

3-1991

A Unified Framework For Three-Valued Semantical Treatments of Logic Programming

Feng Yang

Follow this and additional works at: http://surface.syr.edu/eecs_techreports



Part of the [Computer Sciences Commons](#)

Recommended Citation

Yang, Feng, "A Unified Framework For Three-Valued Semantical Treatments of Logic Programming" (1991). *Electrical Engineering and Computer Science Technical Reports*. Paper 118.
http://surface.syr.edu/eecs_techreports/118

This Report is brought to you for free and open access by the College of Engineering and Computer Science at SURFACE. It has been accepted for inclusion in Electrical Engineering and Computer Science Technical Reports by an authorized administrator of SURFACE. For more information, please contact surface@syr.edu.

SU-CIS-91-11

***A Unified Framework for
Three-Valued Semantical Treatments
of Logic Programming***

Feng Yang

March 1991

*School of Computer and Information Science
Syracuse University
Suite 4-116, Center for Science and Technology
Syracuse, New York 13244-4100*

A Unified Framework For Three-valued Semantical Treatments of Logic Programming

Feng Yang

School of Computer & Information Science
Syracuse University
Syracuse, N.Y. 13244-4100

Abstract

Based on Fitting's Φ operator a unified framework for three-valued semantics of logic programming is presented. The truth space used in the framework is the class of partial interpretations. Underlying the truth space is two partial orderings, *knowledge ordering* and *truth ordering*. It turns out that the truth space with the truth ordering is a complete lattice and the truth space with knowledge ordering is a semi-complete lattice. Φ is proved to be continuous over the complete lattice and monotonic over the semi-complete lattice. With the use of Φ operator two well-known three-valued semantics for logic programming, Fitting's three-valued semantics and well-founded semantics, are characterized within the framework in a simple and elegant way. We show that Fitting's semantics is the least stable three-valued model with respect to the *knowledge ordering* and well-founded semantics is the least stable three-valued model with respect to the *truth ordering*.

1 Introduction

There have been many recent advances in elucidating the semantics of negation in logic programming by using three-valued logic. Two best-known three-valued semantics are Fitting's three-valued semantics and well-founded semantics. Fitting has proposed using Kripke/Kleene three-valued semantics [2, 3]. Van Gelder *et al* has proposed *well-founded semantics* based on *unfounded set* [1]. Both approaches seem like good approximations to the intended semantics of negation in Prolog. There has been of lack of investigation of the relationship between those approaches. The goal of this paper is to present a framework which can accommodate those approaches so that they can be easily understood and compared to one another. The main contributions of this work are (1) that we introduce $F_{\mathcal{P}}$ *immediate failure operator* and prove that $\Phi_{\mathcal{P}}$ operator proposed by Fitting can be defined with the combination of $T_{\mathcal{P}}$ and $F_{\mathcal{P}}$, and (2) $\Phi_{\mathcal{P}}$ is actually continuous over the space of the partial interpretations of logic programs, and (3) we prove that there are two different stable three-valued semantics for logic programs with negation: one of them is the well-founded semantics and the other coincides with Fitting's three-valued semantics.

The framework consist of $\Phi_{\mathcal{P}}$ operator developed by Fitting [2] with the class of partial interpretations of logic programs. There are two important partial ordering underlying the the class of partial interpretations of logic programs. One of them is called *knowledge ordering* which represents an increase or decrease of knowledge; the other one is called *truth ordering* which represents an increase in 'truth' or decrease in 'falseness'. The class of partial interpretations with knowledge ordering forms a semi-lattice and is a complete lattice under truth ordering. We prove that $\Phi_{\mathcal{P}}$ is monotonic with respect to knowledge ordering and is continuous with respect to truth ordering. We show that Fitting's semantics is the least stable three-valued model of a general logic program with

respect to knowledge ordering and the well-founded model is the least stable three-valued model with respect to truth ordering.

As the base of the framework we also extend the definition of logic programs to allow explicitly use of the “undefined” truth constant in the bodies of program clauses. We prove that each partial logic program has two least three-valued models under the two orderings.

We begin with a few definitions and results concerning logic programming. In section 3 present the theory of partial logic programs. We show the relationship between Fitting’s semantics and well-founded semantics in section 4.

