Theoretical Computer Science 105 (1992) 7–25
Elsevier

On the relations between stable and well-founded semantics of logic programs*

Phan Minh Dung

Institute of Computer Science, National Center for Scientific Research of Vietnam, Lieugiai, Badinh, Hanoi, Vietnam and Division of Computer Science, Asian Institute of Technology, Bangkok 10501, Thailand

Abstract

Dung, P.M., On the relations between stable and well-founded semantics of logic programs, Theoretical Computer Science 105 (1992) 7–25.

We study the relations between stable and well-founded semantics of logic programs.

(1) We show that stable semantics can be defined in the same way as well-founded semantics based on the basic notion of unfounded sets. Hence, stable semantics can be considered as "two-valued well-founded semantics".

(2) An axiomatic characterization of stable and well-founded semantics of logic programs is given by a new completion theory, called *strong completion*. Similar to the Clark's completion, the strong completion can be interpreted in either two-valued or three-valued logic. We show that

- Two-valued strong completion specifies the stable semantics.
- Three-valued strong completion specifies the well-founded semantics.

(3) We study the equivalence between stable semantics and well-founded semantics. At first, we prove the equivalence between the two semantics for strict programs. Then we introduce the bottom-stratified and top-strict condition generalizing both the stratifiability and the strictness, and show that the new condition is sufficient for the equivalence between stable and well-founded semantics. Further, we show that the call-consistency condition is sufficient for the existence of at least one stable model.

1. Introduction

There are many alternative approaches to the semantics of negation in logic programming. Kunen [14] distinguishes two main competing approaches: the logical

Correspondence to: Phan Minh Dung, Division of Computer Science, Asian Institute of Technology, G.P.O. Box 2754, Bangkok 10501, Thailand. Email: dung@ait.th.

* A shortened version of this paper appeared in the Proceedings of the Second International Conference on Algebraic and Logic Programming.

consequence approach based on program completion, and the canonical model approach which picks out some specific models of the program.

In the logical consequence approach, the semantics may be defined by the Clark's completion [4]. Given a logic program P, the completion of P, comp(P), consists of some equality axioms plus a completed definition of each predicate symbol. Roughly, this completed definition is obtained by replacing the "if" by "iff". The completion of a program can be interpreted either in the two-valued logic [4] or in a three-valued logic [11]. While the three-valued completion is always consistent, this is not the case for the two-valued completion. But if the program is call-consistent [15, 24], then two-valued completion is consistent, too. The three-valued semantics is weaker than the two-valued, in the sense that every query supported in the three-valued semantics is also supported in the two-valued semantics but not conversely. But if the program is strict then these two semantics are equivalent [15].

However, the Clark's completion does not always capture the intended meaning of the program. For example, let P consist of the single clause $p \leftarrow p$. Intuitively, we expect that any meaningful semantics of P would imply that p is false. But since the comp(P) is $p \leftrightarrow p$, we cannot conclude from comp(P) that p is false. To clarify once more the shortcomings of the Clark's completion, consider the following example taken from [21].

Example 1.1. Let P be

$$\text{edge}(a, b) \leftarrow$$

$$\text{edge}(c, d) \leftarrow$$

$$\text{reachable}(a) \leftarrow$$

$$\text{reachable}(x) \leftarrow \text{reachable}(y), \text{edge}(y, x)$$

$$\text{unreachable}(x) \leftarrow \neg\, \text{reachable}(x)$$

P can be illustrated by the following picture:

$$\ldots\ldots\ldots a \longrightarrow b$$

$$c \longrightarrow d$$

We obviously expect the vertices c, d to be unreachable; indeed, Clark's semantics implies unreachable(c), unreachable(d).

Now adding to P the clause edge(d, c) will result in a new program P' which is illustrated by the following picture:

$$\ldots\ldots\ldots a \longrightarrow b$$

$$c \longleftrightarrow d$$

The Clark's completion of P' could not imply that c, d are unreachable, although it still appears to be expected from the given information.

Two major semantics in the canonical model approach are the (two-valued) stable semantics and the (three-valued) well-founded semantics.

The stable semantics of a program is defined by the set of its stable models. This semantics has its root in nonmonotonic logics, where a logic program is considered as an autoepistemic theory whose stable extensions correspond to the stable models of the logic program [13]. In general, the stable semantics overcomes the drawbacks of the Clark's completion, e.g. the stable semantics of the program in Example 1.1 provides the expected conclusion. The problem of stable semantics is that it is not defined for every logic program, e.g. the program consisting of only the clause $p \leftarrow \neg p$ has no stable models. To illustrate the seriousness of this problem, let us consider one more example.

Example 1.2 (The Barber's paradox). "Beardland is a small city where the barber Noel shaves every citizen who does not shave himself.

Does Noel shave the city mayor Casanova?

Does Noel shave himself?"

The problem can be represented by a logic program consisting of the clauses

$$\text{shave}(\text{Noel}, x) \leftarrow \neg \text{shave}(x, x)$$

$$\text{mayor}(\text{Casanova}) \leftarrow$$

Despite the confusion about who shaves Noel, we expect that Noel shaves the city mayor Casanova. But this program has no stable model, i.e. we could not conclude anything with respect to the stable semantics.

The idea of well-founded semantics is negation as (possibly infinite) failure, i.e. the failure (possibly in infinitary) to prove a fact (a ground atom) to be true leads to the acceptance of this fact being false. Formally, the well-founded semantics is defined by the well-founded model, which is defined as the least fixpoint of a monotone operator [12]. In contrast to the stable semantics, the well-founded semantics is defined for every logic program. It is interesting to note that the well-founded semantics delivers the expected conclusion in the Barber's paradox example. The major shortcoming of the well-founded semantics is its inability to handle conclusions which can be reached only by "proof by cases". The following example illustrates this problem.

Example 1.3. Let P be

$$a \leftarrow \neg b$$

$$b \leftarrow \neg a$$

$$c \leftarrow a$$

$$c \leftarrow b$$

It is reasonable to expect that c holds. But with respect to the well-founded semantics, all a, b, c are unknown. Note that in this case the stable semantics provides the expected conclusions.

The fact that each approach to semantics of negation has its own strength and weakness suggests that there is probably not a "best" semantics for logic programs. Which semantics should be used depends on concrete applications. To be able to choose the "right" semantics among different ones, *it is of great importance to understand the inherent relations between them.*

Although a logic program is a set of clauses P, its canonical model semantics is defined by picking out some specific models of P. Here we are interested in the question whether or not it is possible to extend P into a richer theory Th which specifies the canonical model semantics of P in the sense that an interpretation is a model of Th iff it is a canonical model of P. This problem is strongly related to one open question between nonmonotonic logic and logic programming, the question whether or not there is a circumscriptive specification of the stable semantics [22, 18].

