Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 342 (2005) 351–381
Theoretical Computer Science
www.elsevier.com/locate/tcs

Any-world assumptions in logic programming

Yann Loyer[a], Umberto Straccia[b,*]

[a] *Laboratoire PRiSM, Université de Versailles Saint Quentin, Versailles, France*
[b] *Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", Consiglio Nazionale delle Ricerche, Pisa 56124, Italy*

Received 21 July 2004; received in revised form 3 February 2005; accepted 18 April 2005

Communicated by G. Levi

Abstract

Due to the usual incompleteness of information representation, any approach to assign a semantics to logic programs has to rely on a default assumption on the missing information. The *stable model semantics*, that has become the dominating approach to give semantics to logic programs, relies on the Closed World Assumption (CWA), which asserts that by default the truth of an atom is *false*. There is a second well-known assumption, called *Open World Assumption* (OWA), which asserts that the truth of the atoms is supposed to be *unknown* by default. However, the CWA, the OWA and the combination of them are extremal, though important, assumptions over a large variety of possible assumptions on the truth of the atoms, whenever the truth is taken from *an arbitrary truth space*.

The topic of this paper is to allow *any* assignment (i.e. interpretation), over a truth space, to be a default assumption. Our main result is that our extension is conservative in the sense that under the "everywhere false" default assumption (CWA) the usual stable model semantics is captured. Due to the generality and the purely algebraic nature of our approach, it abstracts from the particular formalism of choice and the results may be applied in other contexts as well.

Keywords: Logic programming; Bilattices; Default reasoning

* Corresponding author. Tel.: +39 050 315 2894; fax: +39 050 315 2810.
E-mail addresses: Yann.Loyer@prism.uvsq.fr (Y. Loyer), straccia@isti.cnr.it (U. Straccia).

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.04.005

1. Introduction

The incompleteness of information representation is a well-known phenomena in knowledge representation and reasoning. As a consequence, one usually relies on some default assumption on the missing information, to complete the intended meaning of the represented information. For instance, in logic programming, the *stable model semantics* [20], which is likely the most widely studied and most commonly accepted approach to give meaning to logic programs, relies on the *Closed World Assumption* (CWA) [46] to *complete* the available knowledge. The CWA assumes that *all* atoms not entailed by a program are *false*, and is motivated by the fact that explicit representation of negative information in logic programs, and in knowledge representation languages in general, is not feasible because the addition of explicit negative information could overwhelm a system. Another well-known assumption is the so-called *Open World Assumption* (OWA), which asserts that the default truth value of *every* atom is supposed to be *unknown*. For instance, the OWA is used for many biological databases that explicitely contain incomplete knowledge. Moreover in such a context, as some microorganisms have been completely sequenced, while many others have not, one would like to have the ability to specify the OWA or the CWA over particular "regions" of the database. However, the CWA, the OWA and the combination of them [1], which find their application in *Extended Logic Programming* (see e.g. [2]), are extremal, though important, assumptions over a large variety of possible assumptions on the truth of the atoms, whenever the truth is taken from *an arbitrary truth space*.

The main topic of this study is a *generalization* of the use of assumptions in logic programming. That is, rather than to rely on the same default truth value for all atoms ('false' under CWA, 'unknown' under OWA) we allow *any interpretation* over a given truth space to be a default assumption, to be used to complete the meaning of a logic program. For instance, while integrating information coming from different sources, some sources may be considered as less reliable than others. In such a case, a distinction could be made between "supposed" and "sure" knowledge. Indeed the knowledge provided by less reliable sources should be considered as supposed knowledge to be used to complete, but without introducing contradictions, the knowledge provided by the other sources that should be considered as sure. So, e.g. an insurance company may rely on a priori computed statistics and/or on information provided by another insurance company to complete its own available information about a new client in order to compute the risk coefficient (for more examples, see Section 4).

In summary, our main results in this paper are that (i) our extension is conservative in the sense that under the "everywhere false" default assumption (CWA) the usual stable model semantics is captured; and (ii) due to the generality and the purely algebraic nature of our approach, it abstracts from the particular formalism of choice and the results may be applied in other, non-logic programming, contexts as well.

Our presentation is structured as follows. In the next section, we briefly recall some preliminary notions. In Section 3, we define epistemic and fixed-point characterizations of the semantics that can be associated with any logic program with respect to any given

[1] With "combination of OWA and CWA" we mean that *some* atoms truth is by default unknown, while for the others the truth is by default false.

world assumption. We also show that our approach captures the usual semantics of logic programs, and that, as a consequence, our approach provides new characterizations of the stable models semantics. In Section 4 we provide some examples of uses of the newly defined semantics, while Section 5 concludes, mentions related work and gives an outlook for further research. Proofs are given in the Appendix.

2. Preliminaries

The truth spaces we consider are the so-called *bilattices* [23], which are a much richer structure than the classical {f, t} space.

Due to their interesting mathematical structure, bilattices play an important role in (especially in theoretical aspects of) logic programming, and in knowledge representation in general, allowing to develop unifying semantical frameworks. Notably, our setting conforms to Fitting's general logic programming framework [15,16,18].

Informally, a bilattice is a non-empty, possibly infinite set of truth-values provided with two 'orthogonal' partial orders, each one giving to the set of truth values the structure of a lattice.

In the sequel we define the main notions related to lattices, bilattices, logic programs and describe the usual semantics associated to logic programs.

2.1. Lattices and bilattices

A *lattice* is denoted as $\langle L, \preccurlyeq\rangle$, where $\preccurlyeq$ is a partial order over the non-empty set L. We write $x \prec y$ if $x \preccurlyeq y$ and $x \neq y$. The notions of *least upper bound* (also, *join*) of $x, y \in L$, and that of *greatest lower bound* (also, *meet*) of x and y are as usual. We assume that lattices are *complete*, i.e. every subset of L has both a least upper and a greatest lower bound. With $\bot$ and $\top$ we denote the least element and the greatest element of a lattice, respectively. For ease, given $S \subseteq L$, with $\preccurlyeq$-*least* and $\preccurlyeq$-*greatest* element w.r.t. S we always mean the greatest lower and the least upper bound, respectively. With $\min_{\preccurlyeq}(S)$ we denote the set of minimal elements in S, i.e. $\{x \in S : \nexists y \in S \ s.t. \ y \prec x\}$. If $\{x\} = \min_{\preccurlyeq}(S)$ (if there is an unique minimal element), for convenience we may also write $x = \min_{\preccurlyeq}(S)$. A function (also called *operator*) from L to L is *monotone*, iff for all $x, y \in L$, $x \preccurlyeq y$ implies $f(x) \preccurlyeq f(y)$, while f is *antitone* if $x \preccurlyeq y$ implies $f(y) \preccurlyeq f(x)$. A *fixed-point* of f is an element $x \in L$ such that $f(x) = x$.

