two groups: one in which every world $v \models (A_1 \lor \cdots \lor A_{n-m})$, and the other in which every world $v \not\models (A_1 \lor \cdots \lor A_{n-m})$. But then in the latter group, each world $v \models (A_1 \land \cdots \land \overline{A_{n-m}})$. Therefore, each world $v$ in the latter group is a subset of $w$. Hence, it follows that \(I_1(A_1 \lor \cdots \lor A_{n-m}) = 1 - (\sum_{v \subseteq w} WP_1(v))\). Similarly, it follows that \(I_2(A_1 \lor \cdots \lor A_{n-m}) = 1 - (\sum_{v \subseteq w} WP_2(v))\). But for all $v \subseteq w$, it is necessary that $|v| \leq m-1$. Thus, by the induction hypothesis, \(WP_1(v) = WP_2(v)\) for all $v \subseteq w$. In other words, it must be the case that \((\sum_{v \subseteq w} WP_1(v)) = (\sum_{v \subseteq w} WP_2(v))\). But then since \(I_1 = I_2\), it is necessary that \((\sum_{v \subseteq w} WP_1(v)) = (\sum_{v \subseteq w} WP_2(v))\). Hence, it follows that \(WP_1(w) = WP_2(w)\). This completes the induction and the proof of the lemma.

3.2. Fixpoint Theories for pf-programs

We are now in a position to define a fixpoint operator $T_p$ for pf-programs $P$.

**Definition 12.** (1) Let $C \equiv F: \mu \leftarrow F_1: \mu_1 \land \cdots \land F_n: \mu_n$ be a pf-clause. A ground instance of $C$ is any clause obtained by replacing all object variables in $C$ by members of the Herbrand universe of $L$ and all annotation variables in $C$ by numbers in \([0,1]\). In addition, different occurrences of the same variable must be replaced by the same member of the Herbrand universe of the same number in \([0,1]\). Any ground clause whose head becomes the empty set in this process is discarded.

(2) If $P$ is a pf-program, let $\text{grd}(P)$ denote the set of all ground instances of clauses in $P$.

The reason clauses having $\emptyset$ as the annotation in their heads are not included in $\text{grd}(P)$ is that they may give rise to inconsistency. We show later that $\text{grd}(P)$ is equivalent to $P$ (25). Hereafter we use the notation $\text{min}_Q(Exp)$ and $\text{max}_Q(Exp)$ to denote the minimization and maximization of the expression $Exp$ subject to the set of constraints $Q$.

**Definition 13.** Suppose $P$ is a pf-program and $h$ is a formula function.

(1) Define an intermediate operator $S_p: \mathcal{F} \mathcal{F} \rightarrow \mathcal{F} \mathcal{F}$ as follows: For all $F \in \text{bf}(B_L)$, $S_p(h)(F) = \bigcap M_p$, where $M_p = \{x| F: x \leftarrow F_1: x_1 \land \cdots \land F_n: x_n \text{ is a ground instance of a clause in } P, \text{ and for all } 1 \leq i \leq n, h(F_i) \subseteq x_i\}$. In particular, if $M_p$ is empty, set $S_p(h)(F) = [0,1]$. 

(2) Define \( T_p : \mathcal{F} \mathcal{F} \to \mathcal{F} \mathcal{F} \) as follows: (i) If \( \mathcal{L} \mathcal{P}(S_p(h)) \) is non-empty (i.e., \( \mathcal{L} \mathcal{C}(S_p(h)) \) has solutions), then for all \( F \in hf(B_L) \), \( T_p(h)(F) = [c_F, d_F] \), where

\[
    c_F = \min_{\mathcal{L} \mathcal{P}(S_p(h))} \left( \sum_{W_j \models F \text{ and } W_j \in 2^{B_L}} p_j \right)
\]

and

\[
    d_F = \max_{\mathcal{L} \mathcal{P}(S_p(h))} \left( \sum_{W_j \models F \text{ and } W_j \in 2^{B_L}} p_j \right).
\]

(ii) Otherwise, if \( \mathcal{L} \mathcal{P}(S_p(h)) \) is empty, then for all \( F \in hf(B_L) \), \( T_p(h)(F) = \emptyset \).

Informally, \( S_p(h) \) is a one-step immediate consequence operator that determines the probability ranges of basic formulas by one-step deductions of the pf-clauses in \( P \). But since basic formulas can appear as the heads of pf-clauses, as an example, the following situation may arise: \( S_p(h)(A \lor B) = [0, 0] \), but \( S_p(h)(A) = S_p(h)(B) = [1, 1] \). By regarding \([1, 1]\) as true and \([0, 0]\) as false, these range assignments are not consistent. In general, "local" assignments of probability ranges to formulas may not be "globally" consistent. Hence, the linear program \( \mathcal{L} \mathcal{C}(S_p(h)) \) is set up to ensure that all assignments are consistent. Then \( T_p \) assigns to each formula a probability range that satisfies every constraint in the linear program. In (25) we show that \( T_p \) is monotonic, and thus there exists a least fixpoint \( \text{lfp}(T_p) \) of \( T_p \). The upward iteration of \( T_p \) is defined in the usual way (cf. (25)).

Given two formula functions \( h_1 \) and \( h_2 \), we say that \( h_1 \leq h_2 \) iff \( \forall F \in hf(B_L), h_1(F) \supseteq h_2(F) \). The set \( \mathcal{F} \mathcal{F} \) of formula functions forms a complete lattice with respect to the \( \leq \) ordering (25). In (25) we also show that \( T_p \) is monotonic, and thus there exists a least fixpoint \( \text{lfp}(T_p) \) of \( T_p \).

3.3. Probabilistic Model Theory for pf-Programs

As shown above, the fixpoint operator \( T_p \) associated with a pf-program \( P \) maps formula functions to formula functions which assign probability ranges to basic formulas. Recall from Definition 10 that a formula function \( h \) corresponds to a family \( \mathcal{L} \mathcal{P}(h) \) of world probability density functions that obey the ranges assigned by the formula function. Thus, each world probability density function can be extended to an interpretation. Thus, we are now in a position to describe the relationship between fixpoints (i.e., formula functions) and probabilistic models (i.e., probabilistic interpretations that satisfy the program). The notion of probabilistic satisfaction, denoted by \( \models_p \), is described in (25), but is informally recounted below.
Suppose $WP$ is a world probability density function, and $I$ is the interpretation associated with $WP$. Further assume that $F$ is in $bf(B_L)$, and $[c, d]$ is a closed sub-interval of $[0, 1]$. $I \models_p F : [c, d]$ if and only if $I(F) \in [c, d]$. The interpretations of conjunction, disjunction, and quantification are the usual ones (cf. (25) for full details). The only unusual case is that of universal quantification over an annotation variable. $I \models_p (\forall V)(C)$ if and only if $I \models_p (C(V/c))$ for all $c \in [0, 1]$ such that $\mu(V/c) \neq \emptyset$, where $V$ represents an annotation variable occurring in annotation term $\mu$.

Example 6. Suppose we want to check whether a given interpretation $I$ satisfies the clause $A : [0.2, V] \leftarrow B : [0.1, V]$. Then we need to check whether $I$ satisfies $A : [0.2, c] \leftarrow B : [0.1, c]$ for all $c \geq 0.2$. To do so, we check for each $c \geq 0.2$, whether $I(A) \in [0.2, c]$, or whether $I(B) \notin [0.1, c]$.

As usual, we also use the notation $\models_p$ to denote logical consequence. We say that program $P$ probabilistically entails formula $F$, denoted by $P \models_p F$, if and only if $I$ is an interpretation that satisfies each clause in $P$, then $I \models_p F$. It turns out that $P$ and $grd(P)$ have the same (probabilistic) models.

We denote the family of interpretations associated with a formula function $h$ by $\mathcal{I}(h)$. In (25), we prove that the family $\mathcal{I}(h)$ of interpretations is a family of models for pf-program $P$ if and only if $h$ is pre-fixpoint of $T_P$, i.e. $T_P(h) \leq h$. We will refer to this result as the "pre-fixpoint theorem." A consequence of the pre-fixpoint theorem is that the family of models associated with the least fixpoint of $T_P$, denoted $lfp(T_P)$, completely characterizes all (probabilistic) models for $P$. In other words, the family $\mathcal{I}(lfp(T_P))$ of models for $P$ contains every family of models associated with a pre-fixpoint of $T_P$. Consequently, given a program $P$, for all basic formulas $F \in bf(B_L)$, it is proven that $P \models_p (F : \beta)$ if and only if $lfp(T_P)(F) \subseteq \beta$.

4. Stability of Formula Functions

In this section we introduce the first notion of stability—stable formula functions. We investigate the properties of a stable formula function, and in particular we study its relationship with a fixpoint operator associated with a gp-program.

4.1. A Fixpoint Operator for gp-Programs

In the presence of negation, the fixpoint operator associated with a pf-program (cf. Definition 13) must be extended to handle negation. Recall that the intuitive meaning of the negated annotated formula $\neg(G : \beta)$ is "It is not provable that the probability of $G$ lies in the interval $\beta." In other
words, if the probability of $G$ lies in the range $\alpha$, then $\alpha$ is not a subset of $\beta$. The following definition formalizes this observation. Hereafter we denote the fixpoint operator associated with a gp-program $P$ by $T'_P$.

**Definition 14.** Suppose $P$ is a gp-program and $h$ is a formula function.

1. Define an intermediate operator $S'_p : \mathcal{F} \mathcal{F} \rightarrow \mathcal{F} \mathcal{F}$ as follows: For all $F \in hf(B_L)$, $S'_p(h)(F) = \bigcap M'_F$, where $M'_F = \{ \alpha | F : \alpha \leftarrow F_1 : \alpha_1 \land \cdots \land F_n : \alpha_n \land \neg (G_1 : \beta_1) \land \cdots \land \neg (G_m : \beta_m) \}$ is a ground instance of a clause in $P$, for all $1 \leq i \leq n, h(F_i) \subseteq \alpha_i$, and for all $1 \leq j \leq m, h(G_j) \subseteq \beta_j$. In particular, if $M'_F$ is empty, set $S'_p(h)(F) = [0, 1]$.

2. Define $T'_p : \mathcal{F} \mathcal{F} \rightarrow \mathcal{F} \mathcal{F}$ as follows: (i) If $\forall \mathcal{P}(S'_p(h))$ is non-empty (i.e., $\forall \mathcal{P}(S'_p(h))$ has solutions), then for all $F \in hf(B_L)$, $T'_p(h)(F) = [c_F, d_F]$, where

$$
c_F = \min_{\forall \mathcal{P}(S'_p(h))} \left( \sum_{W_j \in F \land W_j \in 2^{B_L}} p_j \right),
$$

and

$$
d_F = \max_{\forall \mathcal{P}(S'_p(h))} \left( \sum_{W_j \in F \land W_j \in 2^{B_L}} p_j \right).
$$

(ii) Otherwise, if $\forall \mathcal{P}(S'_p(h))$ is empty, then for all $F \in hf(B_L)$, $T'_p(h)(F) = \emptyset$.

According to the definition above, it is easy to see that when a gp-program $P$ is negative-free, $S'_p$ and $T'_p$ reduce to $S_p$ and $T_p$, as defined in Definition 13. Thus, $T'_p$ extends $T_p$ to handle clauses whose bodies contain negated annotated basic formulas. The following example shows that $T'_p$ is not monotonic.

