Epistemic Foundation of the Well-Founded Semantics over Bilattices

Yann Loyer[1] and Umberto Straccia[2]

[1] PRiSM (CNRS UMR 8144), Université de Versailles, FRANCE
Yann.Loyer@prism.uvsq.fr
[2] I.S.T.I. - C.N.R., Via G. Moruzzi,1 I-56124 Pisa (PI) ITALY
Umberto.Straccia@isti.cnr.it

Abstract. We define new, both model-theoretical and fixpoint-based, characterizations of the well-founded semantics for logic programs in the general setting of bilattices. This work lights the role of the CWA, used in the well-founded semantics as a carrier of falsehood, and shows that the definition of that semantics does not require any separation of positive and negative information nor any program transformation.

1 Introduction

One of the most important problems of logic programming consists in defining the intended meaning or semantics of any given logic program. Classical logic programming has the set $\{\mathtt{f}, \mathtt{t}\}$ (false, true) as its intended truth space, and the usual semantics of a negation-free logic program is given by its unique minimal Herbrand model [7], relying on the *Closed World Assumption* (CWA) to *complete* the available knowledge. The CWA assumes that all atoms not entailed by a program are false [21], and is motivated by the fact that explicit representation of negative information in logic programs is not feasible since the addition of explicit negative information could overwhelm a system. But, in order to increase the expressivity of the language, it is often necessary to allow some non-monotonic modes of negation, having as a consequence that the existence of such a model is no more guaranteed.

A widely used solution consists in allowing partial models by extending the classical set of truth-values $\{\mathtt{f}, \mathtt{t}\}$ to the set $\{\mathtt{f}, \mathtt{t}, \bot\}$, where $\bot$ stands for *unknown* and is closely related to null values in database systems. Over that set can be defined two orderings: the truth order $\preceq_t$ that extends the classical order $\mathtt{f} \preceq_t \bot \preceq_t \mathtt{t}$), and the knowledge order $\preceq_k$ ($\bot$ represents less knowledge than $\mathtt{f}$ and $\mathtt{t}$ that are both incomparable).

A first approach [10,13] has lead to the *Kripke-Kleene semantics* of logic programs, based on an extension $\Phi_{\mathcal{P}}$ to the Van Emden-Kowalski's immediate consequence operator $T_{\mathcal{P}}$ [7], and thus on the classical evaluation of negation (the evaluation of a negative literal $\neg A$ is given by the negation of the evaluation of A). That semantics can be defined as the least model of the program and computed as the least fixpoint of $\Phi_{\mathcal{P}}$ with respect to the knowledge order. Unfortunately

J. Fiala et al. (Eds.): MFCS 2004, LNCS 3153, pp. 513–524, 2004.

that semantics is often considered to be too weak in the sense that it does not provide enough knowledge.

A more informative approach has lead to the *stable model* approach [11], which defines a whole family of models, based on the so-called Gelfond-Lifschitz transform [8]. Informally, the main principle of that approach is the *separation of the role of positive and negative information*. That is, the negative literals of a program are first evaluated to obtain a negation-free program whose minimal Herbrand model can be computed. As a consequence, this separation avoids the natural management of classical negation which is a major feature of the Kripke-Kleene semantics of logic programs with negation. To overcome the fact that some programs have no stable model over the set of truth-values $\{\mathtt{f}, \mathtt{t}\}$, Przymusinski ([19,20]) extended the Gelfond-Lifschitz transform, and thus the notion of stable model semantics to allow partial stable models. Remarkably, one among these stable models, the *minimal* one according to the knowledge ordering, is often considered as the favorite one and, as shown in [20], is one-to-one related with the so-called *well-founded semantics* defined independently from the stable models in [22]. It is not unusual that, rather than to compute the whole set of stable models, one computes the well-founded semantics only.

The following step was to move from three-valued logics, allowing the representation of incomplete information, to the well-known four-valued set of truth-values $\mathcal{FOUR} = \{\mathtt{f}, \mathtt{t}, \bot, \top\}$, introduced by Belnap ([3]) to allow the representation of *inconsistency* (denoted $\top$) as well. This process of enlarging the set of truth-values culminated with Fitting's progressive work (e.g. [8,9]) on giving meaning to logic programs by relying on *bilattices* [12]. Bilattices, where $\mathcal{FOUR}$ is the simplest non-trivial one, play an important role in logic programming, and in knowledge representation in general, allowing in particular reasoning with inconsistency and uncertainty (see e.g. [1,4,6,14,15,16,17,18]). Fitting proposed an extension of the stable models family by extending the Gelfond-Lifschitz transform from logic programming over two- or three-valued logics to logic programming over bilattices [8]. As a consequence, that work proposed an extension to that setting of the well-founded semantics as well, as being the least fixpoint, with respect to a given knowledge order, of the extension of the Gelfond-Lifschitz transform.