Since stable semantics is not defined for every logic program, it is interesting to characterize some class of programs for which the stable semantics is defined. In other words, we are looking for syntactical conditions guaranteeing the existence of at least one stable model.

Similar to the relation between three-valued and two-valued semantics of the Clark's completion, the well-founded semantics is weaker than the stable semantics. Since many ordinary programmers will find two-valued semantics more natural and easier to understand than a three-valued one, it is desirable to find sufficient conditions to guarantee the equivalence between stable and well-founded semantics.

The purpose of this paper is to study these problems.

- We show that stable semantics can be defined in the same way as well-founded semantics using the basic notion of unfounded sets. In other words, we show that stable semantics can be considered as "two-valued well-founded semantics". From this fact, together with the result in [23], which states that well-founded semantics can be viewed as three-valued stable semantics, we can conclude that the two concepts of stability and well-foundedness in the semantics of logic programming are equivalent. Thus, the difference between stable semantics and well-founded semantics indeed results just from the difference between the logics in which these concepts are interpreted.
- We construct for each program P a new completion theory called the *strong completion* of P, written scomp(P), such that
  - stable semantics is specified by two-valued strong completion,
  - well-founded semantics is specified by three-valued strong completion.
- We show that the call-consistency condition is sufficient for the existence of at least one stable model.
- We show that strictness guarantees the equivalence between stable and well-founded semantics.

- We introduce the bottom-stratified and top-strict condition generalizing both the stratifiability and the strictness, and show that the new condition is sufficient for the equivalence between stable semantics and well-founded semantics.

2. Stable models as relatively well-founded models

To illuminate the inherent relations between stable semantics and well-founded semantics, we show in this section that both of them can be defined using the same concept of unfounded sets.

Note that the logic we are working in in this section is the classical two-valued logic, where a Herbrand interpretation is considered as a subset of the Herbrand base.

A logical formula is defined as usual [2, 17]. A sentence is a closed logical formula. A logical theory is a set of sentences. A program clause is a clause of the form

$$A \leftarrow L_1, \ldots, L_n \quad \text{with } n \geqslant 0,$$

where A is an atom and L_i's are literals. If $n=0$, we write $A \leftarrow t$ or just $A \leftarrow$, where t denotes the truth value true. For any clause C, C^- (C^+) denotes the set of literals (the atom) occurring in the body (in the head) of C. Further, the set of positive subgoals in the body of C is denoted by $\text{pos}(C)$ while the set of atoms under negation in C is denoted by $\text{neg}(C)$.

A program is a finite set of program clauses. The set of all ground instances of the clauses in P is denoted by G_P. From now on, P denotes an arbitrary, but fixed, program. When we talk about the first-order language of P, we mean the language defined by the alphabet consisting exactly of the constant, function and predicate symbols occurring explicitly in P. We assume that P contains at least one constant. The Herbrand base (the Herbrand universe) of P is denoted by HB_P (HU_P) (or, shortly, HB and HU if the subscript is clear from the context). $\text{INT} = \{I \mid I \subseteq \text{HB}\}$ is the set of all Herbrand interpretations of P.

We recall now the definition of stable models from [13]. Let $I \subseteq \text{HB}$ be a Herbrand interpretation of P. The Gelfond–Lifschitz transformation of P w.r.t. I is the program $\text{GL}(I, P)$ obtained from G_P by:

- Deleting all clauses in G_P which have negative premises $\neg A$ such that $A \in I$.
- Deleting all negative premises $\neg A$ from the remaining clauses.

It is clear that $\text{GL}(I, P)$ is a definite Horn program. I is called a *stable model* of P if I is the least Herbrand model of $\text{GL}(I, P)$.

Example 2.1. Let P be $a \leftarrow a, \neg b$. Let $I = \emptyset$. Hence, $\text{GL}(I, P)$ consists of only the clause $a \leftarrow a$. I is a stable model of P since the least Herbrand model of $\text{GL}(I, P)$ is also empty.

The following notion of unfounded set is taken from [12].

Definition 2.2. A set S of ground atoms is an unfounded set of P with respect to an

Herbrand interpretation $I \in \mathrm{INT}$ if each atom $A \in S$ satisfies the following condition: For each clause C from G_P whose head is A, at least one of the following holds:

(1) The body of C is false in I.

(2) Some positive subgoal of the body of C occurs in S.

The following lemma reveals the inherent relations between the Gelfond–Lifschitz transformation and the unfoundedness.

Lemma 2.3. *Let $I \subseteq HB$ be a Herbrand interpretation of P. Then a set of ground atoms S is an unfounded set of P w.r.t. I iff S is an unfounded set of* $\mathrm{GL}(I, P)$ *w.r.t. I.*

Proof. Let Q denote $\mathrm{GL}(I, P)$. It is to be noted that Q is a set of ground-definite Horn clauses.

$\Rightarrow$: Let S be an unfounded set of P w.r.t. I. Let C be a clause in Q whose head belongs to S. We need to show that at least one of the following holds:

(1) The body of C is false in I.

(2) Some positive subgoal of the body of C occurs in S.

Assume that the body of C is true in I. This means that $C^- \subseteq I$. Since C belongs to $\mathrm{GL}(I, P)$, there exists a clause D in G_P such that D is of the form $C^+ \leftarrow C^-, \neg A_1, \ldots, \neg A_n$ and $A_i \in \mathrm{HB} \setminus I$ for each $0 < i < n+1$. Therefore, the body of D is also true in I. Since S is an unfounded set of P w.r.t. I, there exists a positive subgoal in the body of D which occurs in S. This means that there exists a positive subgoal in the body of C which occurs in S. Therefore, S is an unfounded set of Q w.r.t. I.

$\Leftarrow$: Let S be an unfounded set of Q w.r.t. I. Let D be a clause in G_P whose head belongs to S. We need to show that at least one of the following holds:

(1) The body of D is false in I.

(2) Some positive subgoal of the body of D occurs in S.

Assume that the body of D is true in I. It follows immediately that $\mathrm{pos}(D) \subseteq I$ and $\mathrm{neg}(D) \subseteq \mathrm{HB} \setminus I$. Let C denote the clause $D^+ \leftarrow \mathrm{pos}(D)$. Hence, C belongs to $\mathrm{GL}(I, P)$. Since S is an unfounded set of Q w.r.t. I, there exists a positive subgoal in the body of C which occurs in S. This means that there exists a positive subgoal in the body of D which occurs in S. Therefore, S is an unfounded set of P w.r.t. I. □

Definition 2.4. A Herbrand model $I \in \mathrm{INT}$ of P is said to be *relatively well-founded* if $I \cap S = \emptyset$ for every unfounded set S of P w.r.t. I.

The following theorem shows the correspondence between stable models and relatively well-founded models.