The basic tool for studying fixed-points of operators on lattices is the well-known Knaster–Tarski theorem [49], which establishes that a monotone operator $f : L \to L$ has a fixed-point, the set of fixed-points of f is a complete lattice and, thus, f has a $\preccurlyeq$-*least* and a $\preccurlyeq$-*greatest* fixed-point. The $\preccurlyeq$-*least* (respectively, $\preccurlyeq$-*greatest*) fixed-point can be obtained by iterating f over $\bot$ (respectively, $\top$).

A *bilattice* is a structure $\langle \mathcal{B}, \preccurlyeq_t, \preccurlyeq_k\rangle$ where $\mathcal{B}$ is a non-empty set and $\preccurlyeq_t$ and $\preccurlyeq_k$ are both partial orderings giving $\mathcal{B}$ the structure of a *complete lattice* with a top and bottom element (see, e.g. [23]). *Meet and join under* $\preccurlyeq_t$, denoted $\wedge$ and $\vee$, correspond to extensions of classical conjunction and disjunction. On the other hand, *meet and join under* $\preccurlyeq_k$ are denoted $\otimes$ and $\oplus$. $x \otimes y$ corresponds to the maximal information x and y can agree on,

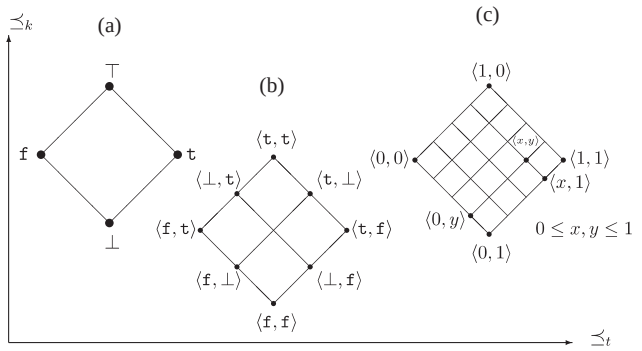

Fig. 1. Bilattices: (a) $\mathcal{FOUR}$, (b) $\{\mathtt{f}, \bot, \mathtt{t}\} \odot \{\mathtt{f}, \bot, \mathtt{t}\}$ and (c) $\mathcal{K}([0, 1])$.

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.

The simplest non-trivial bilattice, called $\mathcal{FOUR}$ (see Fig. 1), is due to Belnap [6], who introduced a logic intended to deal with incomplete and/or inconsistent information. $\mathcal{FOUR}$ already illustrates many of the basic properties concerning bilattices. Essentially, $\mathcal{FOUR}$ extends the classical truth set $\{\mathtt{f}, \mathtt{t}\}$ to $\{\mathtt{f}, \mathtt{t}, \bot, \top\}$, where $\bot$ stands *unknown*, and $\top$ stands for *inconsistent*.

The two orders are the so-called *knowledge ordering* $\preceq_k$ and the *truth ordering* $\preceq_t$. If $x \preceq_k y$ then y represents 'more information' than x. On the other hand, if $x \preceq_t y$ then y represents 'more truth' than x. For instance, in $\mathcal{FOUR}$, $\bot \preceq_k \mathtt{f} \preceq_k \top$, $\bot \preceq_k \mathtt{t} \preceq_k \top$, $\mathtt{f} \preceq_t \bot \preceq_t \mathtt{t}$ and $\mathtt{f} \preceq_t \top \preceq_t \mathtt{t}$.

Furthermore, we 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 (e.g. $x \preceq_t y$ and $x' \preceq_t y'$ implies $x \otimes x' \preceq_t y \otimes y'$). 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 [2] $\neg\neg x = x$.

Bilattices come up in natural ways. Indeed, there are two general, but different, construction methods, which allow to build a bilattice from a lattice and are widely used. We just sketch them here in order to give a feeling of their application (see also [15,23]).

The first bilattice construction method comes from [23]. Suppose we have two complete distributive lattices $\langle L_1, \preceq_1\rangle$ and $\langle L_2, \preceq_2\rangle$. Think of L_1 as a lattice of values we use when we measure the degree of belief of a statement, while think of L_2 as the lattice we use when we measure the degree of doubt of it. Now, we define the structure $L_1 \odot L_2$ as follows. The structure is $\langle L_1 \times L_2, \preceq_t, \preceq_k\rangle$, where

- $\langle x_1, x_2\rangle \preceq_t \langle y_1, y_2\rangle$ if $x_1 \preceq_1 y_1$ and $y_2 \preceq_2 x_2$,
- $\langle x_1, x_2\rangle \preceq_k \langle y_1, y_2\rangle$ if $x_1 \preceq_1 y_1$ and $x_2 \preceq_2 y_2$.

[2] The dual operation to negation is *conflation* i.e. an operator $\sim$ that reverses the $\preceq_k$ ordering, leaves unchanged the $\preceq_t$ ordering, and $\sim\sim x = x$. We do not deal with conflation in this paper.

In $L_1 \odot L_2$ the idea is: knowledge goes up if both degree of belief and degree of doubt go up; truth goes up if the degree of belief goes up, while the degree of doubt goes down. It is easily verified that $L_1 \odot L_2$ is a bilattice. Furthermore, if $L_1 = L_2 = L$, i.e. we are measuring belief and doubt in the same way, then negation can be defined as $\neg\langle x, y\rangle = \langle y, x\rangle$. That is, negation switches the roles of belief and doubt. In Fig. 1 we report the bilattice based on $L_1 = L_2 = \{\mathtt{f}, \bot, \mathtt{t}\}$ and order $\preccurlyeq_1 = \preccurlyeq_2 = \preccurlyeq$, where $\mathtt{f} \preccurlyeq \bot \preccurlyeq \mathtt{t}$.

The second construction method has been sketched in [23] and addressed in more details in [19], and is probably the more used one. Suppose we have a complete distributive lattice of truth values $\langle L, \preccurlyeq\rangle$ (like e.g. in Many-valued Logics [24]). Think of these values as the 'real' values we are interested in, but due to lack of knowledge we are able just to 'approximate' the exact values. That is, rather than considering a pair $\langle x, y\rangle \in L \times L$ as indicator for degree of belief and doubt, $\langle x, y\rangle$ is interpreted as the set of elements $z \in L$ such that $x \preccurlyeq z \preccurlyeq y$. Therefore, a pair $\langle x, y\rangle$ is interpreted as an *interval*. An interval $\langle x, y\rangle$ may be seen as an approximation of an exact value. For instance, in reasoning under uncertainty (see, e.g. [34–36]), L is the unit interval $[0, 1]$ with standard ordering, $L \times L$ is interpreted as the set of (closed) sub-intervals of $[0, 1]$, and the pair $\langle x, y\rangle$ is interpreted as a lower and an upper bound of the exact value of the certainty value. Formally, given a distributive lattice $\langle L, \preccurlyeq\rangle$, the *bilattice of intervals*, denoted $\mathcal{K}(L)$, is $\langle L \times L, \preccurlyeq_t, \preccurlyeq_k\rangle$, where:

- $\langle x_1, x_2\rangle \preccurlyeq_t \langle y_1, y_2\rangle$ if $x_1 \preccurlyeq y_1$ and $x_2 \preccurlyeq y_2$,
- $\langle x_1, x_2\rangle \preccurlyeq_k \langle y_1, y_2\rangle$ if $x_1 \preccurlyeq y_1$ and $y_2 \preccurlyeq x_2$.