The primary goal of this study is to show, in the quite general setting of bilattices as space of truth-values, that neither this separation of positive and negative information is necessary nor any program transformation is required to characterize the well-founded semantics. Indeed, we show that it can be defined as a simple, natural and epistemic extension of the Kripke-Kleene semantics. Informally, we view the CWA as an additional source of information to be used for information completion, or more precisely, as a carrier for falsehood, to be considered cumulatively to the Kripke-Kleene semantics. To this end, given a logic program $\mathcal{P}$ and an interpretation I representing our current knowledge about some intended model of $\mathcal{P}$, we define the notion of *support provided by the CWA to the program $\mathcal{P}$ with respect to I*. The support provided by the CWA to a program $\mathcal{P}$ with respect to an interpretation I extends the notion of unfounded

set [22] from three-valued logics to bilattices, and determines in a principled way how much *false* knowledge, i.e. how much knowledge provided by the CWA, can "safely" be joined to I with respect to the program $\mathcal{P}$. Then, we define the set of *supported models* to be the set of those models that can not be completed anymore by the CWA, i.e. that contains their own support. We provide different characterizations of that family of models in terms of an operator managing negation classically, i.e. in terms of $\Phi_{\mathcal{P}}$. Finally, we show that stable models are supported models and that the well-founded semantics coincides with the least supported model with respect to the knowledge order.

As a consequence, we propose alternative, both epistemic and fixpoint-based, characterizations of the well-founded semantics to the well-known, widely applied and long studied technique based on the separation of positive and negative information, by reverting to the classical interpretation of negation, i.e. we characterize negation-as-failure as standard negation. While Fitting treats negation-as-failure in a special way and unlike other connectives, our approach is an attempt to relate the semantics of logic programs to a standard model-theoretic account of rules. We emphasize the possibility to analyze logic programs using standard logical means as the notion of interpretation and information ordering, i.e. knowledge ordering. Therefore, our approach in principle does not depend on the presence of any specific connective, such as negation-as-failure, nor on any specific syntax of rules. Moreover our approach lights, in the general setting of logic programming over bilattices, the role of the CWA in the well-founded semantics. Due to the generality and the purely algebraic nature of our results, as just monotone operators over bilattices are postulated, the epistemic characterization of the well-founded semantics given in this study can be applied in other contexts as well (e.g. uncertainty and/or paraconsistency in logic programming, or nonmonotonic logics).

The remaining of the paper is organized as follows. In order to make the paper self-contained, in the next section, we will briefly recall definitions and properties of bilattices and logic programs. Section 3 is the main part of this work, where we present our characterizations of the well-founded semantics, while Section 4 concludes. Due to lack of space, proofs are omitted and are available from the authors' on-line version.

2 Preliminaries

2.1 Bilattices

The simplest non-trivial bilattice, called $\mathcal{FOUR}$, is due to Belnap ([3]), who introduced a logic intended to deal with incomplete and/or inconsistent information – see also [2]. $\mathcal{FOUR}$ already illustrates many of the basic properties concerning bilattices. Essentially, $\mathcal{FOUR}$ extends the classical truth set $\{\mathtt{f}, \mathtt{t}\}$ to its power set $\{\{\mathtt{f}\}, \{\mathtt{t}\}, \emptyset, \{\mathtt{f}, \mathtt{t}\}\}$, where we can think that each set indicates the amount of information we have in terms of truth: so, $\{\mathtt{f}\}$ stands for *false*, $\{\mathtt{t}\}$ for *true* and, quite naturally, $\emptyset$ for lack of information or *unknown*, and $\{\mathtt{f}, \mathtt{t}\}$ for *inconsistent* information (for ease, we use $\mathtt{f}$ for $\{\mathtt{f}\}$, $\mathtt{t}$ for $\{\mathtt{t}\}$, $\bot$ for $\emptyset$ and

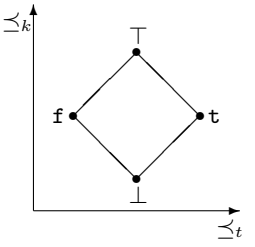

Fig. 1. The logic $\mathcal{FOUR}$.

$\top$ for $\{\mathtt{f}, \mathtt{t}\}$). $\mathcal{FOUR}$ has two quite intuitive and natural 'orthogonal' orderings, $\preceq_k$ and $\preceq_t$ (see Figure 1), each giving to $\mathcal{FOUR}$ the structure of a complete lattice. One is the so-called *knowledge ordering*, denoted $\preceq_k$, and is based on the subset relation, that is, if $x \subseteq y$ then x represents 'more information' than y (e.g. $\bot = \emptyset \subseteq \{\mathtt{t}\} = \mathtt{t}$, i.e. $\bot \preceq_k \mathtt{t}$). The other ordering is the so-called *truth ordering*, denoted $\preceq_t$. Here $x \preceq_t y$ means that x is 'at least as false as y is, and y is at least as true as x is', i.e. $x \cap \{\mathtt{t}\} \subseteq y \cap \{\mathtt{t}\}$ and $y \cap \{\mathtt{f}\} \subseteq x \cap \{\mathtt{f}\}$ (e.g. $\bot \preceq_t \mathtt{t}$).