2 Background From Logic Programming and Relation to Other Work

2.1 Syntax and Classic Semantics of Logic Programs

A *general* logic program is a finite set of universally quantified clauses of the form

$$A \leftarrow \lambda_1, \lambda_2, \dots, \lambda_n$$

where $n \geq 0$, A is an atom and λ_i are literals. If all literals in all clauses are atoms, then the program is call *definite*.

By an *alphabet* $\Sigma_{\mathbf{P}}$ of a logic program \mathbf{P} we mean the set of *constants, predicate and function symbols* which occur in the program¹. Any alphabet is assumed to contain a countably infinite set of variable symbols, connective ($\neg, \vee, \wedge, \forall, \exists, \leftarrow$) and the usual punctuation symbols. The *language* of \mathbf{P} consists of all well-formed formulae constructed in the standard way. We refer the reader to [8, 11] for terminology and notation concerning *terms, atoms* and *literals* that are not otherwise presented in this paper.

In addition, we assume that any program clauses and goals do not contain any occurrence of the equality symbol.

The set of all ground terms of a logic program \mathbf{P} is called the *Herbrand universe*, denoted as $U_{\mathbf{P}}$ and the set of all ground atoms is called the *Herbrand base*, denoted as $B_{\mathbf{P}}$. We use \mathbf{P}^* to denote the ground instantiation of a logic program \mathbf{P} .

van Emden and Kowalski in [12] developed a closure operator, $T_{\mathbf{P}}$, for a definite logic program \mathbf{P} to deduct positive information from the program. The definition of $T_{\mathbf{P}}$ is as follows. Consider a definite logic program \mathbf{P} . Let $I \subseteq B_{\mathbf{P}}$.

$$T_{\mathbf{P}}(I) = \{A | \exists A \leftarrow B_1, B_2, \dots, B_n \in P^*\{B_1, B_2, \dots, B_n\} \subseteq I\}. \quad (1)$$

They show that the operator $T_{\mathbf{P}}$ is upward continuous over $\mathbb{L}_{\mathbf{P}} = \langle 2^{B_{\mathbf{P}}}, \subseteq \rangle$ and the least fixedpoint of $T_{\mathbf{P}}$ is precisely the set of all ground atoms which are logical consequences of the program.

2.2 Duality in Logic Programming

Yang in [13] developed a dual of $T_{\mathbf{P}}$, denoted as $F_{\mathbf{P}}$ for a definite logic program to deduct negative information from the program. Its definition is as follows.

$$F_{\mathbf{P}}(I) = \{A | \forall A \leftarrow B_1, B_2, \dots, B_n \in P^*\{B_1, B_2, \dots, B_n\} \cap I \neq \emptyset\}. \quad (2)$$

He shows that $F_{\mathbf{P}}$ has the properties entirely analogous to those desirable properties of $T_{\mathbf{P}}$. It is downward continuous over $\overline{\mathbb{L}}_{\mathbf{P}} = \langle 2^{B_{\mathbf{P}}}, \supseteq \rangle$ which is a dual of $\mathbb{L}_{\mathbf{P}}$. Its relationship with $T_{\mathbf{P}}$ is characterized by the following equation

¹If \mathbf{P} does not contain any constants, then one is added to the alphabet

$$T_{\mathbf{P}}(I) = B_{\mathbf{P}} - F_{\mathbf{P}}(B_{\mathbf{P}} - I). \quad (3)$$

The fixedpoints of $F_{\mathbf{P}}$ are exactly the complements of the fixedpoints of $T_{\mathbf{P}}$.
One can extended the definitions of $T_{\mathbf{P}}$ and $F_{\mathbf{P}}$ to cope with negation in a natural way.

$$\begin{aligned} T_{\mathbf{P}}(\langle T, F \rangle) = & \{A | \exists A \leftarrow B_1, B_2, \dots, B_m, \neg B_{m+1}, \dots, \neg B_n \in P^* \\ & \{B_1, B_2, \dots, B_m\} \subseteq T \text{ and } \{B_{m+1}, \dots, B_n\} \subseteq F\} \end{aligned} \quad (4)$$

$$\begin{aligned} F_{\mathbf{P}}(\langle T, F \rangle) = & \{A | \forall A \leftarrow B_1, B_2, \dots, B_n \in P^* \\ & \{B_1, B_2, \dots, B_m\} \cap F \neq \emptyset \text{ or } \{B_{m+1}, \dots, B_n\} \cap T \neq \emptyset\}. \end{aligned} \quad (5)$$

However, as we will see in section 3 if we extend the definitions of $T_{\mathbf{P}}$ and $F_{\mathbf{P}}$ to three-valued semantics $F_{\mathbf{P}}$ is generally not a dual of $T_{\mathbf{P}}$ any more.