Theorem 2.5. *Let I be a Herbrand model of P. Then I is stable iff I is relatively well-founded.*

Proof. Let $Q = \mathrm{GL}(I, P)$. Define $T_Q : \mathrm{INT} \rightarrow \mathrm{INT}$ by

$$T_Q(J) = \{ A \mid \exists C \in Q\colon C^+ = A \text{ and } C^- \subseteq J \}.$$

Note that Q is a set of ground-definite Horn clauses. Thus, T_Q is continuous (w.r.t. set inclusion) and the least Herbrand model M of Q coincides with the least fixpoint of T_Q, which can be computed by $M = \bigcup \{ T_Q^i(\emptyset) \mid i \text{ is a natural number} \}$ [17]. Further, since I is a model of P, I is also a model of Q. Thus, $M \subseteq I$ since M is the least Herbrand model of Q.

$\Rightarrow$: Let I be a stable model of P. Let S be an arbitrary nonempty unfounded set of P w.r.t. I. To show that I is relatively well-founded, we need only to show that S and I are disjoint. From Lemma 2.3, it follows that S is an unfounded set of Q w.r.t. I. Since I is stable, I is the least Herbrand model of Q, i.e. $I = M = \bigcup_i T_Q^i(\emptyset)$. We show by induction that $S \cap T_Q^i(\emptyset) = \emptyset$ for any natural number i.

Initial step: $S \cap \emptyset = \emptyset$. Obvious.

Induction step: If $S \cap T_Q^i(\emptyset) = \emptyset$ then $S \cap T_Q^{i+1}(\emptyset) = \emptyset$ for any natural number i. Assume the contrary. Hence, there exists an atom $A \in S \cap T_Q^{i+1}(\emptyset)$. Therefore, there is a clause C in Q such that $A = C^+$ and $C^- \subseteq T_Q^i(\emptyset)$. Because $S \cap T_Q^i(\emptyset) = \emptyset$, we have that $C^- \cap S = \emptyset$. Since S is an unfounded set of Q w.r.t. I, and no subgoal in the body of C occurs in S, C^- must be false w.r.t. I. But this contradicts the fact that $C^- \subseteq T_Q^i(\emptyset) \subseteq I$. Thus, the induction hypothesis holds.

It follows immediately that S and I are disjoint.

$\Leftarrow$: Let I be a relatively well-founded model of P. Assume that I is not stable. Thus, $I \neq M$. Let $S = I \setminus M$. Hence, $S \neq \emptyset$ because $M \subseteq I$. We want to show that S is an unfounded set of Q w.r.t. I. Let $C \in Q$ such that $C^+ \in S$ and C^- is true w.r.t. I. Hence, $C^- \subseteq I$. Since M is a fixpoint of T_Q and $S = I \setminus M$ and $C^+ \in S$, C^- is false in M. Therefore, C^- contains one atom from S. So, S is an unfounded set of Q w.r.t. I. From Lemma 2.3, S is also an unfounded set of P w.r.t. I. Thus, S is disjoint to I. Contradiction!! Thus, $I = M$, i.e. I is a stable model of P. □

Przymusinski [23] has introduced the three-valued stable models, a natural extension of (two-valued) stable models, and showed that the well-founded model of any logic program coincides with its least three-valued stable model. In other words, well-founded semantics can be considered as three-valued stable semantics. From this fact, together with our result which states that stable semantics can be viewed as two-valued well-founded semantics, we can conclude that the two concepts of stability and well-foundedness in the semantics of logic programming are equivalent. Thus, the difference between stable semantics and well-founded semantics indeed results just from the difference between the logics in which these semantics are defined.

3. The strong completion of logic programs[1]

In this section, we first develop a new completion theory, called strong completion, for logic programs; then we show how the new theory specifies the stable semantics.

[1] Note that in this section we are still in the classical two-valued logic.

Let $p_1, \ldots, p_m$ be the predicate symbols occurring in P, and $q_1, \ldots, q_m$ be the corresponding predicate variables. Further, let S be a subset of HB, and for each $0<i<m+1$, S_i be the subset of S consisting of all those atoms in S whose predicate is p_i. Let $\mu=[q_1/S_1, \ldots, q_m/S_m]$ denote the following assignment of (two-valued) relations on the Herbrand universe of P to q_i:

$q_i(t_1, \ldots, t_{ni})$ is true iff $p_i(t_1, \ldots, t_{ni}) \in S_i$.[2]

Let C be a clause in P. Without loss of generality, we can assume that C is of the form

$$p(t_1, \ldots, t_n) \leftarrow p_1(t1_1, \ldots, t1_{n1}), \ldots, p_k(tk_1, \ldots, tk_{nk}), \text{NB}$$

where $p \in \{p_1, \ldots, p_m\}$ and NB consists of all negative subgoals in the body of C.

Define UF(C) as

$$\forall y_1 \cdots \forall y_r M,$$

where M is the following formula:

$$q(t_1, \ldots, t_n) \rightarrow (\neg C^- \vee q_1(t1_1, \ldots, t1_{n1}) \vee \cdots \vee q_k(tk_1, \ldots, tk_{nk}))$$

with q being the predicate variable in $\{q_1, \ldots, q_m\}$ corresponding to p, and y_i's being the free variables in C.

The intuition behind UF(C) is that given a certain subset S of HB, UF(C) is true w.r.t. a Herbrand interpretation I and $\mu=[q_1/S_1, \ldots, q_m/S_m]$ if and only if S is a unfounded set of C w.r.t. I. This can be explained as follows: UF(C) is true w.r.t. I and μ iff for any value assignment $w=[y_1/t_1, \ldots, y_r/t_r]$ (i.e. for any ground instance D of C) if $q(w(t_1), \ldots, w(t_n))$ is true w.r.t. μ (i.e. the head of D is in S) then either C^- is false w.r.t. I, w (i.e. the body of D is false in I) or the disjunction $q_1(w(t1_1), \ldots, w(t1_{n1})) \vee \cdots \vee q_k(w(tk_1), \ldots, w(tk_{nk}))$ is true w.r.t. μ (i.e. some positive subgoal of the body of D is contained in S).

Hence, S is an unfounded set of P w.r.t. I if and only if UF(C) *is true w.r.t. I and $\mu=[q_1/S_1, \ldots, q_m/S_m]$ for each clause C of P.*

Example 3.1. Let P be

C_1: $p_1(x) \leftarrow p_2(x)$

C_2: $p_2(a) \leftarrow \neg p_2(b)$

C_3: $p_2(b) \leftarrow \neg p_2(c)$

The Herbrand universe of P is $\{a, b, c\}$. Let $I=\{p_1(b), p_2(b)\}$ and $S=\{p_1(a), p_2(a)\}$. It is clear that S is unfounded w.r.t. I.