The intuition of those orders is that truth increases if the interval contains greater values, whereas the knowledge increases when the interval becomes more precise. Negation can be defined as $\neg\langle x, y\rangle = \langle \neg y, \neg x\rangle$, where $\neg$ is a negation operator on L. As an example, in Fig. 1 we report the bilattice $\mathcal{K}([0, 1])$.

In practice bilattices has been used in several ways. For instance, Arieli and Avron show [4,5] that the use of four values is preferable to the use of two or three values even for tasks that can in principle be handled using only three values. The algebraic work of Fitting's fixed-point characterisation of stable model semantics on bilattices has been the root of the work carried out by Denecker et al. [11–13], who extended Fitting's work to a more abstract context of fixed-points operators on lattices, by relying on *interval* bilattices. Denecker et al. showed [11,13] interesting connections between (two-valued and four-valued) Kripke–Kleene [17], well-founded and stable model semantics, as well as to Moore's autoepistemic logic [43] and Reiter's default logic [47]. Other well-established applications of bilattices and/or Kripke–Kleene, well-founded and stable models semantics to give semantics to logic programs can be found in the context of reasoning under paraconsistency and uncertainty (see, e.g. [1,3,7,9,10,33–36]). In particular, "belief-doubt" bilattices are used in paraconsistent logic programming [1,9] and anti-tonic logic programming [10], while "interval" bilattices are used in work like [11–13,34–36].

2.2. Logic programs

We follow the definitions of Fitting [15,16]. Consider an alphabet of predicate symbols, of constants, of function symbols and variable symbols. A *term*, t, is inductively defined as usual: 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 the logical operators $\wedge, \vee, \otimes, \oplus, \exists$ 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*. Note that the body may be *any* formula. 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. Note that, e.g. rules with terms in the head, like

$$\begin{aligned} p(s(x)) &\leftarrow p(x), \\ p(0) &\leftarrow \mathtt{t} \end{aligned}$$

may be rewritten in our context as

$$\begin{aligned} p(y) &\leftarrow \exists x(eq(y, s(x)) \wedge p(x)), \\ p(y) &\leftarrow eq(y, 0), \end{aligned}$$

where *eq* is a predicate defining equality.

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 a logic program $\mathcal{P}$, with $\mathcal{P}^*$ we denote the ground instantiation of $\mathcal{P}$ obtained as follows: let us denote with *ground*$(\mathcal{P})$ all ground instances of members of $\mathcal{P}$ (over the Herbrand Universe), then

(1) put in $\mathcal{P}^*$ all ground instances of members of $\mathcal{P}$ (over the Herbrand Universe), i.e. include *ground*$(\mathcal{P})$ in $\mathcal{P}^*$,
(2) replace several ground rules in $\mathcal{P}^*$ having same head, $A \leftarrow \varphi_1$, $A \leftarrow \varphi_2, \ldots$ with $A \leftarrow \varphi_1 \vee \varphi_2 \vee \ldots$. Note that as there could be infinitely many grounded rules with same head, we may end with a countable disjunction, but the semantics behavior is unproblematic; and
(3) additionally, if a ground atom A is not head of any rule in $\mathcal{P}^*$, then the rule $A \leftarrow \mathtt{f}$ is added to $\mathcal{P}^*$. Note that it is a standard practice in logic programming to consider such atoms as *false*. We incorporate this by explicitly adding $A \leftarrow \mathtt{f}$ to $\mathcal{P}^*$. This already acts as a kind of default assumption on non-derivable facts. We will change this point once we allow any default value as assumption later one.

Therefore, *in $\mathcal{P}^*$ all atoms of the Herbrand base appear in the head of exactly one rule*. For instance, for the program $\mathcal{P}$ above, for any ground term t, $\mathcal{P}^*$ contains a unique rule with the atom $p(t)$ in head:

$$p(t) \leftarrow (\exists x(eq(t, s(x)) \wedge p(x))) \vee eq(t, 0).$$

We next specify the usual semantics of logic programs over bilattices. Indeed, we define the notions of model, Kripke–Kleene model, well-founded model and stable model of $\mathcal{P}$. For ease, we will rely on the following simple running example to illustrate the concepts we introduce in the paper.

Table 1
Models, Kripke–Kleene, well-founded and stable models of $\mathcal{P}$

$I_i \models \mathcal{P}$	I_i		$KK(\mathcal{P})$	$WF(\mathcal{P})$	Stable models	$s_{\mathcal{P}}^{\mathtt{I_f}}(I_i)$		$U_{\mathcal{P}}(I_i)$	H-founded models	H-closed models
	p	q				p	q			
I_1	$\bot$	$\bot$	•	•	•	$\bot$	$\bot$	$\emptyset$	•	•
I_2	t	$\bot$				$\bot$	$\bot$	$\emptyset$	•	
I_3	$\top$	$\top$			•	f	f	$\{p, q\}$	•	•
I_4	t	$\top$				f	f	$\{p, q\}$		

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

$$p \leftarrow p \vee q,$$
$$q \leftarrow \neg q.$$

In Table 1 we report the models I_i, the Kripke–Kleene, the well-founded and the stable models of $\mathcal{P}$, marked by bullets. The columns on the right-hand side will be discussed later on.

Let $\langle \mathcal{B}, \preceq_t, \preceq_k \rangle$ be a bilattice. By *interpretation of a logic program* on the bilattice we mean a mapping I from ground atoms to members of $\mathcal{B}$. An interpretation I is extended from atoms to formulae in the usual way: (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 family of all interpretations is denoted by $\mathcal{I}(\mathcal{B})$. The truth and knowledge orderings are extended from $\mathcal{B}$ to $\mathcal{I}(\mathcal{B})$ point-wise 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. Given two interpretations I, J, we define $(I \wedge J)(\varphi) = I(\varphi) \wedge J(\varphi)$, and similarly for the other operations. With $\mathtt{I_f}$ and $\mathtt{I_t}$ we denote the bottom and top interpretations under $\preceq_t$ (they map any atom into f and t, respectively). With $\mathtt{I}_\bot$ and $\mathtt{I}_\top$ we denote the bottom and top interpretations under $\preceq_k$ (they map any atom into $\bot$ and $\top$, respectively). It is easy to see that the space of interpretations $\langle \mathcal{I}(\mathcal{B}), \preceq_t, \preceq_k \rangle$ is an infinitary interlaced and distributive bilattice as well.