2.3 Fitting's Three-valued Semantics

Fitting have proposed a three-valued model semantics for general logic program based on *Kleene's* three-valued logic. The space of truth space underlying the semantics is the collection of partial interpretations of a logic program \mathbf{P} . He associates each general logic program with a monotone operator, denoted as $\Phi_{\mathbf{P}}$, on the space. Let's use a pair of disjoint set $I = \langle T, F \rangle$ to denote a partial interpretation of a logic program \mathbf{P} , where T and F are subsets of Herbrand base of \mathbf{P} . The set T contains all ground atoms true in I , the set F contains all ground atoms false in I and the truth value of the remaining atoms in $B_{\mathbf{P}} - (T \cup F)$ is undefined. $\Phi_{\mathbf{P}}$ is defined by Fitting [2] as follows.

$\Phi_{\mathbf{P}}(\langle T, F \rangle) = \langle T', F' \rangle$ where

- $A \in T'$ if there is a clause in \mathbf{P}^* has conclusion A and a premise that I makes true;
- $A \in F'$ if for every clause in \mathbf{P}^* having conclusion A have a premise that I makes false.

Where a formula is evaluated according to *Kleene's* three-valued logic.

It is clear that for all partial interpretation I

$$\Phi_{\mathbf{P}}(I) = \langle T_{\mathbf{P}}(I), F_{\mathbf{P}}(I) \rangle. \quad (6)$$

He proved $\Phi_{\mathbf{P}}$ is monotonic under knowledge ordering and hence has a least fixpoint in the space which is taken to be the intended meaning of program \mathbf{P} .

Latter Fitting, in a series of papers [4, 5, 6], has argued that one can makes good operational and denotational sense of logic programming with a Heyting algebra as the space of truth values. The truth value spaces essentially have the structure of M. Ginsberg's bilattices. For our propose we work with a concretely defined subspace of the the space of truth values. We show that Φ operator provides a nice framework in which many notions connected with three-valued logic can be treated uniformly, and provides a common generalization of the various logic programming extensions.

3 Partial Logic Programs

In this section we present a natural extension of logic programs by allowing explicitly the use of *undefined* truth constant, denoted as \mathbf{U} , as an atomic formula in the bodies of program clauses. We extend various definitions and results to cater for the *undefined* element. We will call such logic programs *partial* logic programs. This extension itself may not be of any significance to the theory of logic programming. But it serve as a base for the framework of three-valued semantics of logic programming. First we start with the syntax of a partial logic program.

Definition 3.1 A *partial logic program* is a finite set of program clause of the form

$$A \leftarrow \lambda_1, \lambda_2, \dots, \lambda_n$$

where $n \geq 0$, A is an atom and each λ_i is either an atom or the undefined constant U .

Recall that we are always working with a fixed language in first order logic.

Definition 3.2 An *partial interpretation* I of a partial logic program consists of a universe \mathbb{U} and an assignment σ of an n -ary function $f : \mathbb{U}^n \rightarrow \mathbb{U}$ to each n -ary function symbol f , an n -ary predicate symbol (other than U) $p : \mathbb{U}^n \rightarrow \{\text{true}, \text{false}, \text{undefined}\}$ to each n -ary predicate symbol p , and $\sigma(U) = \text{undefined}$.

For convenience we introduce an abbreviation: $\hat{B}_{\mathbf{P}} = B_{\mathbf{P}} - \{U\}$.

We find it simplifies thing to use the following as the definition of a partial interpretation of a partial logic program.

Definition 3.3 Let \mathbf{P} be a program. $I = \langle T, F \rangle$ is a partial interpretation of \mathbf{P} if (1) $T, F \subseteq \hat{B}_{\mathbf{P}}$; and (2) $T \cap F = \emptyset$. I is called a *saturated interpretation* of \mathbf{P} if $T \cup F = \hat{B}_{\mathbf{P}}$.

We will use $\mathbb{I}_{\mathbf{P}}$ to denote the set of all interpretations of \mathbf{P} .

Informally speaking, partial interpretation $I = \langle T, F \rangle$ is a three-valued *Herbrand interpretation* which precisely maps atoms in T to “true”, atoms in F to “false”, and otherwise to “undefined”. We will find it convenient to abbreviate “partial interpretation” to “interpretation”.