**Example 7.** Consider the gp-program $P$ similar to the one described in Example 3:

$$p : [0.95, 1] \leftarrow \neg(q : [0.49, 0.51]).$$

Suppose $h_1$ is a formula function that assigns $[0, 1]$ to $q$, and $h_2$ is one that assigns $[0.5, 0.5]$ to $q$. Suppose that $h_1$ and $h_2$ assign $[0, 1]$ to all other basic formulas. Thus, it is the case that $h_1 \leq h_2$. But then $T'_p(h_1)$ assigns $[0.95, 1]$ to $p$, while $T'_p(h_2)$ assigns $[0, 1]$ to $p$. Therefore, $T'_p(h_1)$ is not necessarily less than or equal to $T'_p(h_2)$.

4.2. **Stable Formula Functions**

In the following we define the notion of stable formula functions, adapted from the stable model semantics proposed by Gelfond and Lifschitz (14).
We ultimately show that if there exists a stable formula function with respect to a gp-program $P$, the formula function is a minimal fixpoint of $T_P$.

**Definition 15.** Given a gp-program $P$ and a formula function $h$, the *formula-function-transform* (**ff-transform** for short) of $P$ based on $h$, denoted by $ff(P, h)$, is defined as follows:

1. Given any clause $C' \equiv F : x \leftarrow F_1 : x_1 \land \cdots \land F_n : x_n \land \neg(G_1 : \beta_1) \land \cdots \land \neg(G_m : \beta_m)$ in $grd(P)$, if $h(G_j) \not\subseteq \beta_j$ for all $1 \leq j \leq m$, then the clause $C \equiv F : x \leftarrow F_1 : x_1 \land \cdots \land F_n : x_n$ is included in $ff(P, h)$.

2. Nothing else is in $ff(P, h)$.

**Example 8.** Consider the gp-program $P$ that consists of the single clause

$$p : [0.95, 1] \leftarrow \neg(q : [0.49, 0.51]),$$

where $p, q$ are ground. Then given the formula function $h_1$ such that $h_1(p) = [0.95, 1]$ and $h_1(q) = [0, 1]$, the ff-transform of $P$ based on $h_1$ consists of the single clause

$$p : [0.95, 1] \leftarrow.$$

On the other hand, given another formula function $h_2$ such that $h_2(p) = [0, 1]$ and $h_2(q) = [0.5, 0.5]$, the ff-transform of $P$ based on $h_2$ does not contain any clause.

According to Definition 15, it is obvious that clauses in $ff(P, h)$ are negation-free. Hence, $ff(P, h)$ is a pf-program, and it is legitimate to apply the fixpoint operator $T_{ff(P, h)}$ as defined in Definition 13.

**Definition 16.** Let $P$ be a gp-program. A formula function $h$ is **stable** with respect to $P$ if $h$ is identical to the least fixpoint of $T_{ff(P, h)}$, i.e., $h = lfp(T_{ff(P, h)})$.

**Example 9.** Continue with Example 8. Since $ff(P, h_1)$ contains only the clause

$$p : [0.95, 1] \leftarrow,$$

the least fixpoint $lfp(T_{ff(P, h_1)})$ assigns $[0.95, 1]$ to $p$ and $[0, 1]$ to $q$. As these ranges are exactly the ones assigned by $h_1$, $h_1$ is stable with respect to $P$. On the other hand, since $ff(P, h_2)$ does not contain any clause, the
least fixpoint $\text{lfp}(T_{ff(P, h)})$ assigns $[0, 1]$ to both $p$ and $q$. Thus, $h_2$ is not stable with respect to $P$. In fact, $h_1$ is the only stable formula function with respect to $P$. An outline of a proof is as follows. Since $P$ does not contain any clause whose head is $q$, then for all formula function $h$, $ff(P, h)$ also cannot contain any clause whose head is $q$. Thus for $h$ to be stable, $h$ must assign $[0, 1]$ to $q$. Then, $ff(P, h)$ must consist of the single clause $p : [0.95, 1] \leftarrow$. Therefore, for $h$ to be stable, $h$ to be stable, $h$ must assign $[0.95, 1]$ to $p$. Hence, $h_1$ is the only stable formula function.

**Example 10.** Consider the gp-program $P$ for Benjy and Fido shown in Example 4. Let a formula function $h_1$ assign $[0, 0]$ to $\text{bark(benjy)}$, $[1, 1]$ to $\text{abnormal(benjy)}$, $[0, 95, 1]$ to $\text{bark(fido)}$, and $[0, 1]$ to $\text{abnormal(fido)}$. Then $ff(P, h_1)$ consists of the following clauses:

\[
\begin{align*}
\text{bark(fido)} : [0.95, 1] &\leftarrow \text{dog(fido)} : [1, 1] \\
\text{dog(fido)} : [1, 1] &\leftarrow \\
\text{dog(benjy)} : [1, 1] &\leftarrow \\
\text{bark(benjy)} : [0, 0] &\leftarrow \\
\text{abnormal(benjy)} : [1, 1] &\leftarrow \text{bark(benjy)} : [0, 0] \\
\text{abnormal(fido)} : [1, 1] &\leftarrow \text{bark(fido)} : [0, 0].
\end{align*}
\]

It is easy to check that $h_1 = \text{lfp}(T_{ff(P, h_1)})$. Therefore, $h_1$ is stable. In fact, it is easy to verify that $h_1$ is the only stable formula function for this gp-program.

The examples below show there are gp-programs that do not have stable formula functions and gp-programs that have more than one stable formula function.

**Example 11.** The gp-program that consists of the single clause

\[
p : [0.95, 1] \leftarrow \neg(p : [0.95, 1])
\]

does not have a stable formula function. The outline of a proof is as follows. Suppose a formula function $h$ assigns any range to $p$ such that $h(p) \notin [0.95, 1]$. Then it is easy to check that the least fixpoint of the operator associated with the $ff$-transform of the program based on $h$ assigns $[0.95, 1]$ to $p$. On the other hand, if a formula function $h$ assigns any range to $p$ such that $h(p) \subseteq [0.95, 1]$, the associated least fixpoint assigns $[0, 1]$ to $p$. 
Example 12. The gp-program that consists of the clauses

\[ p : [0, 95, 1] \leftarrow \neg(q : [0.49, 0.51]) \]
\[ q : [0.49, 0.51] \leftarrow \neg(p : [0.95, 1]) \]

has two stable formula functions: \( h_1 \) such that \( h_1(p) = [0.95, 1] \) and \( h_1(q) = [0, 1] \), and \( h_2 \) such that \( h_2(p) = [0, 1] \) and \( h_2(q) = [0.49, 0.51] \).

4.3. Relationship with Minimal Fixpoints of \( T'_p \)

Intuitively, a stable formula function with respect to a gp-program makes guesses on the probability ranges assigned by the program to basic formulas. In particular, the following theorem shows that a stable formula function with respect to gp-program \( P \) is a minimal fixpoint of \( T'_p \) as defined in Definition 14. But in order to prove the theorem, we need the following lemmas.

**Lemma 2.** Let \( P_1, P_2 \) be gp-programs and \( h_1, h_2 \) be the formula functions. If \( S'_{P_i}(h_j)(F) \subseteq S'_{P_i}(h_2)(F) \) for all basic formulas \( F \in bf(B_L) \), then it must be true that \( T'_{P_i}(h_1)(F) \subseteq T'_{P_i}(h_2)(F) \) for all basic formulas \( F \in bf(B_L) \).

**Proof.** Extension of the proof of a similar lemma for pf-programs (25).

Note that in the above lemma, if any one of the \( P_i \)'s \( (i = 1, 2) \) is a pf-program, then \( S'_{P_i} \) and \( T'_{P_i} \) is equivalent to \( S_P \) and \( T_P \), respectively. This observation is often used when we apply the lemma of other proofs.

**Lemma 3.** Let \( h \) be a stable formula function with respect to the gp-program \( P \). Then \( h \) is a fixpoint of \( T'_P \), i.e., \( T'_P(h) = h \).

**Proof. Claim.** For all basic formulas \( F \in bf(B_L) \), it is the case that \( M_P = M'_P \), where \( M_P = \{ \alpha \mid \exists C \equiv F: \exists F_1 : \alpha_1 \wedge \cdots \wedge F_n : \alpha_n \ \text{is a ground instance of a clause in} \ ff(P, h), \ \text{and for all} \ 1 \leq i \leq n, h(F_i) \subseteq \alpha_i \} \), and \( M'_P = \{ \alpha \mid C' \equiv F: \exists F_1 : \alpha_1 \wedge \cdots \wedge F_n : \alpha_n \wedge \neg(G_1 : \beta_1) \wedge \cdots \wedge \neg(G_m : \beta_m) \ \text{is a ground instance of a clause in} \ P, \ \text{for all} \ 1 \leq i \leq n, h(F_i) \subseteq \alpha_i, \ \text{and for all} \ 1 \leq j \leq m, h(G_j) \not\subseteq \beta_j \} \).

**Case 1.** \( M'_P \) is not empty.

Consider any clause \( C' \) qualified in \( M'_P \), i.e., \( \alpha \in M'_P \). Since for all \( 1 \leq j \leq m, h(G_j) \not\subseteq \beta_j \), then by Definition 15, the clause \( F: \exists F_1 : \alpha_1 \wedge \cdots \wedge F_n : \alpha_n \) is in \( ff(P, h) \) and thus qualified in \( M_F \), i.e., \( \alpha \in M_F \). On the other hand, consider any clause \( C \) qualified in \( M_F \); i.e., \( \alpha \in M_F \). Then by Definition 15, \( C \) must correspond to some ground instance \( C' \equiv F: \exists F_1 : \alpha_1 \wedge \cdots \wedge F_n : \alpha_n \wedge \neg(G_1 : \beta_1) \wedge \cdots \wedge \neg(G_m : \beta_m) \) in \( P \) such that for
all \(1 \leq j \leq m\), \(h(G_j) \not\in \beta_j\). Thus, \(C'\) is qualified in \(M_F\); i.e., \(\alpha \in M_F\). Hence, it is necessary that \(M_F = M'_F\).

**Case 2.** \(M'_F\) is empty.

Then there is no ground clause \(C' \equiv F : \alpha \leftarrow F_1 : \alpha_1 \wedge \cdots \wedge F_n : \alpha_n \wedge \neg(G_1 : \beta_1) \wedge \cdots \wedge \neg(G_m : \beta_m)\) in \(P\) such that for all \(1 \leq i \leq n\), \(h(F_i) \subseteq \alpha_i\), and for all \(1 \leq j \leq m\), \(h(G_j) \not\subseteq \beta_j\). Thus, there cannot be any clause qualified in \(M_F\), and \(M_F = M'_F\) is empty. Conversely, if \(M_F\) is empty, then there is no ground clause \(C \equiv F : \alpha \leftarrow F_1 : \alpha_1 \wedge \cdots \wedge F_n : \alpha_n\) in \(ff(P, h)\) such that for all \(1 \leq i \leq n\), \(h(F_i) \subseteq \alpha_i\). Thus, there cannot be any clause qualified in \(M'_F\), and it is necessary that \(M_F = M'_F\). This completes the proof that \(M_F = M'_F\) for all \(F \in bf(B_L)\).