An interpretation I is a *model* of a logic program $\mathcal{P}$ (see [15,16]), denoted by $I \models \mathcal{P}$, iff for all $A \leftarrow \varphi \in \mathcal{P}^*$, $I(A) = I(\varphi)$ holds. The above definition of model follows the so-called *Clark-completion* procedure [8], where we replace in $\mathcal{P}^*$ each occurrence of $\leftarrow$ with $\leftrightarrow$. Indeed, usually a model has to satisfy $I(\varphi) \preceq_t I(A)$ only, i.e. $A \leftarrow \varphi \in \mathcal{P}^*$ specifies the necessary condition on A, "A is at least as true as φ". Under the Clark-completion, the constraint becomes also sufficient, i.e. the unique rule involving A in $\mathcal{P}^*$ *completely* defines A. Note that the condition $I(A) = I(\varphi)$ may also be seen as the result of a typical *truth minimization* process, i.e. given the unique rule $A \leftarrow \varphi \in \mathcal{P}^*$, in logic programming one usually tries to minimize the truth of A. More formally, let $\mathcal{P}$ and I be a logic program and

[3] As we will see below, the bilattice is complete w.r.t. $\preceq_t$, so existential and universal quantification are well-defined.

an interpretation, respectively. Then

$$I \models \mathcal{P} \text{ iff } I = \min_{\preccurlyeq_t} \{J : I(\varphi) \preccurlyeq_t J(A), \text{ for all } A \leftarrow \varphi \in \mathcal{P}^*\}. \quad (1)$$

In a *classical logic program* the body is a conjunction of literals. Therefore, if $A \leftarrow \varphi \in \mathcal{P}^*$ (except for the case $A \leftarrow \mathtt{f} \in \mathcal{P}^*$), 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 consistent set of literals, i.e. $I \subseteq B_\mathcal{P} \cup \neg.B_\mathcal{P}$ and for all atoms A, $\{A, \neg A\} \not\subseteq I$. Of course, the opposite is also true, i.e. a consistent set of literals can straightforwardly be turned into an interpretation over $\mathcal{FOUR}$.

Given an interpretation I, we introduce the notion of program *knowledge completion*, or simply, *k*-completion with I, denoted $\mathcal{P} \oplus I$. The *program k-completion* of $\mathcal{P}$ with I, is the program obtained by replacing all rules $A \leftarrow \varphi \in \mathcal{P}^*$ by $A \leftarrow \varphi \oplus I(A)$. For instance, given $\mathcal{P}$ and I_2 in Example 1, $\mathcal{P} \oplus I_2$ is $\{p \leftarrow (p \vee q) \oplus \mathtt{t}, q \leftarrow (\neg q) \oplus \bot\}$. Essentially, the idea is to enforce any model J of $\mathcal{P} \oplus I$ to be such that for a given rule $A \leftarrow \varphi \in \mathcal{P}^*$, $J(A)$ carries as much knowledge as determined by $\mathcal{P}$ and I, i.e. $J(\varphi) \oplus I(A)$. Note that the notion of *k*-completion has no analogue in classical logic programming as $\oplus$ is not allowed as a construct. *k*-completions will play an important role in our formalization.

2.3. Usual semantics of logic programs

Usually the semantics of a program $\mathcal{P}$ is determined by selecting a particular interpretation, or a set of interpretations, of $\mathcal{P}$ in the set of models of $\mathcal{P}$. We consider three semantics, which are likely the most popular and widely studied semantics for logic programs, namely the *Kripke–Kleene semantics*, the *well-founded semantics* and the *stable model semantics*, in increasing order of knowledge [15–17,20,21,50].

2.3.1. Kripke–Kleene semantics

The Kripke–Kleene semantics [15,17] has a simple and intuitive characterization, as it corresponds to the least model of a logic program under the knowledge order $\preccurlyeq_k$, i.e. the *Kripke–Kleene model* of a logic program $\mathcal{P}$ is $KK(\mathcal{P}) = \min_{\preccurlyeq_k}\{I : I \models \mathcal{P}\}$. Note that, in the light of Eq. (1), the Kripke–Kleene semantics is a composition of a truth-minimization and then of a knowledge minimization. The *existence and uniqueness* of $KK(\mathcal{P})$ is guaranteed by the fixed-point characterization below, by means of the *immediate consequence operator* $\Phi_\mathcal{P}$. For an interpretation I, for any ground atom A

$$\Phi_\mathcal{P}(I)(A) = I(\varphi),$$

where[4] $A \leftarrow \varphi \in \mathcal{P}^*$.

[4] Recall that all ground atoms are head of exactly one rule in $\mathcal{P}^*$.

It can be shown that (see [15]) (i) in the space of interpretations, the operator $\Phi_{\mathcal{P}}$ is monotone under $\preccurlyeq_k$, (ii) the set of fixed-points of $\Phi_{\mathcal{P}}$ is a complete lattice under $\preccurlyeq_k$ and, thus, $\Phi_{\mathcal{P}}$ has a $\preccurlyeq_k$-least (and $\preccurlyeq_k$-greatest) fixed-point; and (iii) I is a model of a program $\mathcal{P}$ iff I is a fixed-point of $\Phi_{\mathcal{P}}$. Therefore, the Kripke–Kleene model of $\mathcal{P}$ coincides with the least fixed-point of $\Phi_{\mathcal{P}}$ under $\preccurlyeq_k$, which can be computed in the usual way by iterating $\Phi_{\mathcal{P}}$ over $\mathtt{I}_\perp$.

In this paper, we will use the following property, which can easily be shown. Let $\mathcal{P}$ be a logic program and let J and I be interpretations. Then

$$\Phi_{\mathcal{P}\oplus I}(J) = \Phi_{\mathcal{P}}(J) \oplus I. \tag{2}$$

In particular, $J \vDash \mathcal{P} \oplus I$ iff $J = \Phi_{\mathcal{P}}(J) \oplus I$ holds.

2.3.2. Stable model semantics

The *stable model semantics* approach, has been defined first by Gelfond and Lifschitz [20] with respect to the classical two valued truth space $\{\mathtt{f}, \mathtt{t}\}$ and extended by Fitting to bilattices [15,16]. Informally, an interpretation I is a *stable model* of a logic program $\mathcal{P}$ if $I = I'$, where I' is computed according to the so-called *Gelfond–Lifschitz transformation*:

(1) substitute (fix) in $\mathcal{P}^*$ the negative literals by their evaluation with respect to I. Let $\mathcal{P}^I$ be the resulting *positive* program, called *reduct* of $\mathcal{P}$ w.r.t. I; and
(2) compute the *truth-minimal* model I' of $\mathcal{P}^I$.