Formally [12,9], a *bilattice* is a structure $\langle \mathcal{B}, \preceq_t, \preceq_k \rangle$ where $\mathcal{B}$ is a non-empty set and $\preceq_t$ and $\preceq_k$ are both partial orderings giving $\mathcal{B}$ the structure of a *complete lattice* with a top and bottom element. *Meet (or greatest lower bound)* and *join (or least upper bound)* under $\preceq_t$, denoted $\wedge$ and $\vee$, correspond to extensions of classical conjunction and disjunction. On the other hand, *meet and join under* $\preceq_k$ are denoted $\otimes$ and $\oplus$. $x \otimes y$ corresponds to the maximal information x and y can agree on, while $x \oplus y$ simply combines the information represented by x with that represented by y. *Top and bottom under* $\preceq_t$ are denoted $\mathtt{t}$ and $\mathtt{f}$, and *top and bottom under* $\preceq_k$ are denoted $\top$ and $\bot$, respectively. We will assume that bilattices are *infinitary distributive bilattices* in which all distributive laws connecting $\wedge, \vee, \otimes$ and $\oplus$ hold. We also assume that every bilattice satisfies the *infinitary interlacing conditions*, i.e. each of the lattice operations $\wedge, \vee, \otimes$ and $\oplus$ is monotone w.r.t. both orderings. Finally, we assume that each bilattice has a *negation*, i.e. an operator $\neg$ that reverses the $\preceq_t$ ordering, leaves unchanged the $\preceq_k$ ordering, and verifies $\neg\neg x = x$.

2.2 Logic Programs and Models

We recall here the definitions given in [8]. This setting is as general as possible, so that the results proved in this paper will be widely applicable.

Logic programs. Consider an alphabet of predicate symbols, of constants, of function symbols and variable symbols. A *term*, t, is either a variable x, a constant c or of the form $f(t_1, \ldots, t_n)$, where f is an n-ary function symbol and all t_i are terms. An *atom*, A, is of the form $p(t_1, \ldots, t_n)$, where p is an n-ary predicate symbol and all t_i are terms. A literal, l, is of the form A or $\neg A$, where

A is an atom. A *formula*, φ, is an expression built up from the literals and the members of a bilattice $\mathcal{B}$ using $\wedge, \vee, \otimes, \oplus, \exists$ and $\forall$. Note that members of the bilattice may appear in a formula, e.g. in $\mathcal{FOUR}$, $(p \wedge q) \oplus (r \otimes \mathtt{f})$ is a formula. A *rule* is of the form $p(x_1, \ldots, x_n) \leftarrow \varphi(x_1, \ldots, x_n)$, where p is an n-ary predicate symbol and all x_i are variables. The atom $p(x_1, \ldots, x_n)$ is called the *head*, and the formula $\varphi(x_1, \ldots, x_n)$ is called the *body*. It is assumed that the free variables of the body are among $x_1, \ldots, x_n$. Free variables are thought of as universally quantified. A *logic program*, denoted with $\mathcal{P}$, is a finite set of rules. The *Herbrand universe* of $\mathcal{P}$ is the set of *ground* (variable-free) terms that can be built from the constants and function symbols occurring in $\mathcal{P}$, while the *Herbrand base* of $\mathcal{P}$ (denoted $B_{\mathcal{P}}$) is the set of ground atoms over the Herbrand universe.

Given $\mathcal{P}$, the set $\mathcal{P}^*$ is constructed as follows; (*i*) put in $\mathcal{P}^*$ all ground instances of members of $\mathcal{P}$ (over the Herbrand base); (*ii*) if a ground atom A is not head of any rule in $\mathcal{P}^*$, then add the rule $A \leftarrow \mathtt{f}$ to $\mathcal{P}^*$; [1] and (*iii*) replace several ground rules in $\mathcal{P}^*$ having same head, $A \leftarrow \varphi_1$, $A \leftarrow \varphi_2$, ... with $A \leftarrow \varphi_1 \vee \varphi_2 \vee \ldots$.[2] Note that in $\mathcal{P}^*$, each ground atom appears in the head of *exactly one* rule.

Interpretations. An *interpretation of a logic program* on the bilattice $\langle \mathcal{B}, \preceq_t, \preceq_k \rangle$ is a mapping from ground atoms to members of $\mathcal{B}$. An interpretation I is extended from atoms to formulae as follows: (*i*) for $b \in \mathcal{B}$, $I(b) = b$; (*ii*) for formulae φ and φ', $I(\varphi \wedge \varphi') = I(\varphi) \wedge I(\varphi')$, and similarly for $\vee, \otimes, \oplus$ and $\neg$; and (*iii*) $I(\exists x \varphi(x)) = \bigvee \{I(\varphi(t)) : t \text{ ground term}\}$, and similarly for universal quantification[3]. The truth and knowledge orderings are extended from $\mathcal{B}$ to the set $\mathcal{I}(\mathcal{B})$ of all interpretations as follows: (*i*) $I_1 \preceq_t I_2$ iff $I_1(A) \preceq_t I_2(A)$, for every ground atom A; and (*ii*) $I_1 \preceq_k I_2$ iff $I_1(A) \preceq_k I_2(A)$, for every ground atom A. We define $(I_1 \wedge I_2)(\varphi) = I_1(\varphi) \wedge I_2(\varphi)$, and similarly for the other operations. With $\mathtt{I_f}$ and $\mathtt{I}_\bot$ denote the bottom interpretations under $\preceq_t$ and $\preceq_k$ respectively (they map any atom into $\mathtt{f}$ and $\bot$, respectively). It is easy to see that $\langle \mathcal{I}(\mathcal{B}), \preceq_t, \preceq_k \rangle$ is an infinitary interlaced and distributive bilattice as well.