Now according to the claim above, it must be the case that \(S'_{h'}(h)(F) = S_{ff(P, h)}(h)(F)\) for all basic formulas \(F\), by Definitions 13 and 14. Then by Lemma 2, it follows that \(T'_{h'}(h)(F) = T_{ff(P, h)}(h)(F)\) for all basic formulas \(F\), and thus \(T'_{h'}(h) = T_{ff(P, h)}(h)\). But since \(h\) is stable, \(h = \text{fp}(T_{ff(P, h)})\). In particular, \(h\) is a fixpoint of \(T_{ff(P, h)}\); i.e., \(h = T_{ff(P, h)}(h)\). Hence, it is necessary that \(T'_{h'}(h) = h\); i.e., \(h\) is a fixpoint of \(T'_{h'}\).}

**Theorem 1.** Let \(h\) be stable formula function with respect to the gp-program \(P\). Then \(h\) is a minimal fixpoint of \(T'_{h'}\).

**Proof.** By Lemma 3, \(h\) is a fixpoint of \(T'_{h'}\). Thus, it suffices to prove that \(h\) is minimal. Suppose there exists a formula function \(h_1\) such that \(h_1\) is a fixpoint of \(T'_{h'}\), and \(h_1 < h\) (i.e., \(h_1 \not\leq h\) and \(h_1 \neq h\)). Then there must exist a basic formula \(G_0 \in bf(B_L)\) such that \(h_1(G_0) \supseteq h(G_0)\). Now let \(\gamma = \min\{k \mid \text{there exists } F \in bf(B_L) \text{ such that } k \text{ is the smallest ordinal such that } h_1(F) \not\subseteq T_{ff(P, h_1)}(k(F))\}\). Note that as any set of ordinals is well-ordered (16, p. 79), \(\gamma\) is well-defined. We proceed by transfinite induction on \(\gamma\) and show that a contradiction is bound to arise. It is easy to see that \(\gamma > 0\) because \(h_1(F)\) cannot be a strict super set of \([0, 1]\).

(i) **Base case.** \(\gamma = 1\). Then there exists some basic formula \(F_0 \in bf(B_L)\) such that \(h_1(F_0) \not\subseteq T_{ff(P, h_1)}(1(F_0))\). To simplify notation, let \(g = T_{ff(P, h_1)}(1)\). Now from Definition 13, it follows that for any basic formula \(F \in bf(B_L)\), \(S_{ff(P, h_1)}(g)(F) = \bigcap M_F\) where \(M_F = \{\alpha \mid C \equiv F : \alpha \leftarrow \text{ is a ground instance of a clause in } ff(P, h)\}\).

(ii) **Case 1.** \(M_F\) is not empty.

Now for each clause \(C\) qualified in \(M_F\) (i.e., \(\alpha \in M_F\)), \(C' \equiv F : \alpha \leftarrow \neg(G_1 : \beta_1) \wedge \cdots \wedge \neg(G_m : \beta_m)\) is a ground instance of a clause in \(P\) such that \(h(G_j) \not\subseteq \beta_j\) for all \(1 \leq j \leq m\). Since \(h_1(G) \supseteq h(G)\) for all \(G \in bf(B_L)\), it follows that \(h_1(G_j) \not\subseteq \beta_j\) for all \(1 \leq j \leq m\). Hence, the clause \(C'\) is qualified in \(M'_{F}\) (i.e., \(\alpha \in M'_{F}\)) where \(S'_{h_1}(h)(F) = \bigcap M'_{F}\) and \(M'_{F} = \{\alpha \mid F : \alpha \leftarrow F_1 : \alpha_1 \wedge \cdots \wedge F_n : \alpha_n \wedge \neg(G_1 : \beta_1) \wedge \cdots \wedge \neg(G_m : \beta_m)\}\).
ground instance of a clause in $P$, for all $1 \leq i \leq n$, $h_i(F_i) \subseteq \alpha_i$, and for all $1 \leq j \leq m$, $h_i(G_j) \not\in \beta_j$. Hence, $M_F \subseteq M'_F$.

Case 2. $M_F$ is empty.

Then it is obvious that $M_F \subseteq M'_F$.

Thus, combining the two cases above, it is necessary that $S'_p(h_1)(F) = (\bigcap M'_F) \subseteq (\bigcap M_F) = S_{\text{ff}(P, h)}(g)(F)$. Since this subset relation is true for all basic formulas $F$, then by Lemma 2, it must be the case that $T'_p(h_1)(F) \subseteq T_{\text{ff}(P, h)}(g)(F)$ for all basic formulas $F$. In particular, it is necessary that $T'_p(h_1)(F_0) \subseteq T_{\text{ff}(P, h)}(g)(F_0) = T_{\text{ff}(P, h)} \uparrow \gamma(F_0)$. But since $h_1$ is a fixpoint of $T_p$, it must be true that $h_1(F_0) = T_p(h_1)(F_0) \subseteq T_{\text{ff}(P, h)} \uparrow \gamma(F_0)$, a contradiction!

(ii) **Inductive Case.** There are two parts, one in which $\gamma$ is a successor ordinal and one where $\gamma$ is a limit ordinal.

(a) **Successor Ordinal Case.** $\gamma = \eta + 1$. Then there exists some basic formula $F_0 \in bf(B_L)$ such that $h_1(F_0) \not\in T_{\text{ff}(P, h)} \uparrow (\eta + 1)(F_0)$. To simplify notation, let $g$ be $T_{\text{ff}(P, h)} \uparrow \eta$. Now from Definition 13, it is the case that for any basic formula $F \in bf(B_L)$, $S_{\text{ff}(P, h)}(g)(F) = \bigcap M_F$, where $M_F = \{ \alpha | C \equiv F : \alpha \leftarrow F_1 : \alpha_1 \land \ldots \land F_n : \alpha_n \}$ is a ground instance of a clause in $\text{ff}(P, h)$, and for all $1 \leq i \leq n$, $g(F_i) \subseteq \alpha_i$.

Case 1. $M_F$ is empty.

Now for each clause $C$ qualified in $M_F$ (i.e., $\alpha \in M_F$), $C' \equiv F : \alpha \leftarrow F_1 : \alpha_1 \land \ldots \land F_n : \alpha_n \land \neg (G_1 : \beta_1) \land \ldots \land \neg (G_m : \beta_m)$ is a ground instance of a clause in $P$ such that $h(G_j) \not\in \beta_j$ for all $1 \leq j \leq m$. Since $h_1(G) \subseteq h(G)$ for all $G \in bf(B_L)$, it is necessary that $h_1(G) \not\in \beta_j$ for all $1 \leq j \leq m$. Now by the induction hypothesis, for all $\delta \leq \eta$ and $G \in bf(B_L)$, it is the case that $h_1(G) \subseteq T_{\text{ff}(P, h)} \uparrow \delta(G)$. Thus, for all $1 \leq i \leq n$, it is necessary that $h_1(F_i) \subseteq g(F_i)$. Therefore, it is true that for all $1 \leq i \leq n$, $h_1(F_i) \subseteq \alpha_i$. Hence the clause $C'$ is qualified in $M_F$ (i.e., $\alpha \in M_F$), where $S_p(h_1)(F) = \bigcap M_F$ and $M'_F = \{ \alpha | F : \alpha \leftarrow F_1 : \alpha_1 \land \ldots \land F_n : \alpha_n \land \neg (G_1 : \beta_1) \land \ldots \land \neg (G_m : \beta_m)$ is a ground instance of a clause in $P$, for all $1 \leq i \leq n$, $h_1(F_i) \subseteq \alpha_i$, and for all $1 \leq j \leq m$, $h_1(G_j) \not\in \beta_j \}$. Hence, $M_F \subseteq M'_F$.

Case 2. $M_F$ is empty.

Then it is obvious that $M_F \subseteq M'_F$.

Combining the two cases above, it follows that $S'_p(h_1)(F) = (\bigcap M'_F) \subseteq (\bigcap M_F) = S_{\text{ff}(P, h)}(g)(F)$. Since this subset relation is true for all basic formulas $F$, then by Lemma 2, it must be the case that $T'_p(h_1)(F) \subseteq T_{\text{ff}(P, h)}(g)(F)$ for all basic formulas $F$. In particular, it is necessary that $T'_p(h_1)(F_0) \subseteq T_{\text{ff}(P, h)}(g)(F_0) = T_{\text{ff}(P, h)} \uparrow \gamma(F_0)$. But since $h_1$ is a fixpoint of $T_p$, it must be true that $h_1(F_0) = T_p(h_1)(F_0) \subseteq T_{\text{ff}(P, h)} \uparrow \gamma(F_0)$, a contradiction!

(b) **Limit Ordinal Case.** Suppose $\gamma$ is a limit ordinal. Then there exists some basic formula $F_0$ such that $h_1(F_0) \not\in T_{\text{ff}(P, h)} \uparrow \gamma(F_0)$. By the
definition of the upward iteration of the $T$ operator, it is true that $T_{hf(P, h)} \uparrow \gamma(F_0) = \bigcap_{\delta \leq \gamma} T_{hf(P, h)} \uparrow \delta(F_0)$. But by the induction hypothesis, it is necessary that for all $\delta < \gamma$, $h_1(F_0) \subseteq T_{hf(P, h)} \uparrow \delta(F_0)$. Hence, it must be the case that $h_1(F_0) \subseteq \bigcap_{\delta < \gamma} T_{hf(P, h)} \uparrow \delta(F_0) = T_{hf(P, h)} \uparrow \gamma(F_0)$, a contradiction! This completes the induction.

Now since $h$ is stable, there exists an ordinal $\delta$ such that $h(F) = T_{hf(P, h)} \uparrow \delta(F)$, for all basic formulas $F \in hf(B_x)$. Hence, from the induction, it is necessary that $h_1(F) \subseteq h(F)$ for all $F \in hf(B_x)$. In particular, $h_1(G_0) \subseteq h(G_0)$, a contradiction!

By Theorem 1, we can conclude that every stable formula function is a minimal fixpoint of $T'$. But the following example shows that the converse is not true.

**EXAMPLE 13.** Consider the gp-program $P$ that consists of the following clauses:

$$p : [0.95, 1] \leftarrow \neg(p : [0.95, 1])$$
$$p : [0.95, 1] \leftarrow q : [1, 1]$$
$$q : [1, 1] \leftarrow q : [1, 1].$$

It is easy to check that the formula function $h$ that assigns $[0.95, 1]$ to $p$ and $[1, 1]$ to $q$ is a fixpoint of $T'_p$. Consider any formula function $h_1$ such that $h_1(p) = [c, 1]$, where $c < 0.95$. It is easy to check that $h_1$ cannot be a fixpoint of $T'_p$. Similarly, any formula function $h_1$ such that $h_1(q) = [c, 1]$, where $c < 1$, cannot be a fixpoint. This is because when $h_1(q) \neq [1, 1]$, $P$ behaves the same as the program discussed in Example 11. Hence, we can conclude that $h$ is a minimal fixpoint. But $ff(P, h)$ consists of the two clauses

$$p : [0.95, 1] \leftarrow q : [1, 1]$$
$$q : [1, 1] \leftarrow q : [1, 1],$$

then the least fixpoint $lfp(T_{hf(P, h)})$ assigns $[0, 1]$ to both $p$ and $q$. Hence, $h$ is not stable.

Thus far, we have introduced the notion of a stable formula function and shown that a stable formula function for a gp-program $P$ has the property that it is a minimal fixpoint of $T'_p$. However, the converse is not true. Recall (25) that in the case of a gp-program $P$ without negation, every prefixpoint of $T_p$ corresponds to a family of models for $P$. In the presence of negation, a natural question to ask is whether the family of interpretations corresponding to a stable formula function is a family of models for the gp-program. Among other results, this question is answered in the following section.
5. Stability of Families of Probabilistic Models

In this section we introduce another notion of stability—stable families of (probabilistic) models. We examine the properties of a stable family and also identify the relationship between stable formula functions and stable families of models. As shown later, this comparison enhances our understanding of the two notions of stability.