For instance, given $\mathcal{P}$ and I_1 in Example 1, $\mathcal{P}^{I_1}$ is $\{p \leftarrow p \vee q, q \leftarrow \perp\}$, whose $\preccurlyeq_t$-least model is I_1. Therefore, I_1 is a stable model. On the other hand, $\mathcal{P}^{I_2} = \mathcal{P}^{I_1}$, whose $\preccurlyeq_t$-least model is I_1, so I_2 is *not* a stable model.

Note that the main principle of this transformation is based on the *separation of the role of positive and negative information*. As a consequence, this separation avoids the natural management of classical negation (i.e. the evaluation of a negative literal $\neg A$ is given by the negation of the evaluation of A), which is a major feature of the Kripke–Kleene semantics [17,18] of logic programs with negation.

Formally, Fitting [15,16] relies on a binary immediate consequence operator $\Psi_{\mathcal{P}}$, which accepts two input interpretations over a bilattice, the first one is used to assign meanings to positive literals, while the second one is used to assign meanings to negative literals. Let I and J be two interpretations in the bilattice $\langle \mathcal{I}(\mathcal{B}), \preccurlyeq_t, \preccurlyeq_k \rangle$. The notion of *pseudo-interpretation* $I \triangle J$ over the bilattice is defined as follows (I gives meaning to positive literals, while J gives meaning to negative literals): for a pure ground atom A:

$$\begin{aligned} (I \triangle J)(A) &= I(A), \\ (I \triangle J)(\neg A) &= \neg J(A). \end{aligned}$$

Pseudo-interpretations are extended to non-literals in the obvious way.[5] For instance, $(I \triangle J)(\neg A \wedge B) = (I \triangle J)(\neg A) \wedge (I \triangle J)(B) = \neg J(A) \wedge I(B)$. We can now define $\Psi_{\mathcal{P}}$ as follows. For $I, J \in \mathcal{I}(\mathcal{B})$, $\Psi_{\mathcal{P}}(I, J)$ is the interpretation, which for any ground atom A is such that

$$\Psi_{\mathcal{P}}(I, J)(A) = (I \triangle J)(\varphi),$$

[5] Note that negation may appear in front of a literal only.

where $A \leftarrow \varphi \in \mathcal{P}^*$. Note that $\Phi_{\mathcal{P}}$ is a special case of $\Psi_{\mathcal{P}}$, as by construction $\Phi_{\mathcal{P}}(I) = \Psi_{\mathcal{P}}(I, I)$.

It can be shown that (see [15]) in the space of interpretations the operator $\Psi_{\mathcal{P}}$ is monotone in both arguments under $\preccurlyeq_k$, and under the ordering $\preccurlyeq_t$ it is monotone in its first argument and antitone in its second argument.

To define the stable model semantics, Fitting [15] introduces the $\Psi'_{\mathcal{P}}$ operator, whose fixed-points will be the stable models of a program. For any interpretation I, $\Psi'_{\mathcal{P}}(I)$ is the $\preccurlyeq_t$-least fixed-point of the operator $\lambda x.\Psi_{\mathcal{P}}(x, I)$, i.e.

$$\Psi'_{\mathcal{P}}(I) = \mathrm{lfp}_{\preccurlyeq_t}(\lambda x.\Psi_{\mathcal{P}}(x, I)).$$

Due to the $\preccurlyeq_t$-monotonicity, $\Psi'_{\mathcal{P}}$ is well defined. Additionally, (i) the operator $\Psi'_{\mathcal{P}}$ is monotone in the $\preccurlyeq_k$ ordering, and antitone in the $\preccurlyeq_t$ ordering; and (ii) every fixed-point of $\Psi'_{\mathcal{P}}$ is also a fixed-point of $\Phi_{\mathcal{P}}$, i.e. a model of $\mathcal{P}$. Finally, a *stable model* for a logic program $\mathcal{P}$ is a fixed-point of $\Psi'_{\mathcal{P}}$.

The set of fixed-points of $\Psi'_{\mathcal{P}}$, i.e. the set of stable models of $\mathcal{P}$, is a complete lattice under $\preccurlyeq_k$ and, thus, $\Psi'_{\mathcal{P}}$ has a $\preccurlyeq_k$-least (and $\preccurlyeq_k$-greatest) fixed-point, which is denoted $WF(\mathcal{P})$. $WF(\mathcal{P})$ is known as the *well-founded model* of $\mathcal{P}$ and, by definition coincides with the $\preccurlyeq_k$-least stable model, i.e. $WF(\mathcal{P}) = \min_{\preccurlyeq_k}(\{I : I \text{ stable model of } \mathcal{P}\})$.

Concerning the truth order, in [37] it is shown that stable models are incomparable with each other with respect to $\preccurlyeq_t$, i.e. given two stable models I and J such that $I \neq J$, then $I \not\preccurlyeq_t J$ and $J \not\preccurlyeq_t I$.

Finally, $\Psi'_{\mathcal{P}}(I)$ can be computed by iterating $\lambda x.\Psi_{\mathcal{P}}(x, I)$ starting from the everywhere false interpretation $\mathtt{I_f}$, while the $\preccurlyeq_k$-least stable model (i.e. well-founded model) and the $\preccurlyeq_k$-greatest stable model can be computed by iterating $\Psi'_{\mathcal{P}}$ starting from $\mathtt{I}_\bot$ and $\mathtt{I}_\top$, respectively.

It is interesting to note, and we will largely refer to it in this paper, that for classical logic programs there is also an alternative equivalent definition of $WF(\mathcal{P})$ based on the well-known notion of *unfounded set* (see, e.g. [30,50]). The underlying principle of the notion of unfounded sets is to identify the set of atoms that can safely be assumed false if the current information about a logic program is given by an interpretation I. Indeed, given a partial classical interpretation I and a classical logic program $\mathcal{P}$, a set of ground atoms $X \subseteq B_{\mathcal{P}}$ is an *unfounded set* (i.e., the atoms in X can be assumed as false) for $\mathcal{P}$ w.r.t. I iff for each atom $A \in X$, if $A \leftarrow \varphi \in \mathcal{P}^*$, where $\varphi = \varphi_1 \vee \ldots \vee \varphi_n$ and $\varphi_i = L_{i_1} \wedge \ldots \wedge L_{i_n}$, then φ_i is false either w.r.t. I or w.r.t. $\neg.X$, for all $1 \leqslant i \leqslant n$.