Classical setting. Note that in a *classical logic program* the body is a conjunction of literals. Therefore, if $A \leftarrow \varphi \in \mathcal{P}^*$, then $\varphi = \varphi_1 \vee \ldots \vee \varphi_n$ and $\varphi_i = L_{i_1} \wedge \ldots \wedge L_{i_n}$. Furthermore, a *classical total interpretation* is an interpretation over $\mathcal{FOUR}$ such that an atom is mapped into either $\mathtt{f}$ or $\mathtt{t}$. A *partial classical interpretation* is a classical interpretation where the truth of some atom may be left unspecified. This is the same as saying that the interpretation maps all atoms into either $\mathtt{f}, \mathtt{t}$ or $\bot$. For a set of literals X, with $\neg . X$ we indicate the set $\{\neg L : L \in X\}$, where for any atom A, $\neg\neg A$ is replaced with A. Then, a classical interpretation (total or partial) can also be represented as a set of literals, and inversely.

[1] It is a standard practice in logic programming to consider such atoms as *false*.

[2] There could be infinitely many grounded rules with same head, but the semantics behavior is unproblematic.

[3] The bilattice is complete w.r.t. $\preceq_t$, so existential and universal quantification are well-defined.

Models. An interpretation I is a *model* of a logic program $\mathcal{P}$, denoted by $I \models \mathcal{P}$, if and only if for each rule $A \leftarrow \varphi$ in $\mathcal{P}^*$, $I(\varphi) \preceq_t I(A)$.

Fitting ([8,9]) identifies a set of models which obeys the so-called *Clark-completion* procedure [5], by mainly replacing each occurrence of $\leftarrow$ in $\mathcal{P}^*$ with $\leftrightarrow$: an interpretation I is a *Clark-completion model*, *cl-model* for short, of a logic program $\mathcal{P}$, denoted by $I \models_{cl} \mathcal{P}$, iff for each rule $A \leftarrow \varphi$ in $\mathcal{P}^*$, $I(A) = I(\varphi)$. It can easily be shown that Clark-completion models have also an alternative characterization given by $I \models_{cl} \mathcal{P}$ iff $I = \min_{\preceq_t}\{J : J \models \mathcal{P}[I]\}$, where $\mathcal{P}[I] = \{A \leftarrow I(\varphi) : A \leftarrow \varphi \in \mathcal{P}^*\}$.

Program k-completions. We introduce here the notion of program *knowledge completion*, or simply, k-completion with I, denoted $\mathcal{P} \oplus I$. The idea is to enforce any model J of $\mathcal{P} \oplus I$ to contain at least the knowledge determined by $\mathcal{P}$ and I. That is, the *k-completion* of $\mathcal{P}$ with I, is the program obtained by replacing any rule of the form $A \leftarrow \varphi \in \mathcal{P}$ by $A \leftarrow \varphi \oplus I(A)$.

2.3 Semantics of Logic Programs

The semantics of a logic program $\mathcal{P}$ is usually determined by selecting a particular model, or a set of models, of $\mathcal{P}$. We recall the definitions of the three most popular and widely studied semantics for logic programs with negation in increasing order of knowledge.

The Kripke-Kleene semantics. The Kripke-Kleene semantics [10] has a simple, intuitive and epistemic characterization. The *Kripke-Kleene model* of $\mathcal{P}$, denoted $KK(\mathcal{P})$, is the $\preceq_k$-least cl-model of $\mathcal{P}$, i.e.$KK(\mathcal{P}) = \min_{\preceq_k}(\{I : I \models_{cl} \mathcal{P}\})$.

The Kripke-Kleene semantics has also an alternative, and better known, fixpoint characterization, by relying on the well-known $\Phi_{\mathcal{P}}$ immediate consequence operator.

Definition 1 ($\Phi_{\mathcal{P}}$). *The* immediate consequence operator $\Phi_{\mathcal{P}} : \mathcal{I}(\mathcal{B}) \rightarrow \mathcal{I}(\mathcal{B})$ *is defined as follows: for any ground atom A such that $A \leftarrow \varphi$ occurs in $\mathcal{P}^*$, $\Phi_{\mathcal{P}}(I)(A) = I(\varphi)$.*

$\Phi_{\mathcal{P}}$ is a generalization of the Van Emden-Kowalski's immediate consequence operator $T_{\mathcal{P}}$ [7] to bilattices under Clark's program completion. Interesting properties of $\Phi_{\mathcal{P}}$ are that (*i*) $\Phi_{\mathcal{P}}$ relies on the classical evaluation of negation, i.e. the evaluation of a negative literal $\neg A$ is given by the negation of the evaluation of A; (*ii*) the cl-models of a program $\mathcal{P}$ coincide with the fixpoints of $\Phi_{\mathcal{P}}$; and (*iii*) $\Phi_{\mathcal{P}}$ is monotone with respect to the knowledge ordering and, thus, has a $\preceq_k$-least fixpoint, which coincides with the Kripke-Kleene semantics of $\mathcal{P}$. Note that $\Phi_{\mathcal{P}}(I) = \min_{\preceq_t}\{J : J \models \mathcal{P}[I]\}$.

As a consequence, *all definitions and properties given in this paper in terms of $\Phi_{\mathcal{P}}$ and/or cl-models may be given in terms of models as well.* As $\Phi_{\mathcal{P}}$ is a well-known operator, for ease of presentation we will continue to rely on it.