5.1. Stable Families of Probabilistic Models

While we have investigated the fixpoint theory for gp-programs in the previous section, here we study the model theory. Following the stable semantical approach, we base this model theory on the one for gp-programs without negation, i.e., pf-programs. Below we introduce the notion of stable families of (probabilistic) models.

**Definition 17.** Let \( P \) be a gp-program and \( S \) be a set of (probabilistic) interpretations. Define the family-probabilistic-model transform of \( P \) based on \( S(fpm\text{-transform}) \) for short, denoted by \( fpm(P, S) \), as follows:

1. Given a clause \( C' \equiv F : x \leftarrow F_1 : x_1 \land \cdots \land F_n : x_n \land \neg (G_1 : \beta_1) \land \cdots \land \neg (G_m : \beta_m) \) in \( \text{grd}(P) \), if there exists an interpretation \( I \in S \) such that \( I(G_j) \notin \beta_j \) for all \( 1 \leq j \leq m \), then the clause \( C \equiv F : x \leftarrow F_1 : x_1 \land \cdots \land F_n : x_n \) is included in \( fpm(P, S) \).

2. Nothing else is in \( fpm(P, S) \).

Unlike the situation with stable formula functions, we begin this time by guessing a family of interpretations. According to the definition above, it is obvious that clauses in \( fpm(P, S) \) are negation-free. Therefore, \( fpm(P, S) \) is a pf-program, and it is legitimate to use the fixpoint operator \( T_{fpm}(P, S) \) as defined in Definition 13.

**Definition 18.** A set \( S \) of (probabilistic) interpretations is called a stable family of (probabilistic) models for a gp-program \( P \) if \( S \) is identical to the family of models associated with the least fixpoint of \( T_{fpm}(P, S) \), i.e., \( S = \mathcal{J}(fp(T_{fpm}(P, S))) \).

**Example 14.** Consider the gp-program \( P \) again:

\[
p : [0.95, 1] \leftarrow \neg (q : [0.49, 0.51]).
\]

Let \( S \) be the family of interpretations defined as follows: \( S = \{ I \mid I(p) = s_1 + s_2, I(q) = s_1 + s_3, I(p \land q) = s_1, I(p \lor q) = s_1 + s_2 + s_3, 0.95 \leq s_1 + s_2 \leq 1, s_1 + s_2 + s_3 + s_4 = 1, \text{ and } s_i \geq 0 \text{ for all } i = 1, 2, 3, 4 \} \).
Then consider the interpretation \( I_1 \) that assigns 0 to \( q \) and \((p \land q)\), and that assigns 1 to \( p \) and \((p \lor q)\). It is obvious that by choosing \( s_2 = 1 \) and \( s_1 = s_3 = s_4 = 0 \), \( I_1 \) is a member of \( S \). Thus, by Definition 17, \( fpm(P, S) \) consists of the following clause:

\[
p : [0.95, 1] \leftarrow.
\]

Now the least fixpoint \( lfp(T_{fpm(P, S)}) \) assigns the range \([0.95, 1]\) to \( p \). By Definition 10, it is easy to verify that the family \( \mathcal{F}(lfp(T_{fpm(P, S)})) \) of interpretations associated with the least fixpoint is the same as \( S \). Hence, \( S \) is called a stable family of models.

So far, even though we have been using the word "model" in our definitions and discussion in this section, we have not shown that every interpretation in \( S \) is indeed a model for gp-program \( P \). This is proved in Theorem 2 below. But first we need to extend the notion of probabilistic satisfaction to handle negated annotated basic formulas.

**Definition 19 (Extension of Probabilistic Satisfaction).** Let \( I \) be a (probabilistic) interpretation, \( G \) in \( bf(B_L) \), and \([c, d]\) a closed sub-interval of \([0, 1]\).

1. \( I \models \neg G : [c, d] \) iff \( I \not\models G : [c, d] \) iff \( I(G) \notin [c, d] \);
2. All other cases are exactly the same as before.

**Theorem 2.** Let \( P \) be a gp-program, and let \( S \) be a family of interpretations. Suppose \( S \) is identical to the family of models associated with the least fixpoint of \( T_{fpm(P, S)} \); i.e., \( S = \mathcal{F}(lfp(T_{fpm(P, S)})) \). Then for all interpretations \( I \) in \( S, I \) is a model for \( P \).

**Proof.** Let \( I \) be any member (interpretation) of \( S \). Consider any clause \( C' \equiv F : x \leftarrow F_1 : x_1 \land \cdots \land F_n : x_n \land \neg(G_1 : \beta_1) \land \cdots \land \neg(G_m : \beta_m) \) in \( grd(P) \). Suppose \( I \) satisfies the body of \( C' \). In other words, it is the case that \( I(F_i) \in x_i \) for all \( 1 \leq i \leq n \) and \( I(G_j) \notin \beta_j \) for all \( 1 \leq j \leq m \). But then by Definition 17, the clause \( C \equiv F : x \leftarrow F_1 : x_1 \land \cdots \land F_n : x_n \) is in \( fpm(P, S) \). By the pre-fixpoint theorem, the family \( \mathcal{F}(lfp(T_{fpm(P, S)})) \), and hence \( S \), is a family of models for \( fpm(P, S) \). Therefore, \( I \) must satisfy the clause \( C \). But since \( I(F_i) \in x_i \) for all \( 1 \leq i \leq n \), it must be the case that \( I \) satisfies the head of \( C \), i.e. \( I(F) \in x \). Hence, \( I \) satisfies \( C' \).

5.2. **Relationship with Stable Formula Functions**

Thus far, we have introduced the notion of stable families of models for a gp-program \( P \). We have also proved that every interpretation in the stable family is a model for \( P \). A stable family taken as a whole can be
considered to make guesses on the probability ranges of basic formulas. Then a natural question to ask is the relationship between the stable families of models and the stable formula functions, which also make guesses on probability ranges. In this section we show that these two concepts are indeed closely related to each other. The following lemmas are crucial in establishing the relationship between the two notions.

**Lemma 4.** Let $P$ be a gp-program and $h$ be a formula function. Then: the ff-transform of $P$ based on $h$ contains the fpm-transform of $P$ based on the family $I(h)$ of interpretations associated with $h$, i.e. $fpm(P, I(h)) \subseteq ff(P, h)$.

**Proof.** Consider any clause $C \equiv F : A \leftarrow F_1 : x_1 \land \cdots \land F_n : x_n$ in $fpm(P, I(h))$. Then by Definition 17, there exists an interpretation $I \in I(h)$ and a clause $C' \equiv F : A \leftarrow F_1 : x_1 \land \cdots \land F_n : x_n \land \neg (G_1 : \beta_1) \land \cdots \land \neg (G_m : \beta_m)$ in $\text{grd}(P)$ such that $I(G_j) \notin \beta_j$ for all $1 \leq j \leq m$. Recall from Definition 10 that the linear program $\mathcal{L}'(h)$ associated with $h$ contains constraints of the form

$$c \leq \left( \sum_{W_j \in G \text{ and } W_j \in 2^L} p_j \right) \leq d,$$

where $h(G) = [c, d]$, for all $G \in hf(B_L)$. Also recall from Definition 10 that the world probability density function from which $I$ is generated is a solution to $\mathcal{L}'(h)$. Therefore, it is necessary that for all $G \in hf(B_L)$, $c \leq I(G) \leq d$ or $I(G) \in [c, d] = h(G)$. But since $I(G_j) \notin \beta_j$ and $I(G_j) \in h(G_j)$ for all $1 \leq j \leq m$, it must be the case that $h(G_j) \notin \beta_j$ for all $1 \leq j \leq m$. Hence according to Definition 15, the clause $C$ must also be in $I(h)$, $h(G_j) \notin \beta_j$ for all $1 \leq j \leq m$. This completes the proof that $fpm(P, I(h))$ must be a subset of $ff(P, h)$.

The above lemma states that the ff-transform of a gp-program based on a formula function $h$ is a superset of the fpm-transform based on the family $I(h)$ of interpretations. While the converse of the lemma is not true for any arbitrary formula function, the converse is true for all formula functions that are tight, a notion defined as follows.

**Definition 20.** A formula function $h$ is tight it for all basic formulas $F \in hf(B_L)$, $h(F) = \{ I(F) \mid I \in I(h) \}$.

Lemma 6 below, which is useful in establishing later results, provides a sufficient condition for a formula function to be tight. We need the following lemma to prove Lemma 6.

**Lemma 5.** Suppose $P$ is a gp-program and $h$ is a formula function. Then the solution set for $\mathcal{L}'(S'_p(h))$ is the same as that for $\mathcal{L}(T_p(h))$; i.e., $\mathcal{W}(S'_p(h)) = \mathcal{W}(T_p(h))$.
Proof. Extension of the proof of a similar lemma for pf-programs (25).

**Lemma 6.** Suppose there is a gp-program P such that h is fixpoint of $T'_p$, i.e., $h = T'_p(h)$. Then h is tight.

**Proof.** Suppose $T'_p(h) = h$. We need to show that for all $F \in h/B_L$, $h(F) = \{ I(F) | I \in \mathcal{I}(T'_p(h)) \}$.

(i) **Claim.** For all $F \in h/B_L$, $T'_p(h)(F)$ contains $\{ I(F) | I \in \mathcal{I}(T'_p(h)) \}$.

**Case 1.** $\mathcal{S}'_p(h)$ does not have solutions.

Then by Definition 14, $T'_p(h)(F) = \emptyset$. Thus, according to Definition 10 and Lemma 5, it is necessary that $\mathcal{W}(T'_p(h))$ and hence $\mathcal{I}(T'_p(h))$ be empty. Then it must be the case that $\{ I(F) | I \in \mathcal{I}(T'_p(h)) \} = \emptyset$. Hence, it follows that $\{ I(F) | I \in \mathcal{I}(T'_p(h)) \} \subseteq T'_p(h)(F)$.

**Case 2.** $\mathcal{S}'_p(h)$ has solutions.

Consider any $WP \in \mathcal{W}(T'_p(h)) = \mathcal{W}(T'_p(h))$. Let $WP(W_i) = p$, for all $W_i \in B_L$. Since $WP$ is a solution of $\mathcal{L}'_p(h)$, then for all $F \in h/B_L$, $WP$ satisfies every constraint $C_F$ of the form $c_F \leq (\sum_{W_j \in F} w_j + \sum_{W_j \in B_L} p_j) \leq d_F$, where $T'_p(h)(F) = [c_F, d_F]$. It follows that $(\sum_{W_j \in F} w_j + \sum_{W_j \in B_L} p_j) = I(F)$, where I is the interpretation associated with WP. Thus $I(F) \in [c_F, d_F] = T'_p(h)(F)$. Furthermore, this membership relationship is true for all $WP \in \mathcal{W}(T'_p(h))$ and correspondingly all $I \in \mathcal{I}(T'_p(h))$. Hence, it follows that $\{ I(F) | I \in \mathcal{I}(T'_p(h)) \} \subseteq T'_p(h)(F)$. This completes the proof of case 2 and claim (i).

(ii) **Claim.** Let $P$ be a gp-program. Then for all $F \in h/B_L$, $T'_p(h)(F) = \{ I(F) | I \in \mathcal{I}(T'_p(h)) \}$.