The following definition is from Fitting’s [5]. We restrict the truth space to the set of partial interpretations of a partial logic program for our propose.

Definition 3.4 Consider a partial logic program \mathbf{P} . Let $I_1 = \langle T_1, F_1 \rangle, I_2 = \langle T_2, F_2 \rangle$ be two interpretations of \mathbf{P} . Assume that $\mathcal{I}_{\mathbf{P}}$ is the set of index of $\mathbb{I}_{\mathbf{P}}$.

- We say that $I_1 \preceq_k I_2$ if

$$T_1 \subseteq T_2 \text{ and } F_1 \subseteq F_2.$$

If $I_s = \langle T_s, F_s \rangle$, for $s \in S \subseteq \mathcal{I}_{\mathbf{P}}$, are interpretations, then by their *k-meet* we mean the interpretation

$$\prod_{s \in S}^k I_s = \langle \bigcap_{s \in S} T_s, \bigcap_{s \in S} F_s \rangle.$$

Similarly, by *k-join* of those interpretations we means the interpretation

$$\prod_{s \in S}^k I_s = \langle \bigcup_{s \in S} T_s, \bigcup_{s \in S} F_s \rangle.$$

- We say that $I_1 \preceq_t I_2$ if

$$T_1 \subseteq T_2 \text{ and } F_1 \supseteq F_2.$$

If $I_s = \langle T_s, F_s \rangle$, for $s \in S \subseteq \mathcal{I}_{\mathbf{P}}$, are interpretations, then by their *t-meet* we mean the interpretation

$$\prod_{s \in S}^t I_s = \langle \bigcap_{s \in S} T_s, \bigcup_{s \in S} F_s \rangle.$$

Similarly, by *t-join* of those interpretations we means the interpretation

$$\prod_{s \in S}^t I_s = \langle \bigcup_{s \in S} T_s, \bigcap_{s \in S} F_s \rangle.$$

The partial ordering \preceq_t is called *truth ordering* because it represents an increase in ‘truth’ or a decrease in ‘falseness’; The partial ordering \preceq_k is called *knowledge ordering* because it represents an increase in knowledge.

Both orderings are suitable partial ordering for the truth space \mathbb{I}_P .

Proposition 3.1 *Let P be a partial logic program. Then $\mathcal{K}_P = \langle \mathbb{I}_P, \preceq_k \rangle$ is a semi-complete lattice and $\mathcal{T}_P = \langle \mathbb{I}_P, \preceq_t \rangle$ is a complete lattice*

We will use $\llbracket lfp \rrbracket_k(T)$ and $\llbracket gfp \rrbracket_k(T)$ to denote the least and the greatest fixedpoint of T over \mathcal{K}_P respectively; and we use $\llbracket lfp \rrbracket_t(T)$ and $\llbracket gfp \rrbracket_t(T)$ to denote the least fixedpoint and the greatest fixedpoint of T over \mathcal{T}_P .

The semantics we give partial logic programs will base on Kleene’s three-valued logic. The truth tables for the connective \wedge and \neg are as follows:

\wedge				\neg	
\wedge	T	F	U	P	$\neg P$
T	T	F	U	T	F
F	F	F	F	F	T
U	U	F	U	U	U

Tables for the other propositional connectives can be found in [9].

Definition 3.5 *Let P be a partial logic program, and $I = \langle T, F \rangle$ be an interpretation of P . I is a partial model of P if every clause in P is evaluated to true under I .*

3.1 The least fixedpoints of partial logic programs

We associate a ‘‘meaning’’ with a partial logic program in two different ways, corresponding to the two partial ordering over \mathbb{I}_P . First we introduce a mapping Φ_P .

Definition 3.6 *Let P be a partial logic program and $I = \langle T, F \rangle \in \mathbb{I}_P$. Then we define*

$$\Phi_P(\langle T, F \rangle) = \langle T_P(T), F_P(F) \rangle .$$

where T_P and F_P is defined as in (1) and (2) except the program is a partial logic program.

The definition immediately yields the following proposition since the explicit use of **U** will not voids the continuity of T_P and F_P .

Proposition 3.2 *Let P be a partial logic program. Then Φ_P is monotonic over \mathcal{K}_P , and continuous over \mathcal{T}_P .*

Theorem 3.1 *Let P be a logic program. Then*

$$\llbracket lfp \rrbracket_k(\Phi_P) = \langle lfp(T_P), gfp(F_P) \rangle = \langle T_P \uparrow \omega, \hat{B}_P - gfp(T_P) \rangle .$$

From Tarski Theorem in [8] we get the following result.