Well-founded semantics and stable models. The original definition of the well-founded semantics [22] was formulated for classical logic programs. That

definition relies on the immediate consequence operator $T_{\mathcal{P}}$ over the set of partial interpretations, defined by $T_{\mathcal{P}}(I) = \{A \mid \exists A \leftarrow L_1, ..., L_n \in \mathcal{P} \ (\forall L_i \ (L_i \in I))\}$, for inferring positive information, and on the notion of *unfounded set* to complete that information with negative information. A set of instantiated atoms U is said to be unfounded with respect to a partial interpretation I if for all instantiated atoms $A \in U$ and for all rules $r \in P$ the following holds: $head(r) = A \Rightarrow \exists L \in body(r) \ (\neg L \in I \quad or \quad L \in U)$. Intuitively, a set of atoms is unfounded if every way to infer an atom in that set, i.e. every rule with such an atom in head, fails because some atom in the condition, i.e. in the body of the rule, is in contradiction with the current knowledge and/or is itself unfounded. The *well-founded semantics* of P is then defined to be the least fixpoint of the well-founded operator W_P, defined by $W_P(I) = T_P(I) \cup \neg.U_P(I)$, with respect to set inclusion, where $U_P(I)$ is the $\subseteq$-greatest unfounded set with respect to I.

The extension of the notions of the well-founded semantics to the context of bilattices is due to Fitting [8], through the fact that the least partial stable model coincides with the well-founded semantics [20]. He proposes a generalization of the Gelfond-Lifschitz transformation [11,20], based on the same basic principle that consists in separating the roles of positive and negative information. Formally, let I and J be two interpretations in $\langle \mathcal{I}(\mathcal{B}), \preceq_t, \preceq_k \rangle$. The *pseudo-interpretation* $I \triangle J$ over the bilattice is defined as follows: for a pure ground atom A, $(I \triangle J)(A) = I(A)$ and $(I \triangle J)(\neg A) = \neg J(A)$. Pseudo-interpretations are extended to non-literals in the obvious way. The *immediate consequence operator* $\Psi_{\mathcal{P}}: \mathcal{I}(\mathcal{B}) \times \mathcal{I}(\mathcal{B}) \rightarrow \mathcal{I}(\mathcal{B})$ is then defined by: for any ground atom A such that $A \leftarrow \varphi$ occurs in $\mathcal{P}^*$, $\Psi_{\mathcal{P}}(I, J)(A) = (I \triangle J)(\varphi)$. Using the monotonicity of $\Psi_{\mathcal{P}}$ in its first argument with respect to $\preceq_t$, Fitting defined the *stability operator* $\Psi'_{\mathcal{P}}: \mathcal{I}(\mathcal{B}) \rightarrow \mathcal{I}(\mathcal{B})$ as follows : $\Psi'_{\mathcal{P}}(I)$ is the $\preceq_t$-least fixpoint of the operator $\lambda x.\Psi_{\mathcal{P}}(x, I)$, i.e. $\Psi'_{\mathcal{P}}(I) = \mathrm{lfp}_{\preceq_t}(\lambda x.\Psi_{\mathcal{P}}(x, I))$.

Finally, following Fitting's formulation, a *stable model* for a logic program $\mathcal{P}$ is a fixpoint of $\Psi'_{\mathcal{P}}$. The operator $\Psi'_{\mathcal{P}}$ is monotone in the $\preceq_k$ ordering and, thus, has a $\preceq_k$-least fixpoint, denoted $WF(\mathcal{P})$. By definition, $WF(\mathcal{P})$ is known as the *well-founded model* of $\mathcal{P}$.

3 Well-Founded Semantics Revisited

In the previous sections we have seen that, while for the Kripke-Kleene semantics there is an intuitive epistemic and model theory-based characterization, for the well-founded semantics on bilattices this is likely not the case. In the following, we present our contribution, which mainly consists in defining both epistemic and fixpoint-based characterizations of the well-founded semantics over bilattices. We contribute, thus, to an alternative view of well-founded semantics over bilattices, to the well-known and long studied separation of positive and negative information in Fitting's computation.

We will rely on the following running example.

Example 1 (running example) *Consider the following logic program $\mathcal{P}$ with the following rules.*

$$p \leftarrow p$$
$$q \leftarrow \neg r$$
$$r \leftarrow \neg q \wedge \neg p$$

In the following table, we report the different interpretations and models studied in this paper: cl-models, supports, Kripke-Kleene (K) and well-founded (W) semantics, stable and supported models of $\mathcal{P}$.