A well-known property of unfounded sets is that the union of two unfounded sets of $\mathcal{P}$ w.r.t. I is an unfounded set as well and that there is a unique *greatest unfounded set* for $\mathcal{P}$ w.r.t. I, denoted by $U_{\mathcal{P}}(I)$. See Table 1 for the greatest unfounded sets w.r.t. the models of the logic program in Example 1.

Now, consider the usual immediate consequence operator $T_{\mathcal{P}}$, where for any ground atom A,

$$T_{\mathcal{P}}(I)(A) = \mathtt{t} \text{ iff there is } A \leftarrow \varphi \in \mathcal{P}^* \text{ s.t. } I(\varphi) = \mathtt{t},$$

and consider the well-founded operator [50] over partial classical interpretations I, defined as

$$W_{\mathcal{P}}(I) = T_{\mathcal{P}}(I) \cup \neg.U_{\mathcal{P}}(I). \tag{3}$$

$W_{\mathcal{P}}(I)$ can be rewritten as $W_{\mathcal{P}}(I) = T_{\mathcal{P}}(I) \oplus \neg.U_{\mathcal{P}}(I)$, by defining $\oplus = \cup$, $\otimes = \cap$ in the lattice $\langle 2^{B_{\mathcal{P}} \cup \neg.B_{\mathcal{P}}}, \subseteq \rangle$ (the partial order $\subseteq$ corresponds to the knowledge order $\preceq_k$). Then, the well-founded semantics is defined to be the $\preceq_k$-least fixed-point of $W_{\mathcal{P}}$ in [50], and it is shown in [30] that the set of *total* stable models of $\mathcal{P}$ coincides with the set of total fixed-points of $W_{\mathcal{P}}$. In particular, this formulation reveals that the greatest unfounded set, $\neg.U_{\mathcal{P}}(I)$, is the additional "false default knowledge", which is brought into by the CWA to the usual semantics of logic programs based on $T_{\mathcal{P}}$. However, $W_{\mathcal{P}}$ does not characterize partial stable models. Indeed, there are partial interpretations being fixed-points of $W_{\mathcal{P}}(I)$, but are not stable models.

As we will see in the next section, the notion of unfounded set and the $W_{\mathcal{P}}$ operator are more suitable to be extended to the case where any interpretation is considered as the default, rather than relying on the Gelfond–Lifschitz transform and, thus, on the $\Psi'_{\mathcal{P}}$ operator, which are 'hard-wired' on the everywhere false assumption.

This concludes the preliminary part.

3. The AWA in logic programming

In the following, a *hypothesis* (denoted H) is always an interpretation over a bilattice and represents our default assumption over the world. The principle underlying the *Any-World Assumption* (AWA) H is to regard H as an additional source of default information to be used to complete the implicit knowledge provided by a logic program. The AWA H dictates that any atom A, whose truth-value cannot be inferred from the facts and rules, is assigned to the default truth value $H(A)$. For comparison, under the CWA every atom has default truth value false, so $H = \mathtt{I}_{\mathtt{f}}$ is assumed, while under the OWA, by default we have no information (*knowledge*) about the atoms truth, and, thus we may assume that the default truth value of an atom is $\bot$ (no knowledge), i.e. $H = \mathtt{I}_{\bot}$ is assumed.

We have seen in the previous section that any ground atom A not appearing in the head of any rule and, thus, not derivable, is mapped into 'false' compliant with the CWA. We added $A \leftarrow \mathtt{f}$ to $\mathcal{P}^*$. Now, according to the AWA, any such atom A is mapped into $H(A)$ and, thus, we should add $A \leftarrow H(A)$ to $\mathcal{P}^*$ rather than $A \leftarrow \mathtt{f}$. *If not specified otherwise, in the following we will always assume that given a hypothesis H, a program $\mathcal{P}$ and a ground atom A not appearing in the head of the ground instantiation of $\mathcal{P}$, we change Point 3 of the definition of $\mathcal{P}^*$ and always add $A \leftarrow H(A)$ to $\mathcal{P}^*$*. It should be noted that this implicitly affects also all definitions based on $\mathcal{P}^*$, e.g. the definitions of model, Kripke–Kleene model, k-completion and that of $\Phi_{\mathcal{P}}$ (which now maps such atoms into $H(A)$ rather than into $\mathtt{f}$).

Now, we proceed in three steps:

(1) In the next section, we introduce the notion of *support*, denoted $s^H_{\mathcal{P}}(I)$, provided by a hypothesis H to a program $\mathcal{P}$ w.r.t. an interpretation I. The support is a generalization of the notion of unfounded sets (which determines the atoms that can be assumed to be false) w.r.t. I. Indeed, $s^H_{\mathcal{P}}(I)$ determines the amount of default information, provided by

the AWA H, that can safely be joined to I to complete it. It turns out that for classical logic programs under the everywhere false hypothesis $H = \mathtt{I_f}$, the support coincides with the negation of the greatest unfounded set[6], i.e. $s^H_{\mathcal{P}}(I) = \neg.U_{\mathcal{P}}(I)$. Roughly speaking, we may say that the "unfounded atoms" are the only atoms whose truth value can be modified by their default value.

(2) Any model I of $\mathcal{P}$ $\preccurlyeq_k$-subsuming its support, i.e. $s^H_{\mathcal{P}}(I) \preccurlyeq_k I$, tells us that the additional source for default information cannot contribute further to improve our knowledge about the program. We call such models *H-founded models* of $\mathcal{P}$, which will be discussed in Section 3.2. H-founded models can be characterized as fixed-points of the operator

$$\tilde{\Pi}_{\mathcal{P}}(I) = \Phi_{\mathcal{P}}(I) \oplus s^H_{\mathcal{P}}(I),$$

which is very similar to the $W_{\mathcal{P}}$ operator in Eq. (3), but generalized to logic programming over bilattices w.r.t. any hypothesis H. H-founded models have an interesting property when $H = \mathtt{I_f}$. Indeed, as expected, in this case it can be shown that the $\preccurlyeq_k$-*least H-founded model is the well-founded model of* $\mathcal{P}$.

(3) Unfortunately, while for classical logical programs and *total* interpretations, $\tilde{\Pi}_{\mathcal{P}}(I)$ (under $H = \mathtt{I_f}$) characterizes total stable models (in fact, $\tilde{\Pi}_{\mathcal{P}} = W_{\mathcal{P}}$), this characterization is not true in the general case of interpretations over bilattices. Therefore, in Section 3.3, we further refine the class of H-founded models, by introducing the class of *H-closed models*. This class requires H-founded models to satisfy some minimality condition with respect to the knowledge order $\preccurlyeq_k$. Indeed, a H-closed model I has to be deductively closed according to the Kripke–Kleene semantics of the program k-completed with its support, i.e.

$$I = KK(\mathcal{P} \oplus s^H_{\mathcal{P}}(I)).$$

Therefore, we can identify the support as the added-value (in terms of knowledge), which is brought into by a default hypothesis with respect to the standard Kripke–Kleene semantics of $\mathcal{P}$. H-closed models have the desired property that, under the everywhere false hypothesis $H = \mathtt{I_f}$, the set of stable models (over bilattices) *coincides* with the set of H-closed models.