**Case 1.** $\mathcal{S}'_p(h)$ does not have solutions.

Then by Definition 14, $T'_p(h)(F) = \emptyset$. Thus, according to Definition 10, it is necessary that $\mathcal{W}(T'_p(h))$ and thus $\mathcal{I}(T'_p(h))$ be empty. Then it must be the case that $\{ I(F) | I \in \mathcal{I}(T'_p(h)) \} = \emptyset$. Hence, it follows that $\{ I(F) | I \in \mathcal{I}(T'_p(h)) \} = T'_p(h)(F)$.

**Case 2.** $\mathcal{S}'_p(h)$ has solutions.

Suppose $T'_p(h)(F) \neq \{ I(F) | I \in \mathcal{I}(T'_p(h)) \}$. Then by claim (i) above, it follows that $\{ I(F) | I \in \mathcal{I}(T'_p(h)) \} \subseteq T'_p(h)(F)$. But since $\mathcal{S}'_p(h)$ has solutions, it follows from Definition 14 that

$$T'_p(h)(F) = \min_{\mathcal{S}'_p(h)} \left( \sum_{W_j \in F \text{ and } W_j \in B_L} p_j \right),$$

$$\max_{\mathcal{S}'_p(h)} \left( \sum_{W_j \in F \text{ and } W_j \in B_L} p_j \right).$$
However, since \( \mathcal{WP}(S'_{p}(h)) = \mathcal{WP}(T'_{p}(h)) \) according to Lemma 5 \((\sum w_{j} \models F \text{ and } w_{j} \in 2^{\eta_{L}, p}) = I(F)\). It follows that \( \min_{\mathcal{WP}(S_{p}(h))}(\sum w_{j} \text{ and } w_{j} \in 2^{\eta_{L}, p}) = \min \{ I(F) \mid I \in \mathcal{I}(T'_{p}(h)) \} \) and \( \max_{\mathcal{WP}(S_{p}(h))}(\sum w_{j} \models F \text{ and } w_{j} \in 2^{\eta_{L}, p}) = \max \{ I(F) \mid I \in \mathcal{I}(T'_{p}(h)) \} \). Therefore, since \( \{ I(F) \mid I \in \mathcal{I}(T'_{p}(h)) \} \subset T'_{p}(h)(F) \), it must be the case that \( I(F) \models I \in \mathcal{I}(T'_{p}(h)) \quad \text{or} \quad \min \{ I(F) \mid I \in \mathcal{I}(T'_{p}(h)) \} \). But as proved in Chap. 7 of (29) the solution region of a linear program is convex, and thus \( \{ I(F) \mid I \in \mathcal{I}(T'_{p}(h)) \} \) forms an interval. The above strict subset relationship is only possible if either \( \min \{ I(F) \mid I \in \mathcal{I}(T'_{p}(h)) \} \neq \max \{ I(F) \mid I \in \mathcal{I}(T'_{p}(h)) \} \) or \( \max \{ I(F) \mid I \in \mathcal{I}(T'_{p}(h)) \} \neq \max \{ I(F) \mid I \in \mathcal{I}(T'_{p}(h)) \} \), clearly a contradiction. This completes the proof of case 2 and claim (ii).

Now since \( h \) is a fixpoint of \( T'_{p} \), it is necessary that \( h = T'_{p}(h) \). Then according to claim (ii) above, it follows that for all basic formulas \( F \in \text{bf}(B_{L}) \), \( h(F) = T'_{p}(h)(F) = \{ I(F) \mid I \in \mathcal{I}(T'_{p}(h)) \} = \{ I(F) \mid I \in \mathcal{I}(h) \} \). Hence, \( h \) is tight.

In Lemma 4 we show that the \( \mathcal{FP} \)-transform of a gp-program based on a formula function \( h \) is a superset of the \( \text{fpm} \)-transform based on the family \( \mathcal{I}(h) \) of interpretations. The following lemma shows that the converse is true for tight formula functions.

**Lemma 7.** Let \( P \) be a gp-program and \( h \) be a tight formula function. Then the \( \mathcal{FP} \)-transform of \( P \) based on \( h \) is a subset of the \( \text{fpm} \)-transform of \( P \) based on the family \( \mathcal{I}(h) \) of interpretations associated with \( h \); i.e., \( \mathcal{FP}(P, h) \subseteq \text{fpm}(P, \mathcal{I}(h)) \).

**Proof.** Consider any clause \( C \equiv F_{1} : \ldots : F_{n} : \alpha_{1} \land \cdots \land F_{n} : \alpha_{n} \in \mathcal{FP}(P, h) \). Then by Definition 15, there is a clause \( C' \equiv F_{1} : \ldots : F_{n} : \alpha_{1} \land \cdots \land F_{n} : \alpha_{n} \land \neg(G_{1} : \beta_{1}) \land \cdots \land \neg(G_{m} : \beta_{m}) \in \text{grd}(P) \) such that \( h(G_{j}) \not\equiv \beta_{j} \) for all \( 1 \leq j \leq m \). Thus, for all \( 1 \leq j \leq m \), there exists a point \( \gamma_{j} \in h(G_{j}) \) such that \( \gamma_{j} \neq \beta_{j} \). Now consider the solution set of the linear program \( \mathcal{LP}(h) \), as defined in Definition 10. Recall that every solution is a world probability density function which is extended to an interpretation. Therefore, for all interpretations \( I \in \mathcal{I}(h) \), it is necessary that for all \( 1 \leq j \leq m \). Now by Definition 4, since \( h \) is tight, \( h(G) = \{ I(G) \mid I \in \mathcal{I}(h) \} \) for all basic formulas \( G \in \text{bf}(B_{L}) \). But since the solution region of a linear program is convex (cf. Chap. 7 of (29)), every point inside the solution region is a solution. Then since \( \gamma_{j} \in h(G_{j}) \) for all \( 1 \leq j \leq m \), there must exist an interpretation \( I \in \mathcal{I}(h) \) such that \( I(G_{j}) = \gamma_{j} \) for all \( 1 \leq j \leq m \). Therefore, it is the case that \( I(G_{j}) \neq \beta_{j} \) for all \( 1 \leq j \leq m \). Then by Definition 17, it is necessary that the clause \( C \) must be in \( \text{fpm}(P, \mathcal{I}(h)) \).

**Corollary 1.** Let \( P \) be a gp-program and \( h \) be a tight formula function. Then the \( \mathcal{FP} \)-transform of \( P \) based on \( h \) is identical to the \( \text{fpm} \)-transform of
P based on the family $\mathcal{F}(h)$ of interpretations associated with $h$; i.e., $\text{ff}(P, h) = \text{fpm}(P, \mathcal{F}(h))$.

**Proof.** An immediate consequence of Lemmas 4 and 7.

The above corollary is important in proving the following theorem which establishes the strong correspondence between stable formula functions and stable families of models.

**Theorem 3 (Duality Theorem).** (1) Given a stable formula function $h$ with respect to a gp-program $P$, there exists a stable family $S$ of models for $P$ such that $S = \mathcal{F}(h)$.

(2) Given a stable family $S$ of models for $P$, there exists a stable formula function $h$ with respect to $P$ such that $\mathcal{F}(h) = S$.

**Proof.** (1) Given a stable formula function $h$, pick $S = \mathcal{F}(\text{lfp}(T_{\text{ff}(P, h)}))$. Since $h$ is a stable formula function, $h = \text{lfp}(T_{\text{ff}(P, h)})$. Therefore, it follows that $S = \mathcal{F}(h)$. Now consider the family $\mathcal{F}(\text{lfp}(T_{\text{fpm}(P, S)}))$. Since $S = \mathcal{F}(h)$, it is obvious that $\mathcal{F}(\text{lfp}(T_{\text{fpm}(P, S)})) = \mathcal{F}(\text{lfp}(T_{\text{fpm}(P, \mathcal{F}(h)}))).$ But then according to Theorem 1, since $h$ is stable, $h$ is a minimal fixpoint of $T'$. Thus by Lemma 6, it is necessary that $h$ be tight. Therefore, according to Corollary 1, it follows that $\mathcal{F}(\text{lfp}(T_{\text{fpm}(P, S)})) = \mathcal{F}(\text{lfp}(T_{\text{ff}(P, h)})) = S$. Hence, $S$ is a stable family of models for $P$.

(2) Given a stable family $S$ of models for $P$, pick $h = \text{lfp}(T_{\text{fpm}(P, S)})$. Since $S$ is a stable family, $S = \mathcal{F}(\text{lfp}(T_{\text{fpm}(P, S)}))$. Therefore, it follows that $\mathcal{F}(h) = S$. Now since $h$ is the least fixpoint of $T_{\text{fpm}(P, S)}$, then according to Lemma 6, $h$ is tight. Therefore, by Corollary 1, it is necessary that $\text{lfp}(T_{\text{ff}(P, h)}) = \text{lfp}(T_{\text{fpm}(P, \mathcal{F}(h))})$. But since $\mathcal{F}(h) = S$, it follows that $\text{lfp}(T_{\text{ff}(P, h)}) = \text{lfp}(T_{\text{fpm}(P, S)}) = h$. Hence, $h$ is a stable formula function with respect to $P$.

The above theorem establishes a duality between stable formula functions and stable families of models. The correspondence between the two notions of stability enhances our understanding of the properties of the two concepts. For instance, throughout the discussion of stable formula functions in the previous section, we have not studied whether the family of interpretations associated with a stable formula function is a family of models for the gp-program. This question is answered by the following corollary.

**Corollary 2.** Let $h$ be a stable formula function with respect to a gp-program $P$. Then the family $\mathcal{F}(h)$ of interpretations is a family of models for $P$. 
Proof. By Theorem 3, there exists a stable family $S$ of models for $P$, such that $\mathcal{I}(h) = S$. Moreover, by Theorem 2, it is necessary that every member (interpretation) in $S$, and hence in $\mathcal{I}(h)$, be a model for $P$. 

The above corollary applies our knowledge of stable families to enhance our understanding of stable formula functions. On the other hand, the following lemma applies our knowledge of stable formula functions to the understanding of stable families.

**Definition 21.** A stable family $S$ of (probabilistic) models for gp-program $P$ is maximal if there does not exist another stable family $S'$ of (probabilistic) models such that $S' \supsetneq S$.

**Lemma 8.** A stable family $S$ of models for gp-program $P$ is maximal.

**Proof.** By Theorem 3, there exists a stable formula function $h$ such that $S = \mathcal{I}(h)$. By Theorem 1, it follows that $h$ is a minimal fixpoint of $T_P$. Suppose there exists another stable family $S'$ of models for $P$ such that $S' \supsetneq S$. Then again by Theorem 3, there exists a stable formula function $h'$ such that $S' = \mathcal{I}(h')$. And by Theorem 1, $h'$ is also a minimal fixpoint of $T_P$. Now since $S' \supsetneq S$, it must be the case that $\mathcal{I}(h') \supsetneq \mathcal{I}(h)$ and $h' \neq h$. Thus, by Definition 10, the solution set to the set $\mathcal{L}^\in\mathcal{E}(h')$ of linear constraints strictly contains the solution set to $\mathcal{L}^\in\mathcal{E}(h)$. But by Definition 10, this is only possible if $h(G) \leq h'(G)$ for all $G \in \text{bf}(B_L)$. In other words, it is the case that $h' \preceq h$, violating the minimality of $h$.

In brief, we have introduced in this section the notion of stable families of models for a gp-program $P$. We have shown that each member in a stable family is a model for $P$. In the second half of this section, we have proved the duality between stable families and stable formula functions introduced in the previous section. As we have shown above, this strong correspondence between the fixpoint theory and model theory for gp-programs enhances our understanding of the two notions of stability.