	I_i			$s_\mathcal{P}(I_i)$						
$I_i \models_{cl} \mathcal{P}$	p	q	r	p	q	r	K	W	*stable models*	*supported models*
I_1	$\bot$	$\bot$	$\bot$	f	$\bot$	$\bot$	•			
I_2	$\bot$	t	f	f	$\bot$	f				
I_3	f	$\bot$	$\bot$	f	$\bot$	$\bot$		•	•	•
I_4	f	f	t	f	f	$\bot$			•	•
I_5	f	t	f	f	$\bot$	f			•	•
I_6	f	$\top$	$\top$	f	f	f			•	•
I_7	t	t	f	f	$\bot$	f				
I_8	$\top$	t	f	f	$\bot$	f				•
I_9	$\top$	$\top$	$\top$	f	f	f				•

3.1 Support

First we introduce the notion of *support*, denoted $s_\mathcal{P}(I)$, provided by the CWA to a logic program $\mathcal{P}$ with respect to a given interpretation I. Intuitively, we regard I as what we already know about an intended model of $\mathcal{P}$. On the basis of both the current knowledge I and the information expressed by the rules of $\mathcal{P}$, we want to complete our available knowledge I, by using the CWA. We regard the CWA as an additional source of information for *falsehood*. The support $s_\mathcal{P}(I)$ of $\mathcal{P}$ w.r.t. I determines in a principled way the maximal amount of falsehood provided by the CWA that can be "safely" joined to I. The main principle underlying safe interpretations can be explained as follows. For ease, let us consider $\mathcal{FOUR}$. Consider an interpretation I, which is our current knowledge about $\mathcal{P}$. Let us assume that the interpretation J, with $J \preceq_k \mathtt{I_f}$, indicates which atoms may be assumed as f. For any ground atom A, $J(A)$ is the default 'false' information provided by J to the atom A. The completion of I with J is the interpretation $I \oplus J$. In order to accept this completion, we have to ensure that the assumed false knowledge about A, $J(A)$, is entailed by $\mathcal{P}$ w.r.t. the completed interpretation $I \oplus J$. In other words, for $A \leftarrow \varphi \in \mathcal{P}^*$, assuming that A is false is safe if φ is false as well with respect to the current knowledge I completed by the assumed falsehood J, i.e. $J(A) \preceq_k (I \oplus J)(\varphi)$ should hold.

Definition 2 (safe interpretation). *Let $\mathcal{P}$ and I be a logic program and an interpretation, respectively. An interpretation J is* safe *w.r.t. $\mathcal{P}$ and I iff:*

1. $J \preceq_k \mathtt{I_f}$;
2. $J \preceq_k \Phi_\mathcal{P}(I \oplus J)$.

In the above definition, the first item dictates that any safe interpretation is a carrier of falsehood, i.e. a part of the CWA. If $J = \mathtt{I_f}$, then every ground atom is false. But, given I and $\mathcal{P}$, not necessarily all atoms can be considered as false (e.g., some atoms may be inferred true from the program) and we have to consider some weaker assumption $J \preceq_k \mathtt{I_f}$ of falsehood. The second item dictates that a safe interpretation is *cumulative*, i.e. as we proceed in deriving more precise approximations of an intended model of $\mathcal{P}$, the accumulated falsehood should be preserved. To illustrate that concept, consider the interpretation I_2 of our running example. I_2 dictates that p is unknown, q is true and that r is false. Consider the interpretations J_i defined as follows:

J_i	p	q	r
J_1	$\bot$	$\bot$	$\bot$
J_2	$\mathtt{f}$	$\bot$	$\bot$
J_3	$\bot$	$\bot$	$\mathtt{f}$
J_4	$\mathtt{f}$	$\bot$	$\mathtt{f}$

It is easy to verify that all the J_is are safe. The $\preceq_k$-least safe interpretation is J_1, while the $\preceq_k$-greatest one is $J_4 = J_1 \oplus J_2 \oplus J_3$. J_4 dictates that under I_2, we can 'safely' assume that both p and r are false. Note that if we join J_4 to I_2 we obtain the stable model I_5, where $I_2 \preceq_k I_5$. J_4 improves the knowledge expressed by I_2. One might wonder why we do not consider q false as well. Indeed, if we consider p, q and r false, after joining to I and applying $\Phi_{\mathcal{P}}$, we have that q becomes true, which is *knowledge-incompatible* with q's previous knowledge status (q is false). So, q's falsehood is not preserved, i.e. cumulative.

To give another intuitive reading of that notion, as anticipated, safe interpretations have an interesting reading once we restrict our attention to the classical framework of logic programming: indeed, the concept of safe interpretation reduces to that of unfounded set.

Theorem 1. *Let $\mathcal{P}$ and I be a classical logic program and a classical interpretation, respectively. Let X be a subset of $B_{\mathcal{P}}$. Then X is an unfounded set of $\mathcal{P}$ w.r.t. I iff $\neg.X \preceq_k \Phi_{\mathcal{P}}(I \oplus \neg.X)$* [4]*, i.e. $\neg.X$ is safe w.r.t. $\mathcal{P}$ and I.*

The safe assumptions are parts of the CWA that can be added to the current knowledge to complete it, thus among all possible safe interpretations, we privilege the maximal one under $\preceq_k$ in order to infer as much knowledge as possible.

Definition 3 (support). *Let $\mathcal{P}$ be a logic program and I an interpretation. The* support provided by the CWA *to $\mathcal{P}$ w.r.t. I, or simply* support *of $\mathcal{P}$ w.r.t. I, denoted $s_{\mathcal{P}}(I)$, is the $\preceq_k$-greatest safe interpretation w.r.t. $\mathcal{P}$ and I.*

It is easy to show that the support is a well-defined concept: given two safe interpretations J and J', then $J \oplus J' \preceq_k \mathtt{I_f}$ and, from the monotonicity of $\Phi_{\mathcal{P}}$ under $\preceq_k$, $J \oplus J' \preceq_k \Phi_{\mathcal{P}}(I \oplus J \oplus J')$ and, thus, $J \oplus J'$ is safe. Therefore, $s_{\mathcal{P}}(I) = \bigoplus\{J\colon J \text{ is safe w.r.t. } \mathcal{P} \text{ and } I\}$. We give also an alternative fixpoint characterization of the support to the model-theoretical one, and thus an effective method to compute it.