Theorem 3.2 *Let P be a logic program. Then*

$$\llbracket lfp \rrbracket_t(\Phi_P) = \langle lfp(T_P), lfp(F_P) \rangle = \langle T_P \uparrow \omega, F_P \uparrow \omega \rangle$$

and

$$\llbracket gfp \rrbracket_t(\Phi_P) = \langle gfp(T_P), \hat{B}_P - gfp(T_P) \rangle .$$

Remark Notice that the least model of a partial logic program may not be saturated. The following example illustrates that.

Example Let P be a partial logic program as follows.

$$A \leftarrow U.$$

The least model $\llbracket lfp \rrbracket_t(\Phi_P) = \llbracket lfp \rrbracket_k(\Phi_P) = \langle \emptyset, \emptyset \rangle$.

From the above results one can see that both $\llbracket lfp \rrbracket_k(\Phi_P)$ and $\llbracket lfp \rrbracket_t(\Phi_P)$ agree on the truth part, but differ on the false part in general.

4 Three-valued Stable Models of General Logic Programs

Since Gelfond and Lifschitz in [7] work on *stable model* there have been advances in extension of stable model semantics to three-valued stable model semantics of logic programs. In this section we will propose a new three-valued stable model semantics for logic programs which is shown to coincide with Fitting's semantics. Before we present the definition of three-valued stable models we recall the notion of *general logic programs*.

Definition 4.1 A general logic program is a finite set of program clauses of the form:

$$A \leftarrow l_1, l_2, \dots, l_n$$

where each l_i is a literal.

The following definition is due to Przymusinski [10].

Definition 4.2 Consider a general logic program P and $I \in \mathbb{I}_P$. Then Π_P^I is the partial logic program obtained from P^* by

1. deleting each program clause that has a literal $\neg A$ in its body such that $A \in T$.
2. deleting all negated atom $\neg A$ in the bodies of remaining program clauses such that $A \in F$.
3. replacing all negated atoms with U in the bodies of remaining program clauses.

From the definition above we know that Π_P^I is a partial logic program. According to results in section 3 it has two different fixedpoint semantics, which suggest us that there is a need to define two different notions of stable models.

Definition 4.3 Consider a general logic program P . If $I = \llbracket lfp \rrbracket_k(\Phi_{\Pi_P^I})$, then I is called a *k-stable model* of P ; If $I = \llbracket lfp \rrbracket_t(\Phi_{\Pi_P^I})$, then I is called a *t-stable model* of P .

The following is an immediate consequence of the definition of the transformation. Its easy proof is omit here.

Theorem 4.1 Consider a general logic program P . Let $I = \langle T_I, F_I \rangle, J = \langle T_J, F_J \rangle \in \mathbb{I}_P$. Then we have that

$$\Phi_{\Pi_P^{\langle T_I, F_I \rangle}}(\langle T_J, F_J \rangle) = \langle T_P(\langle T_J, F_I \rangle), \mathcal{F}_P(\langle T_I, F_J \rangle) \rangle.$$

Definition 4.4 Consider a general logic program P . Let $I \in \mathbb{I}_P$. We define

$$\Psi_P^k(I) = \llbracket lfp \rrbracket_k(\Phi_{\Pi_P^I})$$

and

$$\Psi_P^t(I) = \llbracket lfp \rrbracket_t(\Phi_{\Pi_P^I})$$

We use \mathbb{W}_P to denote the well-founded model of a general logic program P and \mathbb{F}_P to denote the least fixedpoint of Φ_P throughout this section.

Theorem 4.2 *Let P be a general logic program. Then operators Ψ_P^k is monotonic with respect to knowledge ordering.*

Proof: Let $I = \langle T_I, F_I \rangle, J = \langle T_J, F_J \rangle, I' = \langle T_{I'}, F_{I'} \rangle, J' = \langle T_{J'}, F_{J'} \rangle \in \mathbb{I}_P$ and suppose that $I \preceq_k J$ and $I' \preceq_k J'$. By Theorem 4.1 it follows that

$$\Phi_{\Pi_P^I}(\langle T_{I'}, F_{I'} \rangle) = \langle \mathcal{T}_P(\langle T_{I'}, F_{I'} \rangle), \mathcal{F}_P(\langle T_I, F_{I'} \rangle) \rangle$$

and

$$\Phi_{\Pi_P^J}(\langle T_{J'}, F_{J'} \rangle) = \langle \mathcal{T}_P(\langle T_{J'}, F_{J'} \rangle), \mathcal{F}_P(\langle T_J, F_{J'} \rangle) \rangle.$$