### 6. Stable Classes of Formula Functions

We have studied how stable formula functions and stable families of models provide fixpoint and model-theoretic semantics for gp-programs. One weakness of these approaches is that a gp-program does not necessarily have a stable formula function and/or a stable family of models (cf. Example 11). In logic programming without probabilities, many attempts have been made to define a semantics for all programs. The most important technique developed thus far is the well-founded semantics.
for logic programs due to van Gelder et al. (32). In (4) Baral and Subrahmanian propose a stable and extension class theory for logic programs and default logics. Subsequently, in (5), Baral and Subrahmanian proved that well-founded semantics coincides precisely with stable classes that are minimal with respect to Hoare's power-domain ordering. In this section, we develop the theory of stable classes for probabilistic logic programming. We demonstrate that by selecting different stable classes, we obtain alternative semantics that encompass all probabilistic logic programs. The semantics obtained by considering Hoare-minimal stable classes appears to be the natural analog of well-founded semantics (32) for probabilistic logic programming.

**Definition 22.** Let \( P \) be a gp-program, and \( SF \) be a set of formula functions. Then \( SF \) is a stable class of formula functions with respect to \( P \) iff \( SF = \{ \text{lfp}(T_{ff(p, h_j)}) | h_j \in SF \} \).

Intuitively, a formula function \( h_j \) in a stable class is the same as the least fixpoint of an operator associated with the \( ff \)-transform of \( P \) based on some member \( h_j \) in the stable class; i.e., \( h_j = \text{lfp}(T_{ff(p, h_j)}) \). In general, every member in the class is related in the same way with some other member in the class. See (4) for more details on stable class theory. In short, a stable class of formula functions with respect to a gp-program represents a set of guesses on the probability ranges assigned by the program to basic formulas.

**Example 15.** Consider the gp-program in Example 11 again. A stable class of the program consists of the two formula functions \( h_1(p) = [0.95, 1] \) and \( h_2(p) = [0, 1] \). It is easy to check that \( h_1 = \text{lfp}(T_{ff(p, h_1)}) \) and \( h_2 = \text{lfp}(T_{ff(p, h_1)}) \).

The following lemma is straightforward to prove.

**Lemma 9.** A formula function \( h \) is a stable formula function with respect to the gp-program \( P \) iff the singleton set \( \{ h \} \) is a stable class with respect to \( P \).

The lemma above shows that the notion of a stable formula function is a special case of that of a stable class. The aim of the remainder of this section is to prove that every gp-program has a non-empty stable class of formula functions (cf. Theorem 5). But first we define an operator that will be proved in Lemma 10 to be anti-monotonic.

**Definition 23.** Let \( P \) be a gp-program and \( h \) a formula function. Define the operator \( \mathcal{S} \mathcal{F}_p : \mathcal{F} \mathcal{F} \to \mathcal{F} \mathcal{F} \) as \( \mathcal{S} \mathcal{F}_p(h) = \text{lfp}(T_{ff(p, h)}) \).
Lemma 10. The operator $\mathcal{F}_p$ is anti-monotonic; i.e., given formula functions $h_1 \leq h_2$, $\mathcal{F}_p(h_2) \leq \mathcal{F}_p(h_1)$.

Proof. Since $h_1 \leq h_2$, it is true that $h_1(F) \supseteq h_2(F)$ for all $F \in \text{hf}(B_L)$. Consider every clause $C$ in $\text{grd}(P)$; i.e., $C \equiv F : \alpha \leftarrow F_1 : \alpha_1 \wedge \cdots \wedge F_n : \alpha_n \wedge \neg(G_1 : \beta_1) \wedge \cdots \wedge \neg(G_m : \beta_m)$. For all $1 \leq j \leq m$, if $h_2(G_j) \not\subseteq \beta_j$, it is also true that $h_1(G_j) \not\subseteq \beta_j$. Hence, every clause in $\text{ff}(P, h_1)$ is in $\text{ff}(P, h_2)$. To simplify notation, abbreviate $T_{\text{ff}(P, h_1)}$ and $T_{\text{ff}(P, h_2)}$ by $T_1$ and $T_2$, respectively.

Claim. $T_2 \uparrow \gamma \leq T_1 \uparrow \gamma$ for all $\gamma$.

Proceed by transfinite induction on $\gamma$.

(i) Base case: $\gamma = 0$. For all $F \in \text{hf}(B_L)$, it follows that $T_1 \uparrow 0(F) = [0, 1] \subseteq [0, 1] = T_2 \uparrow 0(F)$.

(ii) Inductive case:

Case 1. $\gamma$ is a successor ordinal. Abbreviate $T_1 \uparrow (\gamma - 1)$ and $T_2 \uparrow (\gamma - 1)$ by $g_1$ and $g_2$, respectively. Also abbreviate $S_{\text{ff}(P, h_1)}$ and $S_{\text{ff}(P, h_1)}$ by $S_1$ and $S_2$, respectively. For any basic formula $F \in \text{hf}(B_L)$, $S_2(g_2)(F) = \bigcap M^2_F$, where $M^2_F = \{\alpha \mid C \equiv F : \alpha \leftarrow F_1 : \alpha_1 \wedge \cdots \wedge F_n : \alpha_n \text{ is a ground instance of a clause in } \text{ff}(P, h_2)\}$, and for all $1 \leq i \leq n$, $g_2(F_i) \subseteq \alpha_i$. There, it is necessary that $M^2_F \subseteq M^1_F$.

Case 1.1. $M^2_F$ is not empty. Now for each clause $C$ qualified in $M^2_F$ (i.e., $\alpha \in M^2_F$), $C$ is also a clause in $\text{ff}(P, h_1)$. Moreover, by the induction hypothesis, it is true that $g_2 \subseteq g_1$; i.e., $g_2(F) \supseteq g_1(F)$ for all $F \in \text{hf}(B_L)$. Thus, for all $1 \leq i \leq n$, it is necessary that $g_1(F_i) \subseteq \alpha_i$. Hence, $C$ qualifies in $M^1_F$, where $S_1(g_1)(F) = \bigcap M^1_F$, where $M^1_F = \{\alpha \mid C \equiv F : \alpha \leftarrow F_1 : \alpha_1 \wedge \cdots \wedge F_n : \alpha_n \text{ is a ground instance of a clause in } \text{ff}(P, h_1)\}$, and for all $1 \leq i \leq n$, $g_1(F_i) \subseteq \alpha_i$. Therefore, it is necessary that $M^2_F \subseteq M^1_F$.

Case 1.2. $M^2_F$ is empty. Then it is obvious that $M^2_F \subseteq M^1_F$.

Thus, combining cases 1.1 and 1.2 above, it is necessary that $S_1(g_1)(F) = \bigcap M^1_F \subseteq \bigcap M^2_F = S_2(g_2)(F)$. Since this subset relation is true for all basic formulas $F$, then by Lemma 2, it must be the case that $T_1(g_1)(F) \subseteq T_2(g_2)(F)$ for all basic formulas $F$. In other words, it is true that $T_2 \uparrow \gamma \leq T_1 \uparrow \gamma$. This completes the proof for the case when $\gamma$ is a successor ordinal.

Case 2. $\gamma$ is a limit ordinal. However, $T_1 \uparrow \gamma(F) = \bigcup \{T_1 \uparrow \alpha(F) \mid \alpha < \gamma\}$ and $T_2 \uparrow \gamma(F) = \bigcup \{T_2 \uparrow \alpha(F) \mid \alpha < \gamma\}$ for all basic formulae $F \in \text{hf}(B_L)$. But by the induction hypothesis, it is the case that $T_2 \uparrow \alpha(F) \supseteq T_1 \uparrow \alpha(F)$ for all $\alpha < \gamma$. Hence, it is necessary that $T_2 \uparrow \gamma(F) \supseteq T_1 \uparrow \gamma(F)$ for all basic formulas $F$. Therefore, it must be the case that $T_2 \uparrow \gamma \leq T_1 \uparrow \gamma$. This completes the proof for the transfinite induction. Then according to the claim, it is necessary that $\text{lf}(T_2) \leq \text{lf}(T_1)$, and thus $\mathcal{F}_p(h_2) \leq \mathcal{F}_p(h_1)$.
Theorem 4 (Yablo (33)). Suppose \( \Phi \) is an anti-monotone operator on a complete lattice \((S, \leq)\). Let \( \text{lfp}(\Phi^2) \) and \( \text{gfp}(\Phi^2) \) denote the least and greatest fixed-points, respectively, of \( \Phi^2 \). Then \( \Phi(\text{lfp}(\Phi^2)) = \text{gfp}(\Phi^2) \) and \( \Phi(\text{gfp}(\Phi^2)) = \text{lfp}(\Phi^2) \).

The following result is now an immediate consequence of the preceding result and the anti-monotonicity of \( \mathcal{F}_p \).

Theorem 5. Every gp-program has a non-empty stable class of formula functions.

The theorem above states that every gp-program has a non-empty stable class of formula functions. Suppose \( C_1, C_2 \) are two sets of formula functions. Recall that the ordering \( \leq \) applies to formula functions. We extend this ordering now to sets of formula functions (and hence to stable classes) in two ways. Both these orderings are well known orderings on algebraic structures called power domains due to Hoare and Smyth (30).

Definition 24. Let \( S_1, S_2 \) be sets of formula functions.

1. We denote that \( S_1 \leq_{\text{smyth}} S_2 \) iff \( (\forall s_1 \in S_1)(\exists s_2 \in S_2) s_1 \leq s_2 \), and \( S_1 \leq_{\text{hoare}} S_2 \) iff \( (\forall s_2 \in S_2)(\exists s_1 \in S_1) s_1 \leq s_2 \).

2. A non-empty stable class \( C \) is said to be Hoare-minimal iff:
   
   (a) \( C \) is inclusion-minimal, i.e. there is no non-empty stable class \( C' \) such that \( C' \subseteq C \), and

   (b) for every inclusion-minimal non-empty stable class \( C' \) either \( C \leq_{\text{hoare}} C' \) or \( C \not\leq_{\text{hoare}} C' \) and \( C' \not\leq_{\text{hoare}} C \).

3. \( C \) is said to be Smyth-minimal iff conditions (a) and (b) above hold with \( \leq_{\text{hoare}} \) replaced by \( \leq_{\text{smyth}} \).

We may choose either Hoare-minimal stable classes or Smyth-minimal stable classes as the intended meaning of our program. However, depending on the choice we make, we may get differing semantics, as shown in the following example.

Example 16. Suppose we consider the gp-program

\[
p : [1, 1] \leftarrow a : [1, 1]
\]

\[
p : [1, 1] \leftarrow b : [1, 1]
\]

\[
a : [1, 1] \leftarrow \neg(b : [1, 1])
\]

\[
b : [1, 1] \leftarrow \neg(a : [1, 1]).
\]
This program has two stable formula functions:

$h_1$ which assigns $[1, 1]$ to both $p$ and $a$

$h_2$ which assigns $[1, 1]$ to both $p$ and $b$.

Furthermore, suppose $h_1$ is the function that assigns $[1, 1]$ to all of $p, a, b$ and $h_4$ is the function that assigns $[0, 1]$ to all of $p, a, b$. Then the set \{h_3, h_4\} is a stable class of formula functions.

In this example, \{h_1\} and \{h_2\} are both Smyth-minimal stable classes of formula functions. Hence, the Smyth-minimal stable class semantics assigns $[1, 1]$ to $p$.