Example 2 (*running example cont.*). Consider Example 1. In Table 1 we also report the support, H-founded models and H-closed models. Note that stable models and H-closed models coincide. Similarly, the support and the negation of the greatest unfounded set coincide as well. We also consider the AWA $H = \mathtt{I_t}$ (see Table 2). Note that now under $H = \mathtt{I_t}$, contrary to the case $H = \mathtt{I_f}$, I_2 and I_4 are H-closed models of which I_2 is both the Kripke–Kleene and the well-founded model w.r.t. H. Essentially, the difference is in the truth of p, which in the former case is always $\mathtt{t}$, while in the latter case is $\bot$ (I_1) and $\top$ (I_3), respectively.

[6] Recall that $\neg.U_{\mathcal{P}}(I)$ is a set of negative literals and can be viewed as a 3-valued interpretation, assigning $\mathtt{f}$ to elements of $U_{\mathcal{P}}(I)$ and $\bot$ to all others.

Table 2
Models, H-founded models and H-closed models of $\mathcal{P}$ w.r.t. $H = \mathtt{I}_{\mathtt{t}}$

$I_i \models \mathcal{P}$	I_i		$s^H_{\mathcal{P}}(I_i)$		$KK(\mathcal{P} \oplus s^H_{\mathcal{P}}(I_i))$		H-founded models	H-closed models
	p	q	p	q	p	q		
I_1	$\bot$	$\bot$	$\mathtt{t}$	$\bot$	$\mathtt{t}$	$\bot$		
I_2	$\mathtt{t}$	$\bot$	$\mathtt{t}$	$\bot$	$\mathtt{t}$	$\bot$	•	•
I_3	$\top$	$\top$	$\mathtt{t}$	$\mathtt{t}$	$\mathtt{t}$	$\top$	•	
I_4	$\mathtt{t}$	$\top$	$\mathtt{t}$	$\mathtt{t}$	$\mathtt{t}$	$\top$	•	•

3.1. Support

The main notion we introduce here is that of *support* provided by a hypothesis H to a logic program $\mathcal{P}$ w.r.t. an interpretation I. If I represents what we already know about an intended model of $\mathcal{P}$, the support represents the $\preccurlyeq_k$-greatest amount of information provided by a hypothesis H, which can be joined to I in order to complete I. The main principle underlying the support can be explained as follows. Consider a ground atom A and its related rule $A \leftarrow \varphi \in \mathcal{P}^*$, an interpretation I, which is our current knowledge about $\mathcal{P}$, and an AWA H. We would like to determine how much default knowledge can be 'safely' taken from H to complete I. So, let us assume that $J \preccurlyeq_k H$ amounts to the default knowledge taken from H. $J(A)$ is the default 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 knowledge $J(A)$ is entailed by $\mathcal{P}$ w.r.t. the completed interpretation $I \oplus J$, i.e. for $A \leftarrow \varphi \in \mathcal{P}^*$, $J(A) \preccurlyeq_k (I \oplus J)(\varphi) = \Phi_{\mathcal{P}}(I \oplus J)(A)$ should hold. This notion is given by the following definitions.

Definition 3 (*safe interpretation*). Let $\mathcal{P}$, I and H be a logic program, an interpretation and a hypothesis, respectively. An interpretation J is *safe* w.r.t. $\mathcal{P}$, I and H iff:
(1) $J \preccurlyeq_k H$,
(2) $J \preccurlyeq_k \Phi_{\mathcal{P}}(I \oplus J)$.

Note that in the above definition, the first item dictates that any safe interpretation is a carrier of information, which is taken from the hypothesis H. As it is not realistic to expect that the whole hypothesis H can be considered as "consistent" with the program, only some parts of it will be taken ($J \preccurlyeq_k H$). For instance, in Example 1, w.r.t. $H = \mathtt{I}_{\mathtt{t}}$ and I_1, we can safely assume by default that the truth of p is $\mathtt{t}$ (i.e., $J(p) = \mathtt{t} \preccurlyeq_k \Phi_{\mathcal{P}}(I \oplus J)(p) = \mathtt{t}$), while the truth of q taken from the assumption H should be $\bot$ ($J(q) = \bot$) and not $\mathtt{t}$, as $J(q) = \mathtt{t}$ is not consistent with the program, i.e. $J(q) = \mathtt{t} \not\preccurlyeq_k \Phi_{\mathcal{P}}(I \oplus J)(q) = \mathtt{f}$.

Safe interpretations have an interesting reading once we restrict our attention to the classical framework of logic programming: the concept of safe hypothesis reduces to that of unfounded set.

Theorem 4. *Let $\mathcal{P}$ and I be a classical logic program and a partial classical interpretation, respectively. Consider the CWA $H = \mathtt{I}_{\mathtt{f}}$. Let X be a subset*[7] *of $B_{\mathcal{P}}$. Then X is an unfounded set of $\mathcal{P}$ w.r.t. I iff $\neg.X \preccurlyeq_k \Phi_{\mathcal{P}}(I \oplus \neg.X)$, i.e. $\neg.X$ is safe w.r.t. $\mathcal{P}$, I and H.*

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

Like for unfounded sets, among all possible safe interpretations w.r.t. $\mathcal{P}$, I and H, we are interested in the maximal one under $\preccurlyeq_k$, which is unique. The $\preccurlyeq_k$-greatest safe interpretation will be called the support provided by H to $\mathcal{P}$ w.r.t. I.

Definition 5 (*support*). Let $\mathcal{P}$, I and H be a logic program, an interpretation and a hypothesis, respectively. The *support provided by* H to $\mathcal{P}$ w.r.t. I, or simply *support* of $\mathcal{P}$ w.r.t. I, H, denoted $s^H_{\mathcal{P}}(I)$, is the $\preccurlyeq_k$-greatest safe interpretation J w.r.t. $\mathcal{P}$, I and H.