Hence we have

$$\Phi_{\Pi_P^I}(\langle T_{I'}, F_{I'} \rangle) \preceq_k \Phi_{\Pi_P^J}(\langle T_{J'}, F_{J'} \rangle) \quad (7)$$

It is easy to prove by using transfinite induction that

$$[[lfp]]_k(\Phi_{\Pi_P^I}) \preceq_k [[lfp]]_k(\Phi_{\Pi_P^J})$$

which completes our proof. \square

Theorem 4.3 (Przymusinski 1989) *Let P be a general logic program. Then Ψ_P^t are monotonic with respect to knowledge ordering.*

Theorem 4.4 (Przymusinski 1989) *Let P be a general logic program. Then*

$$[[lfp]]_k(\Psi_P^t) = \mathbb{W}_P = glb\{I : I = [[lfp]]_t(\Phi_{\Pi_P^I})\}$$

Theorem 4.4 implies that every general logic program P has the smallest stable three-valued model with respect to truth ordering. Moreover, this model always coincides with the well-founded model of P .

We now are in a position to establish results similar to 4.4. First we have following lemma.

Lemma 4.1 *Let P be a general logic program and $I \in \mathbb{I}_P$. then*

$$\Phi_P(I) = \Phi_{\Pi_P^I}(I).$$

Proof: Suppose $I = \langle T, F \rangle \in \mathbb{I}_P$ and we have following equations:

$$\langle T_1, F_1 \rangle = \Phi_P(\langle T, F \rangle)$$

$$\langle T_2, F_2 \rangle = \Phi_{\Pi_P^{\langle T, F \rangle}}(\langle T, F \rangle).$$

We have to prove that $T_1 = T_2$ and $F_1 = F_2$.

From the equation (6) we know

$$\begin{aligned} T_1 = & \{A | \exists A \leftarrow B_1, B_2, \dots, B_m, \neg B_{m+1}, \dots, \neg B_n \in P^* \\ & \{B_1, B_2, \dots, B_m\} \subseteq T \text{ and } \{B_{m+1}, \dots, B_n\} \subseteq F\} \end{aligned} \quad (8)$$

and

$$\begin{aligned} F_1 = & \{A | \forall A \leftarrow B_1, B_2, \dots, B_m, \neg B_{m+1}, \dots, \neg B_n \in P^* \\ & \{B_1, B_2, \dots, B_m\} \cap F \neq \emptyset \text{ or } \{B_{m+1}, \dots, B_n\} \cap F \neq \emptyset\} \end{aligned} \quad (9)$$

Similarly from Definition 4.2 we have

$$T_2 = \{A | \exists A \leftarrow B'_1, B'_2, \dots, B'_{n'} \in \Pi_{\mathbf{P}}^{\langle T, F \rangle} \{B'_1, B'_2, \dots, B'_{n'}\} \subseteq T\} \quad (10)$$

and

$$F_2 = \{A | \forall A \leftarrow B'_1, B'_2, \dots, B'_{n'} \in \Pi_{\mathbf{P}}^{\langle T, F \rangle} \{B'_1, B'_2, \dots, B'_{n'}\} \cap F \neq \emptyset\}. \quad (11)$$

(a): $T_1 = T_2$.

Assume that $A \in T_1$. Then by (8) there is a ground clause in P^* of the form

$$A \leftarrow B_1, B_2, \dots, B_m, \neg B_{m+1}, \dots, \neg B_n$$

such that

$$\{B_1, B_2, \dots, B_m\} \subseteq T$$

and

$$\{B_{m+1}, \dots, B_n\} \subseteq F$$

By Definition 4.2 there is a ground clause in $\Pi_{\mathbf{P}}^{\langle T, F \rangle}$ of the form

$$A \leftarrow B_1, B_2, \dots, B_m$$

such that

$$\{B_1, B_2, \dots, B_m\} \subseteq T.$$

Hence we have that $A \in T_2$.