However, \{h_3, h_4\} is the unique Hoare-minimal stable class of formula functions. This Hoare-minimal class would only allow us to conclude that $p$ gets the value $[0, 1]$.

In this section we have studied the notion of a stable class of formula functions. As shown in Lemma 9, it generalizes the notion of a stable formula function. The advantage of such a notion is that it gives a semantics for all gp-programs. In future work, we investigate whether this notion can be used as a basis for defining well-founded semantics for probabilistic logic programming.

7. Stability of Probabilistic Models

In previous sections, we have introduced the notions of stable formula functions and stable families of models. We have studied their properties and examined the relationship between the two concepts. In both cases, they can be regarded as making guesses on the probability ranges within which basic formulas must lie. Since probability ranges are not directly probabilistic truth values (point probabilities) for basic formulas in our framework, a natural question to ask is whether we can directly guess probabilistic truth values of basic formulas. Thus, in this section we study another notion of stability—stable (probabilistic) models (not to be confused with the stability of a family of probabilistic models, as discussed in previous sections). Then we investigate the relationship between this notion of stability and the ones we presented in previous sections.

7.1. Stable Probabilistic Models

In classical logic programming, the stable semantical approach proposed by Gelfond and Lifschitz (14) can be viewed as a two-step process: guessing a Herbrand interpretation and then testing whether this guess satisfies the defined stability criterion. For general probabilistic logic
programs, an analogous notion of stability is then to extend this original idea by guessing probabilities of basic formulas and verifying whether the guess satisfies some criterion of stability. This process is formalized as follows.

**Definition 25.** Let a *probability assignment* \( I \) be a mapping from all basic formulas to \([0, 1]\); i.e., \( I : bf(B_L) \rightarrow [0, 1] \).

Intuitively, a probability assignment \( I \) is a purely arbitrary guess on probabilities of basic formulas. For instance, it is valid for \( I \) to assign a probability 0.5 to some \( p \in bf(B_L) \) while assigning 0 to \((p \lor q)\), even though such an assignment may violate general properties of probability, such as the ones identified by Fenstad (11). Note that probability assignments are not necessarily derived from world probability density functions and hence they are not the same as interpretations. It is up to the definition of stability (cf. Definition 27 below) to guarantee that when a probability assignment \( I \) satisfies the stated criterion, \( I \) is indeed a valid probabilistic interpretation and actually a model for the gp-program (cf. Theorem 6 below).

**Definition 26.** Let \( P \) be a gp-program and \( I \) be a probability assignment. Define the *probabilistic-model transform* of \( P \) based on \( I \) (*pm-transform* for short), denoted by \( pm(P, I) \), as follows:

1. Given a clause \( C' \equiv F : \alpha \leftarrow F_1 : \alpha_1 \land \cdots \land F_n : \alpha_n \land \neg (G_1 : \beta_1) \land \cdots \land \neg (G_m : \beta_m) \) in \( grd(P) \), if \( I(G_j) \notin \beta_j \) for all \( 1 \leq j \leq m \), then the clause \( C \equiv F : \alpha \leftarrow F_1 : \alpha_1 \land \cdots \land F_n : \alpha_n \) is in \( pm(P, I) \).

2. Nothing else is in \( pm(P, I) \).

It is obvious that clauses in \( pm(P, I) \) are negation-free. Hence, \( pm(P, I) \) is a pf-program, and it is therefore legitimate to use the fixpoint operator \( T_{pm(P, I)} \) as defined in Definition 13.

**Definition 27.** A probability assignment \( I \) is called a *stable (probabilistic) model* for the gp-program \( P \) if \( I \) is contained in the family of models associated with the least fixpoint of \( T_{pm(P, I)} \); i.e., \( I \in \mathcal{A}(lfp(T_{pm(P, I)})) \).

It is obvious from the above definition that when a probability assignment \( I \) is contained in \( \mathcal{A}(lfp(T_{pm(P, I)})) \), the membership in the family guarantees that \( I \) must be an interpretation.

**Example 17.** Again consider the gp-program \( P \) that consists of the single clause

\[
p : [0.95, 1] \leftarrow \neg (q : [0.49, 0.51]).
\]
Suppose $I_1$ is the probability assignment that assigns the probabilistic truth value 1 to $p$ and the probabilistic truth value 0 to $q$. Then, $pm(P, I_1)$ consists of the clause 

$$p : [0.95, 1] \leftarrow.$$ 

Then the least fixpoint $lfp(T_{pm(P, I_1)})$ assigns the range $[0.95, 1]$ to $p$ and the range $[0, 1]$ to $q$. Obviously, $I_1$ obeys those ranges and is contained in $\mathcal{J}(lfp(T_{pm(P, I_1)}))$. Thus, $I_1$ is stable.

On the other hand, suppose $I_2$ is the probability assignment that assigns 0 to $p$ and 0.5 to $q$. Then, $pm(P, I_2)$ is empty, and the least fixpoint $lfp(T_{pm(P, I_2)})$ assigns $[0, 1]$ to both $p$ and $q$. Thus, $I_2$ is clearly contained in $\mathcal{J}(lfp(T_{pm(P, I_2)}))$ and is stable also.

Now consider the probability assignment $I_3$ that assigns 0 to both $p$ and $q$. Then, $pm(P, I_3)$ is the same as $pm(P, I_1)$, and the least fixpoint $lfp(T_{pm(P, I_3)})$ assigns the range $[0.95, 1]$ to $p$ and the range $[0, 1]$ to $q$. Thus, since $I_3(p) = 0 \notin [0.95, 1]$, $I_3$ is not contained in $\mathcal{J}(lfp(T_{pm(P, I_3)}))$, and is therefore not stable.

The following theorem shows that whenever a probability assignment (or an interpretation) $I$ satisfies the stability criterion stated in Definition 27, $I$ is a model for the gp-program. Hence, it is valid to call $I$ a stable model.

**Theorem 6.** Let $P$ be a gp-program, and $I$ an interpretation such that $I$ is contained in the family of models associated with the least fixpoint of $T_{pm(P, I_1)}$, i.e., $I \in \mathcal{J}(lfp(T_{pm(P, I_1)}))$. Then $I$ is a model for $P$.

**Proof.** Consider any clause $C' \equiv F : \alpha \leftarrow F_1 : \alpha_1 \land \cdots \land F_n : \alpha_n \land \neg (G_1 : \beta_1) \land \cdots \land \neg (G_m : \beta_m)$ in $grd(P)$. Suppose $I$ satisfies the body of $C'$. In other words, it is the case that $I(F_i) \in \alpha$, for all $1 \leq i \leq n$ and $I(G_j) \notin \beta_j$ for all $1 \leq j \leq m$. But then by Definition 26, the clause $C \equiv F : \alpha \leftarrow F_1 : \alpha_1 \land \cdots \land F_n : \alpha_n$ must be in $pm(P, I)$. Now according to the pre-fixpoint theorem, the family $\mathcal{J}(lfp(T_{pm(P, I_1)}))$ is a family of models for $pm(P, I)$. Since $I$ is contained in $\mathcal{J}(lfp(T_{pm(P, I_1)}))$, $I$ is a model for $pm(P, I)$. Therefore, it is necessary that $I$ satisfies the clause $C$ above. But since $I(F_i) \in \alpha$, for all $1 \leq i \leq n$, it must be the case that $I$ satisfies the head of $C$; i.e., $I(F) \in \alpha$. Hence, $I$ satisfies $C'$.  

Intuitively, a stable model makes guesses on the probabilistic truth values of basic formulas. As an interpretation is an extension of a world probability density function which defines a probability distribution on the set of worlds it also makes sense to guess the probabilistic truth values of basic formulas by guessing a world probability density function. However, by Lemma 1, there is a strong correspondence between a world probability density function and an interpretation. Therefore, it suffices to
study stable models, without worrying about an analogous notion of
stability for world probability density functions.

7.2. Relationship with Stable Formula Functions

Thus far, we have introduced three notions of stability: stable formula
functions, stable families of models, and stable models. As shown in
Theorem 3, we have identified the duality between stable formula functions
and stable families of models. In this section we relate the concept of a
stable model to the other two notions of stability.

Recall from Theorem 2 that a stable family of models for a gp-program
$P$ is a family of models for $P$. But according to Theorem 6, every stable
model is also a model for $P$. The following questions now come to mind.

1. Can a stable model be a member of some stable family of models?

2. If so, is it necessary that a stable family of models must be
contained in the set of all stable models, or vice versa?

The answers to these questions tell us which notion of stability is
stronger than the other. In the following we address these questions one by
one.

Example 18. Consider the gp-program described in Examples 9 and 17.
Recall from Example 9 that the formula function $h_1$ that assigns $[0.95, 1]$ to $p$ and $[0, 1]$ to $q$ is stable. Then according to Theorem 3, $\mathcal{A}(h_1)$ is a
stable family of models. But as shown in Example 17, the probabilistic
interpretation $I_1$ that assigns the probabilistic truth value 1 to $p$ and 0 to
$q$ is a stable model. It is also obvious that $I_1 \in \mathcal{A}(h_1)$. Thus, $I_1$, which is a
stable model, is also a member of a stable family of models.

The example answers the first question posed above, and this answer
leads us to the second question. Consider the following example.

Example 19. Consider again the gp-program described in Examples 9 and 17. Recall from Example 9 that the only stable formula function $h_1$
with respect to that program is the one that assigns $[0.95, 1]$ to $p$ and
$[0, 1]$ to $q$. But according to Example 17, the interpretation $I_2$ that assigns
0 to $p$ and 0.5 to $q$ is a stable model. It is, however, easy to check that
$I_2$ is not contained in $\mathcal{A}(h_1)$. Hence, $I_2$, which is a stable model, is not a
member of any stable family of models.

This example shows that the set of all models need not be a subset of any
stable family of models. To study whether the converse is true, we have the
following theorem.

Theorem 7. Every model $I$ of a gp-program $P$ is a stable model of $P$. 
Proof. Let $I$ be a model of $P$.

Claim. $I$ is a model for $pm(P, I)$.

Consider any clause $C = F: x \leftarrow F_1 : x_1 \wedge \cdots \wedge F_n : x_n$ in $pm(P, I)$. Suppose $I$ satisfies the body of $C$. In other words, it is necessary that $I(F_i) \in x_i$ for all $1 \leq i \leq n$. But since $C$ is a clause in $pm(P, I)$, then according to Definition 26, there must exist some clause $C' = F: x \leftarrow F_1 : x_1 \wedge \cdots \wedge F_n : x_n \wedge \neg (G_1 : \beta_1) \wedge \cdots \wedge \neg (G_m : \beta_m)$ in $grd(P)$, where $m \geq 0$. Moreover, it must be the case that $I(G_j) \notin \beta_j$ for all $1 \leq j \leq m$. But then, since $I$ is a model of $P$, $I$ must be a model of $C'$. Therefore, since $I(F_i) \in x_i$ for all $1 \leq i \leq n$ and $I(G_j) \notin \beta_j$ for all $1 \leq j \leq m$, it is necessary that $I(F) \in x$. Hence, $I$ satisfies $C$. This completes the proof of the claim.

Thus from the pre-fixpoint theorem and its consequences, the family $\mathcal{F}(lfp(T_{pm(P, I)}))$ contains all the models for $pm(P, I)$. Therefore, $I$ is contained in $\mathcal{F}(lfp(T_{pm(P, I)}))$. Hence, it follows that $I$ is stable. 