[2] It would be more exact to say that $\mu(q_i)(t_1, \ldots, t_{ni})$ is true if $p_i(t_1, \ldots, t_{ni}) \in S_i$. We abuse the notation here for the sake of simplicity.

We have

$$\mathrm{UF}(C_1)\colon\ \forall x(q_1(x)\rightarrow(\neg p_2(x)\vee q_2(x))$$

$$\mathrm{UF}(C_2)\colon\ q_2(a)\rightarrow p_2(b)$$

$$\mathrm{UF}(C_3)\colon\ q_2(b)\rightarrow p_2(c)$$

Let $\mu=[q_1/S_1, q_2/S_2]$, where $S_1=\{p_1(a)\}$ and $S_2=\{p_2(a)\}$. It is clear that $\mu(q_1)=\{a\}$ and $\mu(q_2)=\{a\}$. It is not difficult to see that $\mathrm{UF}(C_i)$ is true w.r.t. I and μ for each $0<i<4$.

Let $C_1,\ldots,C_n$ be the clauses of P. Let $\mathrm{UF}(P)$ denote the conjunction $\mathrm{UF}(C_1)\wedge\cdots\wedge\mathrm{UF}(C_n)$, and let $\mathrm{WF}(P)$ be the following second-order sentence

$$\forall q_1\cdots\forall q_m(\mathrm{UF}(P)\rightarrow((q_1\leqslant\neg p_1)\wedge\cdots\wedge(q_m\leqslant\neg p_m))),$$

where $q_i\leqslant\neg p_i$ denotes $\forall x_1\cdots\forall x_{ni}(q_i(x_1,\ldots,x_{ni})\rightarrow\neg p_i(x_1,\ldots,x_{ni}))$.

It is not difficult to see that $q_i\leqslant\neg p_i$ is true w.r.t. I and $\mu=[q_1/S_1,\ldots,q_m/S_m]$ if and only if S_i and I are disjoint to each other. *Hence,* $\mathrm{WF}(P)$ *is true w.r.t. Herbrand interpretation I iff for each subset S of the Herbrand base, if S is unfounded w.r.t. I then S is disjoint to I. Therefore, if I is a Herbrand model of P and* $\mathrm{WF}(P)$*, then I is a relatively well-founded model of P.*

Definition 3.2. The *strong completion of* P, $\mathrm{scomp}(P)$, is defined as the following theory

$$P+\mathrm{WF}(P)+\mathrm{CET},$$

where CET is Clark's equality theory [17] of the language of P.

Example 3.3 (Continuation of Example 3.1). Let P, I be defined as in Example 3.1. $\mathrm{WF}(P)$ is the following sentence:

$$\forall q_1\forall q_2(\mathrm{UF}(P)\rightarrow(\forall x(q_1(x)\rightarrow\neg p_1(x))\wedge\forall x(q_2(x)\rightarrow\neg p_2(x)))).$$

Let τ be an arbitrary assignment of unary relations on $\{a,b,c\}$ to q_1, q_2 such that $\mathrm{UF}(P)$ is true w.r.t. I and τ. Then there exists an (w.r.t. I) unfounded set R such that $\tau=[q_1/R_1, q_2/R_2]$. It is not difficult to see that the following properties are satisfied:

For any $t\in\{a,b,c\}$: $p_1(t)\in R$ iff $p_2(t)\in R$ and $p_2(b)\notin R$.

Therefore, it is easy to see that if $q_i(t)$ is true w.r.t. τ then $t\neq b$. Since $\neg p_i(t)$ is true iff $t\neq b$, it follows that $\forall x(q_i(x)\rightarrow\neg p_i(x))$ is true w.r.t. I and τ.

Thus, $\mathrm{WF}(P)$ is true w.r.t. I. Since I is clearly a model of P, I is a relatively well-founded model of P.

The following theorem follows immediately from Theorem 2.5.

Theorem 3.4. *I is a stable model of P iff I is a Herbrand model of* scomp(P).

Since the semantics of negation in logic programming is nonmonotonic, in the sense that an extension to a logic program does not necessarily lead to an extension of its consequences, the question about its relations to other nonmonotonic logics arises naturally. In [18, 13], it is shown that both default logic and autoepistemic logic specify the stable semantics of logic programs. But there is only a partial answer in the literature to the question about the relations between McCarthy's circumscription and stable semantics [6, 20]. If the programs are stratified then prioritized circumscription specifies the perfect-model semantics (another name for the stable semantics of stratified programs) [20]. An extension of the prioritized circumscription for locally stratified programs is given in [22]. But the problems here is that because of the undecidability of the local stratification [3], given a logic program it is undecidable whether or not we could circumscribe this program. An extension of McCarthy's circumscription in a three-valued logic which specifies the stable semantics is given in [6]. But the approach proposed there is a bit strange since the circumscriptive specification is three-valued whereas the stable semantics is two-valued. Although our strong completion theory is not directly a form of McCarthy's circumscription [19], it has the same spirit as it also circumscribes the effects of the predicates. The interesting point is that while McCarthy's circumscription circumscribes the "positive parts" of the predicates, our approach circumscribes their "negative parts". This may indicate that, in general, a dual form of McCarthy's circumscription may be defined based on an appropriate generalization of the well-foundedness where, instead of circumscribing the positive informations, the negative informations should be circumscribed.

4. Well-founded semantics: three-valued strong completion

We show in this section that three-valued strong completion specifies well-founded semantics.

The logic for this section is three-valued with the truth values **t** (true), **f** (false), and **u** (undefined).

The logical operators $\wedge$, $\vee$, and $\neg$ are the Kleene's operators defined by the following truth tables:

$\wedge$	**t**	**f**	**u**
t	**t**	**f**	**u**
f	**f**	**f**	**f**
u	**u**	**f**	**u**

$\vee$	**t**	**f**	**u**
t	**t**	**t**	**t**
f	**t**	**f**	**u**
u	**t**	**u**	**u**

$\neg$	
t	**f**
f	**t**
u	**u**

The operator $\leftarrow$ is defined by the following truth table:

$\leftarrow$	t	f	u
t	t	t	t
f	f	t	t
u	f	t	t

The intuition behind "$p \leftarrow q$" is that if q is true then p is true, too. For more about this operator see [25].