[4] Note that this condition can be rewritten as $\neg.X \subseteq \Phi_{\mathcal{P}}(I \cup \neg.X)$.

Theorem 2. *Consider the iterated sequence of interpretations defined by: for any $i \geq 0$,*

$$F_0^I = \mathtt{I_f}, \quad \textit{and}$$
$$F_{i+1}^I = \mathtt{I_f} \otimes \Phi_{\mathcal{P}}(I \oplus F_i^I).$$

The sequence F_i^I is (i) monotone non-increasing under $\preceq_k$ and, thus, reaches a fixpoint F_ω^I, for a limit ordinal ω; and (ii) monotone non-decreasing under $\preceq_t$. Furthermore, $s_{\mathcal{P}}(I) = F_\omega^I$ holds.

Note that, for classical logic programs, the notions of support and greatest unfounded set are closely related as the set of (negative) literals corresponding to $s_{\mathcal{P}}(I)$ coincides with the negation of the greatest unfounded set $U_{\mathcal{P}}(I)$. It follows that the above theorem leads to a new fixpoint computation of $U_{\mathcal{P}}(I)$ based on $\Phi_{\mathcal{P}}$ only. Indeed, $\neg.U_{\mathcal{P}}(I)$ coincides with the limit of the sequence:

$$F_0^I = \neg.B_{\mathcal{P}}$$
$$F_{i+1}^I = \neg.B_{\mathcal{P}} \cap \Phi_{\mathcal{P}}(I \cup F_i^I)$$

Note also that the support can be seen as a monotone operator over the space of interpretations w.r.t. $\preceq_k$.

Theorem 3. *The support operator $s_{\mathcal{P}}$ is monotone w.r.t. $\preceq_k$. Moreover, if $I \preceq_k J$, then $s_{\mathcal{P}}(J) \preceq_t s_{\mathcal{P}}(I)$.*

Theorem 3 has an intuitive reading: the more knowledge we have about a ground atom A, the more we *know* (the more precise and informative we are) about A's falsehood, i.e. the more falsehood can be provided by the CWA to A.

3.2 Supported Models

Among all possible models of a program $\mathcal{P}$, we will be especially interested in the models I that integrate their own support ($s_{\mathcal{P}}(I) \preceq_k I$). Such models tell us that we have reached the point where the additional source for falsehood provided by the CWA can not contribute further to improve our knowledge about $\mathcal{P}$.

Definition 4 (supported model). *Consider a logic program $\mathcal{P}$. An interpretation I is a* supported model *of $\mathcal{P}$ iff $I \models_{cl} \mathcal{P}$ and $s_{\mathcal{P}}(I) \preceq_k I$.*

Restricting our attention to classical logic programs, a partial interpretation is a supported models of $\mathcal{P}$ iff I is a cl-model containing the negation of its greatest unfounded set, i.e. $I \models_{cl} \mathcal{P}$ and $\neg.U_{\mathcal{P}}(I) \subseteq I$. Supported models have interesting properties :

Theorem 4. *The following are equivalent:*

1. *I is a supported model of $\mathcal{P}$;*
2. *$I = \Phi_{\mathcal{P}}(I) \oplus s_{\mathcal{P}}(I)$;*
3. *$I \models_{cl} \mathcal{P} \oplus s_{\mathcal{P}}(I)$;*
4. *$I = \Phi_{\mathcal{P}}(I \oplus s_{\mathcal{P}}(I))$.*

The above theorem states in different ways the same concept: supported models contain the amount of knowledge expressed by the program and their support. Thus, from a fixpoint characterization point of view, the set of supported models can be identified by the fixpoints of the monotone immediate consequence operator $\Pi_{\mathcal{P}}$ defined by $\Pi_{\mathcal{P}}(I) = \Phi_{\mathcal{P}}(I \oplus s_{\mathcal{P}}(I))$ (an equivalent definition, in terms of fixpoints, is $\Pi_{\mathcal{P}}(I) = \Phi_{\mathcal{P}}(I) \oplus s_{\mathcal{P}}(I)$, which could be seen as an extension to bilattices of the well-founded operator $W_{\mathcal{P}}$). Thus, the set of supported models is a complete lattice under $\preceq_k$, and we show finally that the notion of supported models provide new characterizations of the well-founded semantics.

Theorem 5. *Stable models of $\mathcal{P}$ are supported models of $\mathcal{P}$, and the $\preceq_k$-least supported model of $\mathcal{P}$ coincides with the well-founded semantics of $\mathcal{P}$.*

Therefore $WF(\mathcal{P})$ can be computed by iterations of $\Pi_{\mathcal{P}}$ starting from $\mathtt{I}_{\perp}$. This lights the role of the CWA that can be seen as a source of falsehood used to complete the Kripke-Kleene semantics. Indeed, the well-founded semantics can be obtained by iterating $\Phi_{\mathcal{P}}$ from $\mathtt{I}_{\perp}$ (similarly to the Kripke-Kleene semantics), but adding to each iteration some knowledge provided by the CWA. Apart obtaining alternative epistemic characterization and computation method of the well-founded semantics to $\Psi'_{\mathcal{P}}$, the above theorem highlights the fact that, in the general context of logic programming over bilattices, neither a separation of positive and negative information is necessary, nor any program transformation is required for defining and computing the well-founded semantics.