Now we assume that there is a ground clause in $\Pi_{\mathbf{P}}^{\langle T, F \rangle}$ of the form

$$A \leftarrow B_1, B_2, \dots, B_m \in \Pi_{\mathbf{P}}^{\langle T, F \rangle}$$

such that

$$\{B_1, B_2, \dots, B_m\} \subseteq T.$$

By Definition 4.2 there must be a ground clause in P^* of the form

$$A \leftarrow B_1, B_2, \dots, B_m, \neg B_{m+1}, \dots, \neg B_{n'}$$

such that

$$\{B_{m+1}, \dots, B_{n'}\} \subseteq F$$

and

$$\{B_1, B_2, \dots, B_m\} \subseteq T.$$

By (10) we have that $A \in T_1$. Hence $T_1 = T_2$.

(b): $F_1 = F_2$.

Assume that $A \in F_1$. Then by (9) for all ground clauses in P^* of the form

$$A \leftarrow B_1, B_2, \dots, B_m, \neg B_{m+1}, \dots, \neg B_n$$

such that either

$$\{B_1, B_2, \dots, B_m\} \cap F \neq \emptyset$$

or

$$\{B_{m+1}, \dots, B_n\} \cap T \neq \emptyset$$

By Definition 4.2 all ground clauses whose head is A in $\Pi_{\mathbf{P}}^{\langle T, F \rangle}$ (if any) are of the form

$$A \leftarrow B_1, B_2, \dots, B_m$$

or

$$A \leftarrow B_1, B_2, \dots, B_m, \mathbf{U}$$

Furthermore the following condition is satisfied:

$$\{B_1, B_2, \dots, B_m\} \cap F \neq \emptyset.$$

Hence we have that $A \in F_2$. Now suppose that for all ground clauses in $\Pi_{\mathbf{P}}^{\langle T, F \rangle}$ of the form

$$A \leftarrow B_1, B_2, \dots, B_m$$

or

$$A \leftarrow B_1, B_2, \dots, B_m, \mathbf{U}$$

the following holds:

$$\{B_1, B_2, \dots, B_m\} \cap F \neq \emptyset.$$

Let $\Omega_{\mathbf{P}}(A)$ be the set of all ground clauses of the program \mathbf{P} whose head is A . By Definition 4.2 we have

$$\Omega_{\mathbf{P}}(A) = \eta_{\mathbf{P}}(A) \cup \delta_{\mathbf{P}}(A)$$

where $\delta_{\mathbf{P}}(A)$ is a subset of P^* and each clause in $\delta_{\mathbf{P}}(A)$ is form:

$$A \leftarrow B_1, B_2, \dots, B_m, \neg B_{m+1}, \dots, B_{n'}$$

such that

$$\{B_{m+1}, \dots, B_{n'}\} \cap T \neq \emptyset$$

and $\eta_{\mathbf{P}}(A)$ is the set of all ground clauses of of \mathbf{P} of the form

$$A \leftarrow B_1, B_2, \dots, B_m, \neg B_{m+1}, \dots, \neg B_n$$

such that

$$\{B_1, \dots, B_m\} \cap F \neq \emptyset.$$

By (11) we have that $A \in F_1$. Hence $F_1 = F_2$. □

Theorem 4.5 *Let P be a general logic program. Then*

$$\llbracket lfp \rrbracket_k(\Psi_{\mathbf{P}}^k) = \mathbb{F}_{\mathbf{P}} = glb\{I : I = \llbracket lfp \rrbracket_k(\Phi_{\Pi_{\mathbf{P}}^I})\}$$

Proof: By the definition of $\mathbb{F}_{\mathbf{P}}$ we know that $\mathbb{F}_{\mathbf{P}} = \Phi_{\mathbf{P}}(\mathbb{F}_{\mathbf{P}})$ and for all $I \in \mathbb{I}_{\mathbf{P}}$

$$I = \Phi_{\mathbf{P}}(I) \implies \mathbb{F}_{\mathbf{P}} \preceq_k I.$$

Let $\mathbb{S}_{\mathbf{P}} = \llbracket lfp \rrbracket_k(\Psi_{\mathbf{P}}^k)$. The by Definition 4.4 we have that $\mathbb{S}_{\mathbf{P}} = \Psi_{\mathbf{P}}^k(\mathbb{S}_{\mathbf{P}})$ and for all $I \in \mathbb{I}_{\mathbf{P}}$

$$I = \Psi_{\mathbf{P}}^k(I) \implies \mathbb{S}_{\mathbf{P}} \preceq_k I.$$

(a): $\mathbb{F}_{\mathbf{P}} \preceq_k \mathbb{S}_{\mathbf{P}}$.