A partial three-valued interpretation I of a program P is a pair $\langle \mathrm{IT}, \mathrm{IF} \rangle$, where IT, IF are disjoint subsets of HB_P. Often, if no confusion is possible, we shortly say three-valued interpretation for partial three-valued interpretation. The set IT contains all ground atoms true in I, the set IF contains all ground atoms false in I. The truth value of the remaining atoms is undefined. A three-valued interpretation I is *total* if $\mathrm{HB}_P = \mathrm{IT} \cup \mathrm{IF}$. The set of all three-valued interpretations is denoted by PINT. It is clear that any two-valued interpretation $M \subseteq \mathrm{HB}$ corresponds to $\langle M, \mathrm{HB} \setminus M \rangle$. The union between two three-valued interpretations is defined by $\langle \mathrm{IT}, \mathrm{IF} \rangle \cup \langle \mathrm{IT}', \mathrm{IF}' \rangle = \langle \mathrm{IT} \cup \mathrm{IT}', \mathrm{IF} \cup \mathrm{IF}' \rangle$. A partial order $\leqslant$ in PINT is defined by $(\mathrm{IT}, \mathrm{IF}) \leqslant (\mathrm{IT}', \mathrm{IF}')$ iff $\mathrm{IT} \subseteq \mathrm{IT}'$ and $\mathrm{IF} \subseteq \mathrm{IF}'$.

Definition 4.1. T_P^+, T_P^- are mappings from PINT into INT defined as follows:

$$T_P^+(I) = \{A \mid \exists C \in G_P\colon C^+ = A \text{ and } C^- \text{ is true in } I\},$$

$$T_P^-(I) = \{A \mid \forall C \in G_P\colon \text{if } C^+ = A \text{ then } C^- \text{ is false in } I\}.$$

Further, let $T_P(I) = \langle T_P^+(I), T_P^-(I) \rangle$.

From the definition of the implication operator in our three-valued logic, Lemma 4.2 follows immediately.

Lemma 4.2. *A three-valued interpretation $I = \langle \mathrm{IT}, \mathrm{IF} \rangle$ is a model of P iff $T_P^+(I) \subseteq \mathrm{IT}$.*

The notion of unfounded sets can be interpreted straightforwardly for any three-valued interpretation. For the sake of readability we recall it again here.

Definition 4.3. A set S of ground atoms is a unfounded set of P with respect to an interpretation $I = \langle \mathrm{IT}, \mathrm{IF} \rangle$ if each atom $A \in S$ satisfies the following condition: For each clause C from G_P whose head is A, at least one of the following holds:

(1) The body of C is false in I.

(2) Some positive subgoal of the body of C occurs in S.

It is easy to see that the union of unfounded sets is again an unfounded set. So, for any interpretation I there exists always a greatest unfounded set of P with respect to I. This set is denoted by $\mathrm{GU}(I)$. Define

$$V_P(I) = \langle T_P^+(I), \mathrm{GU}(I) \rangle.$$

It is clear that V_P is monotone.

Lemma 4.4. *V_P is monotone.*

Therefore, V_P has a least fixpoint. The *well-founded model* of P, written as WFM_P, is defined as the least fixpoint of V_P [12].

Theorem 4.5. *Let I be a three-valued Herbrand interpretation of P. Then I is a model of* $\mathrm{scomp}(P)$ *iff $V_P(I) \leqslant I$.*

Proof. $\Leftarrow$: Let $I = \langle \mathrm{IT}, \mathrm{IF} \rangle$ such that $V_P(I) \leqslant I$. Since I is a Herbrand interpretation, I clearly satisfies Clark's equality theory CET. From $V_P(I) \leqslant I$, it follows immediately that $T_P^+(I) \subseteq \mathrm{IT}$. Thus, I is a three-valued model of P (Lemma 4.2). It remains to show that I satisfies $\mathrm{WF}(P)$. For each $0 < i < n+1$, let $\mu = [q_1/\mathrm{atr}_1, \ldots, q_m/\mathrm{atr}_m]$ be an assignment of three-valued relations atr_i on Herbrand universe of P to q_i such that $\langle I, \mu \rangle$ satisfies $\mathrm{UF}(C_1) \wedge \cdots \wedge \mathrm{UF}(C_n)$. We have to show that $\langle I, \mu \rangle$ satisfies $(q_i \leqslant \neg p_i)$ for each i. Let $S = \{p_i(t_1, \ldots, t_{ni}) \in \mathrm{HB} \mid \mathrm{atr}_i(t_1, \ldots, t_{ni}) \text{ is true}\}$. From the definition of the three-valued implication operator and the fact that $\langle I, \mu \rangle$ satisfies $\mathrm{UF}(C_1) \wedge \cdots \wedge \mathrm{UF}(C_n)$, it follows immediately that S is an unfounded set of P with respect to I. Thus, S is a subset of $\mathrm{GU}(I)$. Hence, $S \subseteq \mathrm{IF}$, i.e. if $\mu(q_i)(t_1, \ldots, t_{ni})$ is true then $p_i(t_1, \ldots, t_{ni})$ is false in I. Therefore, $\langle I, \mu \rangle$ satisfies $(q_i \leqslant \neg p_i)$.

$\Rightarrow$: Let $I = \langle \mathrm{IT}, \mathrm{IF} \rangle$ be a Herbrand model of $\mathrm{scomp}(P)$. Since I is a model of P, it follows that $T_P^+(I) \subseteq \mathrm{IT}$. It is easy to see that IF contains every unfounded set of P with respect to I. Hence, $\mathrm{GU}(I) \subseteq \mathrm{IF}$. Thus, $V_P(I) \leqslant I$. □

Corollary 4.6 follows immediately.

Corollary 4.6. *The well-founded model of P is the least three-valued Herbrand model of* $\mathrm{scomp}(P)$.

From Corollary 4.6, it follows immediately that the well-founded semantics of P is specified correctly by the three-valued strong completion.

5. Signed dependencies and the equivalence between stable and well-founded semantics

Since not every logic program has a stable model, it is meaningful to ask for sufficient conditions guaranteeing the existence of at least one stable model. Further, because many ordinary programmers will find a two-valued semantics more natural and easier to understand than a three-valued one [15], it is desirable to find sufficient conditions for the equivalence between stable and well-founded semantics. In this section, these problems will be addressed.

From now on, we permit programs to contain possibly infinitely many clauses, but require that only finitely many predicates appear in each program.

To avoid any possible confusion, we want to note that the logic we are working in in this section is three-valued, where a three-valued interpretation $\langle T, F \rangle$ is said to be total if $T \cup F = \mathrm{HB}$.[3] Thus, a total (three-valued) interpretation $\langle M, \mathrm{HB} \setminus M \rangle$ is stable if M is a (two-valued) stable model in the classical two-valued logic as defined in Section 1. A predicate p is *totally defined* w.r.t. a three-valued interpretation I if each ground atom of p is either true or false in I.

Let Pred be the set of all predicate symbols occurring in P. The predicate dependency graph [1] of a program is a directed graph with signed edges. The nodes are the elements of Pred. An edge from p to q is positive (negative) iff p occurs in the head of a clause C of P and q occurs in a positive (negative) literal in the body of C.

Define the binary relations $\geqslant_{+1}$ and $\geqslant_{-1}$ as: $p \geqslant_{+1} q$ ($p \geqslant_{-1} q$) iff there is a path from p to q containing an even (odd) number of negative edges in the predicate dependency graph of P.