4 Conclusions

In this study we have presented alternative formulations of the well-founded semantics. Our approach is purely based on algebraic and semantical aspects of informative monotone operators over bilattices. In this sense, we talk about epistemic foundation of the well-founded semantics. The main concept we rely on is based on the fact that we regard the closed world assumption as an additional source for falsehood and identify with the *support* the amount of falsehood carried on by the closed world assumption. We have shown that the well-founded semantics can be characterized as the knowledge minimal model containing its own support, i.e. $WF(\mathcal{P}) = \min_{\preceq_k}(\{I : I \models \mathcal{P} \text{ and } s_{\mathcal{P}}(I) \preceq_k I\})$. This indicates that the support may be seen as the added-value to the Kripke-Kleene semantics and lights the role of the CWA in the well-founded semantics over bilattices. It also shows that neither a separation of positive and negative information nor any program transformation is necessary.

References

1. O. Arieli. Paraconsistent declarative semantics for extended logic programs. *Annals of Mathematics and Artificial Intelligence*, 36(4):381–417, 2002.
2. O. Arieli and A. Avron. The value of the four values. *Artificial Intelligence Journal*, 102(1):97–141, 1998.

3. N. D. Belnap. A useful four-valued logic. In G. Epstein and J. M. Dunn, editors, *Modern uses of multiple-valued logic*, pages 5–37. Reidel, Dordrecht, NL, 1977.
4. H. Blair and V. S. Subrahmanian. Paraconsistent logic programming. *Theoretical Computer Science*, 68:135–154, 1989.
5. K.L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and data bases*, pages 293–322. Plenum Press, New York, NY, 1978.
6. C. V. Damásio and L. M. Pereira. A survey of paraconsistent semantics for logic programs. In D. Gabbay and P. Smets, editors, *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, pages 241–320. Kluwer, 1998.
7. M. H. Van Emden and R. A. Kowalski. The semantics of predicate logic as a programming language. *Journal of the ACM (JACM)*, 23(4):733–742, 1976.
8. M. C. Fitting. The family of stable models. *Journal of Logic Programming*, 17:197–225, 1993.
9. M. C. Fitting. Fixpoint semantics for logic programming - a survey. *Theoretical Computer Science*, 21(3):25–51, 2002.
10. M. Fitting. A Kripke-Kleene-semantics for general logic programs. *Journal of Logic Programming*, 2:295–312, 1985.
11. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In R. A. Kowalski and K. Bowen, editors, *Proc. of the 5th Int. Conf. on Logic Programming*, pages 1070–1080. The MIT Press, 1988.
12. M. L. Ginsberg. Multi-valued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4:265–316, 1988.
13. K. Kunen. Negation in logic programming. *Journal of Logic Programming*, 4(4):289–308, 1987.
14. Y. Loyer and U. Straccia. Uncertainty and partial non-uniform assumptions in parametric deductive databases. In *Proc. of the 8th European Conf. on Logics in Artificial Intelligence (JELIA-02)*, LNCS 2424, pages 271–282. Springer-Verlag, 2002.
15. Y. Loyer and U. Straccia. The approximate well-founded semantics for logic programs with uncertainty. In *28th Int. Sym. on Mathematical Foundations of Computer Science (MFCS-2003)*, LNCS 2747, pages 541–550. Springer-Verlag, 2003.
16. Y. Loyer and U. Straccia. Default knowledge in logic programs with uncertainty. In *Proc. of the 19th Int. Conf. on Logic Programming (ICLP-03)*, LNCS 2916, pages 466–480. Springer Verlag, 2003.
17. T. Lukasiewicz. Fixpoint characterizations for many-valued disjunctive logic programs with probabilistic semantics. In *In Proc. of the 6th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR-01)*, LNAI 2173, pages 336–350. Springer-Verlag, 2001.
18. R. Ng and V.S. Subrahmanian. Stable model semantics for probabilistic deductive databases. In Z. W. Ras and M. Zemenkova, editors, *Proc. of the 6th Int. Sym. on Methodologies for Intelligent Systems (ISMIS-91)*, LNAI 542, pages 163–171. Springer-Verlag, 1991.
19. T. C. Przymusinski. Extended stable semantics for normal and disjunctive programs. In D. H. D. Warren and P. Szeredi, editors, *Proc. of the 7th Int. Conf. on Logic Programming*, pages 459–477. MIT Press, 1990.
20. T. C. Przymusinski. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae*, 13(4):445–463, 1990.
21. R. Reiter. On closed world data bases. In Hervé Gallaire and Jack Minker, editors, *Logic and data bases*, pages 55–76. Plenum Press, New York, NY, 1978.
22. A. van Gelder, K. A. Ross, and J. S. Schlimpf. The well-founded semantics for general logic programs. *Journal of the ACM*, 38(3):620–650, 1991.