Since $\mathbb{S}_{\mathbf{P}}$ is a fixedpoint of $\Phi_{\Pi_{\mathbf{P}}^{\mathbb{S}_{\mathbf{P}}}}$ we have

$$\begin{aligned}\mathfrak{S}_{\mathbf{P}} &= \Phi_{\Pi_{\mathfrak{S}_{\mathbf{P}}}}(\mathfrak{S}_{\mathbf{P}}) \\ &= \Phi_{\mathbf{P}}(\mathfrak{S}_{\mathbf{P}}) \quad \text{By Lemma 4.1}\end{aligned}$$

So $\mathfrak{S}_{\mathbf{P}}$ is also a fixedpoint of $\Phi_{\mathbf{P}}$. Since $\mathbb{F}_{\mathbf{P}}$ is the least fixedpoint of $\Phi_{\mathbf{P}}$ we have $\mathbb{F}_{\mathbf{P}} \preceq_k \mathfrak{S}_{\mathbf{P}}$.

(b): $\mathfrak{S}_{\mathbf{P}} \preceq_k \mathbb{F}_{\mathbf{P}}$.

By the definition of $\mathbb{F}_{\mathbf{P}}$ we have that

$$\begin{aligned}\mathbb{F}_{\mathbf{P}} &= \Phi_{\mathbf{P}}(\mathbb{F}_{\mathbf{P}}) \\ &= \Phi_{\Pi_{\mathbb{F}_{\mathbf{P}}}}(\mathbb{F}_{\mathbf{P}}) \quad \text{By Lemma 4.1}\end{aligned}$$

Thus $\mathbb{F}_{\mathbf{P}}$ is a fixedpoint of $\Phi_{\Pi_{\mathbb{F}_{\mathbf{P}}}}$. It follows that

$$\Psi_{\mathbf{P}}^k(\mathbb{F}_{\mathbf{P}}) = \llbracket lfp \rrbracket_k(\Phi_{\Pi_{\mathbb{F}_{\mathbf{P}}}}) \preceq_k \mathbb{F}_{\mathbf{P}}$$

By Tarski Theorem we have that $\mathfrak{S}_{\mathbf{P}} \preceq_k \mathbb{F}_{\mathbf{P}}$ which completes our proof. \square

Theorem 4.5 every general logic program \mathbf{P} has the smallest stable three-valued model with respect to knowledge ordering and the model always coincides with Fitting's three-valued model of \mathbf{P} .

Acknowledgements

I wish to thank Dr. Howard Blair and Dr. Allen L. Brown for helpful discussions and comments.

References

- [1] K. Ross A. Van Gelder and Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *Proceedings of the Symposium on Principles of Databases Systems*, 1988.
- [2] Melvin Fitting. A kripke-kleene semantics for logic programs. *Journal of Logic Programming*, 4:295–312, 1985.
- [3] Melvin Fitting. Partial models and logic programming. *Theoretical Computer Science*, 48:229–255, 1986.
- [4] Melvin Fitting. Logic programming on a topological bilattice. *Fundamenta Informaticae*, XI:209–218, 1988.
- [5] Melvin Fitting. Bilattices and the theory of truth. *Journal of Philosophical Logic*, 18:225–256, 1989.
- [6] Melvin Fitting. Bilattices in logic programming. In *Proceedings of the 12th International Symposium on Multiple-Valued Logic*, 1990.
- [7] M. Gelfond and V. Lifschitz. Stable semantics for logic programs. In *Proceedings of 5th International Symposium Conference on Logic Programming*, Seattle, 1988.
- [8] W.L. Lloyd. *Foundation of Logic programs*. Springer-Verlag, second edition, 1988.
- [9] J. Lukasiewicz. O logice trojwartosciowej. *Ruch Filozoficzny*, 5:169–170, 1920.
- [10] T. Przymusiński. Extended stable semantics for normal and disjunctive logic programs. draft manuscript, 1990.

- [11] J. Shoenfield. *Mathematical Logic*. addison-wesley, Reading, Mass., 1967.
- [12] M.H. van Emden and R.A. Kowalski. The semantics of predicate logic as a programming language. *JACM*, 23(4):733–742, oct. 1976.
- [13] Feng Yang. Duality in logic programming. Tech. Report CIS-91-10, School of Computer and Information Science, Syracuse University, 1991.