Further, let us define

$$p \geqslant q \text{ iff } p \geqslant_{+1} q \text{ or } p \geqslant_{-1} q \text{ or } p = q,$$

$$p > q \text{ iff } p \geqslant q \text{ and } p \neq q,$$

$$p \gg q \text{ iff } p \geqslant_{+1} q \text{ and } p \geqslant_{-1} q,$$

$$p \equiv q \text{ iff } p \geqslant q \text{ and } q \geqslant p,$$

$$[p]\sim = \{q \mid p \equiv q\},$$

$$[p]_{\leqslant} = \{q \mid p \geqslant q\}.$$

If $\mathcal{S} \subseteq$ Pred, we say that $\mathcal{S}$ is *downward closed* iff for all $p \in \mathcal{S}$ and $q \in$ Pred, $p \geqslant q$ implies $q \in \mathcal{S}$. This implies that $\mathcal{S} = \cup \{[p]\sim \mid p \in \mathcal{S}\}$.

For any downward closed set $\mathcal{S}$, let $P|_{\mathcal{S}} = \{C \in P \mid$ the predicate of C^+ belongs to $\mathcal{S}\}$. If $\mathcal{S} = [q]_{\leqslant}$ for some q then $P|_{\mathcal{S}}$ is also denoted by $Def_P(q)$.

A program is said to be *stratified* [1] if we never have both $p \equiv q$ and $p \geqslant_{-1} q$, i.e. within each equivalence class all dependencies are positive.

A program P is *call-consistent* [24, 15] if there is no predicate symbol p such that $p \geqslant_{-1} p$. P is *strict* [1, 15] if we never have $p \gg q$.

It is easy to verify that any program which is either strict or stratified is call-consistent but not vice versa.

Example. Let P be

$$c \leftarrow a$$

$$c \leftarrow b$$

$$a \leftarrow \neg b$$

$$b \leftarrow \neg a$$

[3] Note that when we speak about a two-valued interpretation, we always mean a subset of the Herbrand base.

Then $a \geqslant_{-1} b$ and $b \geqslant_{-1} a$. Further, $c \gg a$. Thus, P is call-consistent but not strict.

Let $\$ \subseteq$ Pred. A *signing* is a map, $\text{sig}: \$ \to \{+1, -1\}$ such that whenever $p, q \in \$$ and $p \leqslant_i q$, $\text{sig}(p) = i.\text{sig}(q)$. As we will see very soon, signings will allow us to convert partial fixpoints of T_P into total ones.

Definition 5.1. A program P is called *stable-consistent* if P has at least one stable model.

To show that the call-consistency is sufficient for the existence of at least one stable model, we need the following notion of semantic kernel.

5.1. *Semantic kernels of logic programs*

The semantic kernel of a logic program is defined as the fixpoint of a continuous operator Q_P on quasi-interpretations [8], where a *quasi-interpretation* is a set of ground clauses of the form $A \leftarrow \neg B_1, \ldots, \neg B_n, n \geqslant 0$, with A, B_i being ground atoms, and the operator Q_P on quasi-interpretations is defined as follows:

$$\begin{aligned} Q_P(I) = \{ & A \leftarrow \neg B_1, \ldots, \neg B_n, \text{Body}_1, \ldots, \text{Body}_m \mid \\ & \text{there exist } C \in G_P \text{ and } C_i \in I,\ 1 \leqslant i \leqslant m \text{ s.t. } C \text{ is of the form} \\ & A \leftarrow \neg B_1, \ldots, \neg B_n, A_1, \ldots, A_m \text{ with } n \geqslant 0,\ m \geqslant 0 \text{ and} \\ & C_i \text{ is of the form } A_i \leftarrow \text{Body}_i \text{ with } 1 \leqslant i \leqslant m \}. \end{aligned}$$

Q_P is a continuous operator in the lattice of the quasi-interpretations [8]. The semantics kernel of P, written as $\text{SK}(P)$, is defined by

$$\text{SK}(P) = \bigcup \{ Q_P^n(\emptyset) \mid n \geqslant 1 \}.$$

Lemma 5.2 (Dung and Kanchana Kanchanasut [8, 9]). (1) *Let $I = \langle \text{IT}, \text{IF} \rangle$ be a partial three-valued model of P. Then I is stable iff I is a total fixpoint of $T_{\text{SK}(P)}$.*

(2) *The least fixpoint of $T_{\text{SK}(P)}$ is the well-founded model of P.*

The following lemma holds obviously.

Lemma 5.3. *If P is call-consistent (strict) then $\text{SK}(P)$ is also call-consistent (strict).*

5.2. *Existence of stable models*

Our goal in this section is to show that call-consistent programs are stable-consistent. In [15] the following result has been proved.

Lemma 5.4. *If P is finite and call-consistent then T_P has at least a total fixpoint.*

Hence, call-consistent logic programs with finite semantic kernel always possess at least one stable model. We prove now that Lemma 5.4 holds also for infinite P which contains only finitely many predicates.

Lemma 5.5. *Let I be a three-valued interpretation satisfying the property $I \leqslant T_P(I)$. Then there exists a fixpoint J of T_P such that $I \leqslant J$.*

Proof. Obvious since T_P is monotone. □

Since T_P is monotone (w.r.t $\leqslant$), it has a fixpoint. To show the existence of a total fixpoint of T_P, we need the following lemma.

Lemma 5.6. *If P is call-consistent and $I = \langle \mathrm{IT}, \mathrm{IF} \rangle$ is a fixpoint of T_P such that $\mathrm{IT} \cup \mathrm{IF} \neq \mathrm{HB}$. Then there exists a Herbrand interpretation $I' = \langle \mathrm{IT}', \mathrm{IF}' \rangle$ such that*

(1) $I \leqslant I' \leqslant T_P(I')$.

(2) *There exists a predicate p in* Pred *such that p is totally defined w.r.t. I', but not totally defined w.r.t. I.*

Proof. Let $p = \min_{<} \{q \in \mathrm{Pred} \mid q$ is not totally defined w.r.t. $I\}$. Let $\mathrm{sig}: [p] \sim \to \{+1, -1\}$ be defined as follows: $\mathrm{sig}(p) = +1$ and for all $q \in [p] \sim$, $\mathrm{sig}(q) = i$ if $p \geqslant_i q$. Since P is call-consistent, sig is a signing of $[p] \sim$. Define a three-valued interpretation $I' = \langle \mathrm{IT}', \mathrm{IF}' \rangle$ as follows:

$$\mathrm{IT}' = \mathrm{IT} \cup \{A \mid A = q(t_1, \ldots, t_n) \text{ s.t. } q \in [p] \sim \text{ and } \mathrm{sig}(q) = +1 \text{ and } A \notin \mathrm{IT} \cup \mathrm{IF}\}$$

$$\mathrm{IF}' = \mathrm{IF} \cup \{A \mid A = q(t_1, \ldots, t_n) \text{ s.t. } q \in [p] \sim \text{ and } \mathrm{sig}(q) = -1 \text{ and } A \notin \mathrm{IT} \cup \mathrm{IF}\}$$

It is not difficult to see that $I \leqslant I' \leqslant T_P(I')$ and each predicate q from $[p] \sim$ is totally defined w.r.t. I'. □

Since only finitely many predicates are occurring in P, Lemma 5.7 follows immediately (by induction) from Lemma 5.6.