Corollary 3. $I$ is a stable model for $P$ iff $I$ is a model for $P$.

Proof. An immediate consequence of Theorem 6 and Theorem 7.

On first sight the corollary seems astonishing. But consider the following example which highlights the difference between classical stable model semantics and the stable probabilistic model semantics introduced here.

Example 20. In the framework of classical logic programming, the program

\[ p \leftarrow \neg q \]

has two minimal models, $\{p\}$ and $\{q\}$. According to the stable model semantics (14), $\{p\}$ is the only stable model. Now consider the following gp-program $P$:

\[ p : [1, 1] \leftarrow \neg (q : [1, 1]). \]

Pick the probabilistic interpretation $I_1$ which assigns 1 to $p$ and 0 to $q$. It is easy to see that $I_1$ is a stable probabilistic model, either by Definition 27 or by Corollary 3. Furthermore, $I_1$ is equivalent to the classical stable model $\{p\}$, as the world probability density function associated with $I_1$ assigns a probability 1 to the world $\{p\}$ and 0 to all other worlds.

Now consider the probabilistic interpretation $I_2$ which assigns 0 to $p$ and 1 to $q$. Then according to Definition 26, $pm(P, I_2)$ is empty. Thus by Definitions 13 and 27, $I_2$ is a stable probabilistic model. Similar to the situation of $I_1$ above, $I_2$ is equivalent to the world $\{q\}$. Hence, the semantics based on single stable probabilistic models differs from the classical stable semantics. The difference is due to the fact that this semantics takes
the cautious approach of assigning \([0, 1]\) to a basic formula if nothing can be deduced about that formula. For example, when \(pm(P, I_2)\) is empty, the least fixpoint of \(T_{pm(P, I_2)}\) assigns \([0, 1]\) to all basic formulas. Thus, the family \(I(ffp(T_{pm(P, I_2)}))\) of probabilistic models contains all probabilistic models.

The following corollary shows that a stable family of models must be contained in the set of all (probabilistic) models.

**Corollary 4.** Let \(S\) be a family of models for \(P\). Then every member of \(S\) is a stable (probabilistic) model for \(P\).

**Proof.** An immediate consequence of Theorem 2 and Theorem 7.

Thus for gp-programs, the notion of stable probabilistic models, which is an extension to the original stable model semantics proposed by Gelfond and Lifschitz, seems to be too weak. We believe that for gp-programs, the notions of stable formula functions and stable families of models are more appropriate.

8. Discussion

Like many researchers, we are interested in the use of numerical estimates in default reasoning. Unfortunately the framework we propose in (25) for negation-free probabilistic logic programs is not powerful enough to handle default rules and exceptions. Consider the pf-program in the following example.

**Example 21.** Consider Benjy again from Example 4. Suppose we know that there is over 95% chance that a dog can bark. We also know that Benjy and Fido are dogs. However, Benjy is unable to bark (his vocal cords were injured at some point). This can be represented as

\[
\text{bark}(X) : [0.95, 1] \leftarrow \text{dog}(X) : [1, 1]
\]

\[
\text{dog}(\text{benjy}) : [1, 1] \leftarrow 
\]

\[
\text{bark}(\text{benjy}) : [0, 0] \leftarrow.
\]

Thus, the first clause represents the default rule about dogs and their abilities to bark. However, as represented in the last clause, Benjy is an exception. But according to the semantics for pf-programs, the probability range for \(\text{bark}(\text{benjy})\) is \([0.95, 1] \cap [0, 0] = \emptyset\), and thus there does not exist any model, indicating inconsistency.
In the research described in this paper, we take a first step towards handling default rules and exceptions in our framework. With the support of the non-monotonic negation $\neg$, we can now specify that a default rule is only applicable in the absence of evidence to the contrary. Consider the following gp-program.

**Example 22.** Continue with Example 21 above. The default rule about the ability of dogs to bark is now represented as follows:

\[
bark(X) : [0.95, 1] \leftarrow \text{dog}(X) : [1, 1] \land \neg (\text{abnormal}(X) : [1, 1]).
\]

And a dog is abnormal if it definitely cannot bark, as represented below:

\[
\text{abnormal}(X) : [1, 1] \leftarrow \bark(X) : [0, 0].^2
\]

In addition to the two clauses above, suppose the gp-program $P$ also contains the clause that indicates that Benjy is a dog:

\[
dog(benjy) : [1, 1] \leftarrow.
\]

Consider the formula function $h_1$ that assigns $[1, 1]$ to $\text{dog}(benjy)$, $[0.95, 1]$ to $\bark(benjy)$, but $[0, 1]$ to $\text{abnormal}(benjy)$. Then the ff-transform of $P$ based on $h_1$ consists of the following clauses:

\[
bark(benjy) : [0.95, 1] \leftarrow \text{dog}(benjy) : [1, 1]
\]

\[
\text{abnormal}(benjy) : [1, 1] \leftarrow \bark(benjy) : [0, 0]
\]

\[
dog(benjy) : [1, 1] \leftarrow.
\]

Thus, it is obvious to see that $h_1$ is stable.

Suppose $P$ is now updated with the fact that Benjy definitely cannot bark. Thus, the new gp-program becomes the one described in Example 4. As we have seen in Example 10, the unique stable formula function for that program handles the interaction between default rules and exceptions appropriately. And by Theorem 3, the program has models.

In addition to dealing with exceptions, this form of non-monotonic negation also handles interacting defaults, as shown in the following examples.

**Example 23.** In (28), Reiter and Crisuolo consider the following situation: (i) that John is a high school dropout, (ii) that high school dropouts are typically adults, and (iii) that adults are typically employed.

---

^2 As presented here, the predicate abnormal is not needed because there is only one source of abnormality—inca
cpability of barking. But we prefer to use the predicate, as in general there may be many sources of abnormality.
Due to the transitivity of default rules (ii) and (iii), the conclusion that John is employed can be deduced. They argue that this conclusion is undesirable.

Now consider the following gp-program $P_1$:

$\text{adult}(X) : [0.95, 1] \leftarrow \text{dropout}(X) : [1, 1]$

$\text{employed}(X) : [0.95, 1] \leftarrow \text{adult}(X) : [1, 1] \land \neg (\text{abnormal}(X) : [1, 1])$

$\text{abnormal}(X) : [1, 1] \leftarrow \text{dropout}(X) : [1, 1]$.

Suppose to begin with, $P_1$ only contains the fact that $\text{adult(john)} : [1, 1] \leftarrow$. Then it is easy to check that the only stable formula function with respect to $P_1$ assigns the range $[0.95, 1]$ to $\text{employed(john)}$ correctly.

Suppose the following fact is added to $P_1$:

$\text{dropout(john)} : [1, 1] \leftarrow$.

Call this new program $P_2$. Consider the formula function $h$ that assigns $[1, 1]$ to $\text{adult(john)}, \text{dropout(john)},$ and $\text{abnormal(john)}$, but $[0, 1]$ to $\text{employed(john)}$. It is straightforward to check that $h$ is a unique stable formula function with respect to $P_2$.

Finally, consider the situation where the only known fact about John is that he is a high school dropout, i.e., deleting the fact about John's adulthood from program $P_2$. Call this new program $P_3$. The unique stable formula function with respect to $P_3$ is the one that assigns $[1, 1]$ to $\text{dropout(john)}$ and $\text{abnormal(john)}$, $[0.95, 1]$ to $\text{adult(john)}$, and $[0, 1]$ to $\text{employed(john)}$. Hence, undesirable conclusions due to transitivity of default rules are avoided.

The framework proposed by Dubois and Prade (9) also handles the situation discussed in the above example. However, their semantics is different from ours, as their framework is based on possibility logic and their model theory is based on fuzzy sets which are well-known to be non-probabilistic. The following example on interacting default rules has been discussed extensively, but see (27) for a probabilistic treatment on the subject.

**Example 24.** Consider the situation (i) that tweety is a penguin, (ii) that a penguin is a bird, (iii) that typically penguins cannot fly, and (iv) that birds can typically fly. The situation can be represented by the following gp-program:

$\text{fly}(X) : [0.95, 1] \leftarrow \text{bird}(X) : [1, 1] \land \neg (\text{abnBird}(X) : [1, 1])$

$\text{fly}(X) : [0, 0.05] \leftarrow \text{penguin}(X) : [1, 1] \land \neg (\text{abnPenguin}(X) : [1, 1])$
bird(X) : [1, 1] ← penguin(X) : [1, 1]

abnBird(X) : [1, 1] ← penguin(X) : [1, 1]
penguin(tweety) : [1, 1] ←

Consider the formula function \( h \) that assigns \([1, 1]\) to \( penguin(tweety) \), \( bird(tweety) \), and \( abnBird(tweety) \), \([0, 0.05]\) to \( fly(tweety) \), and \([0, 1]\) to \( abnPenguin(tweety) \). Again it is not difficult to show that \( h \) is the unique stable formula function with respect to the program.

Thus far, we have shown several examples on how to handle default reasoning in our framework. But when compared with the probabilistic frameworks proposed by Bacchus (2) and Buntine (8), our framework is not as expressive as theirs. For instance, given the above example, their frameworks can conclude that “birds typically are not penguins.” Such a conclusion is not deducible in our framework, and in ongoing research we are studying how to extend our theory to handle such cases. However, as the framework of Bacchus extends full first-order logic, it is unclear to us how his framework can be used as a basis for logic programming and deductive databases. Similar comments apply to Buntine’s proposal.

There have also been many proposals on multi-valued logic programming. These include the works by Baldwin (3), Blair and Subrahmanian (6, 7), Fitting (12, 13), Kifer et al. (17–19), Morishita (23), and van Emde (31). However, all of these works are non-probabilistic, and they are not concerned with the support of non-monotonic negation. See (24, 25) for a more detailed comparison between these and other formalisms and our framework for probabilistic logic programs without negation.

9. Conclusions

In this paper we study the semantics and some uses of probabilistic logic programs with non-monotonic negation (i.e., gp-programs). Based on the stable semantical approach for classical logic programming, we investigate three natural of stability: stable formula functions, stable family of probabilistic models, and stable probabilistic models. We show that stable formula functions are minimal fuxpoints of operators associated with programs. We also show that each member of a stable family is a probabilistic model of the program. Then we prove that stable formula functions and stable families behave as duals of each other, tying the fixpoint and model theories for gp-programs closely together. As for stable probabilistic models, even though they are straightforward extensions of the original stable models for classical logic programming, they are too
weak in the probabilistic framework. Finally, we provide a stable class semantics for gp-programs without stable formula functions or stable families of probabilistic models.

In ongoing research, we are studying how to support empirical probabilities in our framework. We are also interested in designing a proof procedure for gp-programs. In particular, we are investigating whether it suffices to augment the proof procedure we developed for positive probabilistic logic programs with some kind of negation as failure rule. Furthermore, we are investigating the possibility of developing probabilistic default theories and proving appropriate connections between probabilistic logic programming and probabilistic default reasoning. Such connections would be analogous to similar correspondences that exist between classical logic programming and default reasoning (22, 4).

ACKNOWLEDGMENTS

We are grateful to the referees for useful comments. VS was supported by NSF Grant IRI-91-09755, by the Office of Graduate Studies and Research of the University of Maryland, and by the Army Research Office under Grant DARL-03-92-G-0225. RN was supported under NSF grants IRI-8719458 and IRI-91-09755. RN also thanks Timos Sellis for this support.

Received January 22, 1991; final manuscript received December 16, 1991

REFERENCES