Lemma 5.7. *If P is call-consistent then T_P has at least one total fixpoint.*

Theorem 5.8.[4] *If P is call-consistent then P is stable-consistent.*

Proof. It is clear that the semantic kernel of P, $\mathrm{SK}(P)$, is also call-consistent. Thus, $T_{\mathrm{SK}(P)}$ has at least one total fixpoint. It follows immediately from Lemma 5.2 that P has a stable model. □

[4] Independently, Fages [10] has shown that order-consistent logic programs [24] are also stable-consistent. Although the result of Fages is slightly more general than that of ours, the proofs of both results are largely similar.

The following lemma is an immediate consequence of the previous lemmas.

Lemma 5.9. *Let P be a call-consistent program and I a fixpoint of T_P. Then T_P has a total fixpoint J such that $I \leqslant J$.*

5.3. Equivalence between stable and well-founded semantics

Let L be a ground literal of P. We write $P \models_s L$ if L is true in every stable model of P. Similarly, we write $P \models_w L$ if L is true in the well-founded model of P.

It is clear that the well-founded semantics is weaker than the stable semantics, i.e. if $P \models_w L$ then $P \models_s L$.

Definition 5.10. We say that the stable semantics is equivalent to the well-founded semantics if for any ground literal L: if $P \models_s L$ then $P \models_w L$.

Theorem 5.11. *If P is strict then the stable semantics is equivalent to the well-founded semantics.*

Proof. Assume the contrary. Then there exists a ground literal L such that $P \models_s L$ and $P \not\models_w L$. To show the contradiction, we construct a stable model in which L is false. Let p be the predicate occurring in L, and

$$\$ = \{q \in \text{Pred} \mid p \geqslant q\}. \text{ Define sig}: \$ \to \{+1, -1\} \text{ by}$$

$$\text{sig}(q) = \begin{cases} -i & \text{if } L \text{ is a positive literal and } p \geqslant_i q, \\ i & \text{if } L \text{ is a negative literal and } p \geqslant_i q. \end{cases}$$

Since P is strict, sig is clearly a signing of $\$$. Let $\text{WFM} = \langle T, F \rangle$ be the well-founded model of P. Define an interpretation $I = \langle \text{IT}, \text{IF} \rangle$ as follows:

$$\text{IT} = T \cup \{A \mid A = q(t_1, \ldots, t_n) \text{ s.t. } q \in \$ \text{ and } \text{sig}(q) = +1 \text{ and } A \notin T \cup F\},$$

$$\text{IF} = F \cup \{A \mid A = q(t_1, \ldots, t_n) \text{ s.t. } q \in \$ \text{ and } \text{sig}(q) = -1 \text{ and } A \notin T \cup F\}.$$

From the facts that WFM is a fixpoint of $T_{\text{SK}(P)}$ (Lemma 5.2) and $\text{SK}(P)$ is also strict, it is not difficult to see that $\text{WFM} \leqslant I \leqslant T_{\text{SK}(P)}(I)$. Hence, there exists a total fixpoint $J = \langle M, \text{HB} \setminus M \rangle$ of $T_{\text{SK}(P)}$ such that $\text{WFM} \leqslant I \leqslant J$ (Lemma 5.9). It follows immediately that L is false in J. But L is also true in J since $P \models_s L$ and J is stable (Lemma 5.2). Contradiction!!. □

Another class of programs whose stable semantics and well-founded semantics are equivalent, is the class of stratified programs.[5] Note that the class of strict programs and the class of stratified programs are independent. One does not include the other. For example, the program consisting of the two clauses $p \leftarrow \neg q$ and $q \leftarrow \neg p$ is strict

[5] It is well known that the stable model and well-founded model of stratified programs coincide.

but not stratified while the program consisting of only the clause $p \leftarrow q, \neg q$ is stratified but not strict. Thus, there arises naturally the question of a sufficient condition for the equivalence between stable and well-founded semantics which generalizes both the stratifiability and the strictness.

Definition 5.12. A program is said to be *bottom-stratified and top-strict* iff for each pair p, q

$$\text{if } p \gg q \text{ then } \mathrm{Def}_P(q) \text{ is stratified.}$$

It is clear that bottom-stratified and top-strict programs are call-consistent. Further, any program which is either strict or stratified is bottom-stratified and top-strict but not vice versa.

Example. The following program is bottom-stratified and top-strict, but neither strict nor stratified.

$$c \leftarrow a$$

$$a \leftarrow \neg b$$

$$b \leftarrow \neg a$$

$$c \leftarrow \neg r$$

$$c \leftarrow r$$

Theorem 5.13. *Well-founded semantics and stable semantics are equivalent for bottom-stratified and top-strict programs.*

Proof. Let $\mathrm{TOP} = \{p \mid \exists q: p \geqslant q \text{ and } \mathrm{Def}_P(q) \text{ is not stratified}\}$. Thus, TOP is strict, i.e. there are no two predicates p, q from TOP such that $p \gg q$. Let $\mathrm{BOTT} = \mathrm{Pred} \setminus \mathrm{TOP}$. Then it is clear that for each $p \in \mathrm{BOTT}$, $\mathrm{Def}_P(p)$ is stratified. It is easy to see that BOTT is downward closed. Let $Q = P|_{\mathrm{BOTT}}$. Further, let $R = P \setminus Q$. Then it is clear that the predicates in BOTT occur only in the clause bodies of clauses in R. It is not difficult to see that Q is stratified. Therefore, its well-founded model $\mathrm{WFM} = \langle T, F \rangle$ is also stable. Let Reduct be the program obtained from G_R as follows:

- Deleting each clause whose body contains one literal L whose predicate is in BOTT such that L is false in WFM.
- Deleting all remained literals whose predicates are in BOTT.

It is easy to see that Reduct satisfies the following two propositions.

Proposition 5.14. *Let* $S \subseteq \mathrm{HB}$ *be a two-valued Herbrand interpretation of* P. *Then* S *is a (two-valued) stable model of* P *iff there is a (two-valued) stable model* S' *of* Reduct *such that*

$$S = S' \cup T.$$

Proposition 5.15. *Let W, W' be the well-founded models of P and* Reduct, *respectively. Then*

$$W = W' \cup \mathrm{WFM}.$$

Proof of Theorem 5.13 (*Conclusion*). It is clear that Reduct is a strict program. Thus, the well-founded semantics and stable semantics of Reduct are equivalent. From Propositions 5.14 and 5.15, it follows immediately that the well-founded semantics and stable semantics of P are equivalent, too. □

The results of this section can be generalized to show the equivalence between two-valued strong completion and three-valued strong completion. But to do so, we would have to introduce the notion of non-Herbrand stable models as well as non-Herbrand well-founded models. Further, an extension of the semantic kernel to accommodate clauses with variables would also be necessary. But this would go beyond the scope of this paper. A forthcoming paper will handle this problem.

6. Conclusion

The goal of this paper was to study the inherent relations between stable semantics and well-founded semantics. The results are threefold. First, we have shown that stable semantics can be considered as two-valued well-founded semantics. We argue that the two concepts of stability and well-foundedness in the semantics of logic programming are equivalent. Second, we have given an axiomatic characterization of the stable and well-founded semantics by introducing a new completion theory called strong completion. Third, we have studied the equivalence between the two semantics and found a new sufficient condition for their equivalence, the bottom-stratified and top-strict condition.

Since the strong completion is a second-order formula, it would be meaningful to ask whether it is possible to transform it into a first-order theory. Unfortunately, the answer is negative. In [5], we show that, in general, strong completion is not first-order-definable. This result suggests that in order to have a first-order characterization of stable and well-founded semantics, new extra function symbols as well as predicate symbols should be used [26].

Acknowledgment

Many thanks to Huynh Ngoc Phien for his support during the time in which this paper was written. I am also grateful to the anonymous referees for their comments pointing out many shortcomings of the previous versions of this paper.

References

[1] K.R. Apt, H.A. Blair and A. Walker, Towards a theory of declarative knowledge, in: J. Minker, ed., *Foundations of Deductive Databases and Logic Programming* (Morgan Kaufmann, Los Altos, CA, 1988) 89–148.

[2] C.L. Chang and R.C. Lee, *Symbolic Logic and Mechanical Theorem Proving* (Academic Press, New York, 1973).
[3] P. Cholak, Post corresponding problems and Prolog programs, Tech. Report, Univ. of Wisconsin, Madison, 1988.
[4] K.L. Clark, Negation as failure, in: H. Gallaire and J. Minker, eds., *Logic and Databases* (Plenum, New York, 1978) 293–322.
[5] P.M. Dung, On first-order definability of stable model semantics, Tech. Report, Division of Computer Science, Asian Institute of Technology, Bangkok, 1990.
[6] P.M. Dung, Strong circumscription specifies the stable semantics of logic programs, in: *Proc. of European Conf. on Artificial Intelligence 90 (ECAI-90)*, Stockholm (1990) 443–448.
[7] P.M. Dung, Negation as hypotheses: An abductive foundation of logic programming, in: *Proc. 8th Internat. Conf. on Logic Programming*, Paris (MIT Press, Cambridge, MA, 1991) 3–17.
[8] P.M. Dung and Kanchana Kanchanasut, A fixpoint approach to declarative semantics of logic programs, in: *Proc. North American Conf. on Logic Programming*, Cleveland, Ohio (MIT Press, Cambridge, MA, 1989) 601–625.
[9] P.M. Dung and Kanchana Kanchanasut, A natural semantics of logic programs with negations, in: C.V. Madhavan, ed., *Proc. Conf. on Foundations of Software Technology and Theoretical Computer Science*, India, Lecture Notes in Computer Science, Vol. 405 (Springer, Berlin, 1989) 78–88.
[10] F. Fages, Consistency of Clark's completion and existence of stable models, in: *Workshop on Nonmonotonic Reasoning and Logic Programming, NACLP-90*, Austin, USA; Research Report 90-15, Ecole Normale Superieure, France, 1990.
[11] M. Fitting, A Kripke–Kleene semantics for logic programs, *J. Logic programming* **2** (1985) 295–312.
[12] A. Van Gelder, K. Ross and J.S. Schlipf, Unfounded sets and well-founded semantics for general logic programs, in: *Proc. Conf. on Principles of Database Systems* (1988) 221–230.
[13] M. Gelfond and V. Lifschitz, The stable model semantics for logic programs, in: *Proc. 5th Internat. Conf./Symp. on Logic Programming* (MIT Press, Cambridge, MA, 1988) 1070–1080.
[14] K. Kunen, Some remarks on the completed database, Tech. Report, Dept. of Computer Science, Univ. of Wisconsin, Madison, 1988.
[15] K. Kunen, Signed data dependencies in logic programming, *J. Logic Programming* **7** (1989) 231–245.
[16] V. Lifschitz, Computing circumscription, in: M.L. Ginsberg, ed., *Reading in Nonmonotonic Reasoning* (Morgan Kaufman, Los Altos, CA, 1987) 167–173.
[17] J.W. Lloyd, *Foundations of Logic Programming* (Springer, Berlin, 2nd ed., 1987).
[18] W. Marek and M. Truszczynski, Stable semantics for logic programs and default theories, in: *Proc. North American Conf. on Logic Programming '89*, Cleveland, Ohio (MIT Press, Cambridge, MA, 1989) 243–256.
[19] J. McCarthy, Applications of circumscription to formalizing common-sense knowledge, *Artificial Intelligence* **28** (1986) 89–116.
[20] T.C. Przymusinski, On the declarative semantics of deductive databases and logic programs, in: J. Minker, ed., *Foundations of Deductive Databases and Logic Programming* (Morgan Kaufmann, Los Altos, CA, 1988) 193–216.
[21] T.C. Przymusinski, Perfect model semantics, in: *Proc. 5th Internat. Conf. on Logic Programming* (MIT Press, Cambridge, MA, 1988) 1081–1096.
[22] T.C. Przymusinski, Three-valued formalizations of non-monotonic reasoning and logic programming, in: *Proc. 1st Internat. Conf. on Principles of Knowledge Representation and Reasoning*, Toronto, Canada (1989) 341–348.
[23] T.C. Przymusinski, Well-founded semantics coincides with three-valued stable semantics, Tech. Report, Dept. of Mathematics, Univ. of Texas at El Paso, 1989.
[24] T. Sato, On consistency of first-order logic programs, Tech. Report TR 87-12, Electrotechnical Laboratory, Univ. of Ibaraki, Japan, 1987.
[25] J.C. Shepherdson, Negation in logic programming, in: J. Minker, ed., *Foundations of Deductive Databases and Logic Programming* (Morgan Kaufmann, Los Altos, CA, 1988) 19–88.
[26] M. Wallace, Unrestricted logic programs or if stratification is the cure, what is the malady?, in: *Proc. European Conf. on Artificial Intelligence '90*, Stockholm (1990) 682–687.