It is easy to show that the support is a well-defined concept. Consider $X = \{J : J \text{ is safe w.r.t. } \mathcal{P}, I \text{ and } H\}$. As the bilattice is a complete lattice under $\preccurlyeq_k$, $lub_{\preccurlyeq_k}(X) = \bigoplus_{J\in X} J$. Let us show that $\bar{J} = lub_{\preccurlyeq_k}(X)$ is safe w.r.t. $\mathcal{P}$, I and H. Now, consider $J \in X$. Therefore $J \preccurlyeq_k \bar{J}$. But, J is safe, so $J \preccurlyeq_k H$ and $J \preccurlyeq_k \Phi_{\mathcal{P}}(I \oplus J) \preccurlyeq_k \Phi_{\mathcal{P}}(I \oplus \bar{J})$ (by $\preccurlyeq_k$-monotonicity of $\Phi_{\mathcal{P}}$). As a consequence, both H and $\Phi_{\mathcal{P}}(I \oplus \bar{J})$ are upper bounds of X. But $\bar{J}$ is the least upper bound of X and, thus, $\bar{J} \preccurlyeq_k H$ and $\bar{J} \preccurlyeq_k \Phi_{\mathcal{P}}(I \oplus \bar{J})$ follows. That is, $\bar{J}$ is safe and the $\preccurlyeq_k$-greatest safe interpretation w.r.t. $\mathcal{P}$, I and H, and, thus, by definition $s^H_{\mathcal{P}}(I) = \bar{J} = lub_{\preccurlyeq_k}(X)$.

Note that it follows directly from the definition of safe interpretation that under the OWA $H = \mathtt{I}_\perp$, $s^H_{\mathcal{P}}(I) = \mathtt{I}_\perp$ holds for any interpretation I, i.e. no additional default knowledge can be inferred from the assumption, as expected.

Example 6. Let us consider the interval bilattice, obtained from the real unit interval $[0, 1]$, representing closed sub-intervals of $[0, 1]$. Consider the logic program $\mathcal{P}$ with rules

$$A \leftarrow B, C$$
$$C \leftarrow C, D$$
$$B \leftarrow \langle 0.7, 0.7\rangle$$
$$D \leftarrow \langle 0.9, 0.9\rangle$$

If we try to infer as much knowledge as possible from that program without using any default knowledge, i.e. by relying on the assumption $H = \mathtt{I}_\perp$, then we find that the truth of B and D are exactly 0.7 and 0.9 respectively, while the truth of A and C are at most 0.7 and 0.9 respectively, i.e. $I(A) = \langle 0, 0.7\rangle$, $I(B) = \langle 0.7, 0.7\rangle$, $I(C) = \langle 0, 0.9\rangle$ and $I(D) = \langle 0.9, 0.9\rangle$. Relying now on a more informative assumption H asserting that e.g. the value of C is by default at least 0.6, i.e. $H(C) = \langle 0.6, 1\rangle$, then we should come up with a more precise characterization of A and C by "adding" that assumed knowledge to $I : C$ is in $\langle 0.6, 0.9\rangle$, and consequently A is in $\langle 0.6, 0.7\rangle$. Indeed, consider the following table, with interpretations I, J_1, J_2, $\bar{J}$ and assumption H.

	A	B	C	D
I	$\langle 0, 0.7\rangle$	$\langle 0.7, 0.7\rangle$	$\langle 0, 0.9\rangle$	$\langle 0.9, 0.9\rangle$
H	$\langle 0.4, 0.5\rangle$	$\langle 0, 1\rangle$	$\langle 0.6, 1\rangle$	$\langle 0, 0\rangle$
J_1	$\langle 0.3, 0.7\rangle$	$\langle 0, 1\rangle$	$\langle 0.6, 1\rangle$	$\langle 0, 0.9\rangle$
J_2	$\langle 0.4, 0.8\rangle$	$\langle 0, 1\rangle$	$\langle 0.6, 1\rangle$	$\langle 0, 0.9\rangle$
$\bar{J}$	$\langle 0.4, 0.7\rangle$	$\langle 0, 1\rangle$	$\langle 0.6, 1\rangle$	$\langle 0, 0.9\rangle$

Then both J_1 and J_2 are safe w.r.t. P, I and H. It is easy to see that the $\preccurlyeq_k$-greatest safe interpretation, i.e. the support, is $\bar{J} = J_1 \oplus J_2$. Interestingly, note how $\bar{J}$ provides to I some additional information (i.e. better lower bounds) on the values of A and C, respectively.

Other examples of supports can be found in Tables 1 and 2.

It then follows immediately from Theorem 4 that in the classical setting the notion of greatest unfounded set is captured by the notion of support, i.e. the support tells us which atoms may safely be assumed as false, given a partial classical interpretation I, a classical logic program $\mathcal{P}$ and the everywhere false assumption (see Table 1).

Corollary 7. *Let $\mathcal{P}$ and I be a classical logic program and a partial classical interpretation, respectively. Then $s_{\mathcal{P}}^{H}(I) = \neg.U_{\mathcal{P}}(I)$, for $H = \mathtt{I_f}$.*

Therefore, the support is an extension of the notion of unfounded sets (i) to logic programming over bilattices; and (ii) to arbitrary default assumptions.

As next, we show how the support can effectively be computed as the iterated fixed-point of the function

$$\sigma_{\mathcal{P}}^{I,H}(J) = H \otimes \Phi_{\mathcal{P}}(I \oplus J).$$

It is easy to verify that $\sigma_{\mathcal{P}}^{I,H}$ is monotone w.r.t. $\preccurlyeq_k$, as only $\preccurlyeq_k$-monotone operators are involved.

Theorem 8. *Let $\mathcal{P}$, I and H be a logic program, an interpretation and a hypothesis, respectively. Consider the iterated sequence of interpretations $F_i^{I,H}$: for any $i \geqslant 0$,*

$$F_0^{I,H} = H,$$

$$F_{i+1}^{I,H} = \sigma_{\mathcal{P}}^{I,H}(F_i^{I,H}).$$

The sequence $F_i^{I,H}$ is monotone decreasing under $\preccurlyeq_k$ and, thus, reaches a fixed-point $F_\lambda^{I,H}$, for a limit ordinal λ. Furthermore, $s_{\mathcal{P}}^{H}(I) = F_\lambda^{I,H}$ holds.

Theorem 9. *Let $\mathcal{P}$ be a logic program. The support operator $s_{\mathcal{P}}^{H}(I)$ is monotone in its arguments I and H w.r.t. $\preccurlyeq_k$.*

Theorem 9 has an intuitive reading: it states that the more knowledge we have about a ground atom A, the more information can be provided by the AWA to A.

Interestingly, for a classical logic program $\mathcal{P}$ and a partial classical interpretation I, by Corollary 7, the above method gives us a simple top-down method to compute the negation of the greatest unfounded set, $\neg.U_{\mathcal{P}}(I)$, by means of the iteration

$$F_0 = \neg.B_{\mathcal{P}},$$

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

3.2. *H*-founded models

Among all possible models of a program $\mathcal{P}$, let us consider those models, which $\preccurlyeq_k$-subsume their own support, i.e. that could not be completed anymore by the AWA.

Definition 10 (*H-founded model*). Let $\mathcal{P}$ and H be a logic program and a hypothesis H, respectively. An interpretation I is a *H-founded model* of $\mathcal{P}$ iff $I \models \mathcal{P}$ and $s^H_{\mathcal{P}}(I) \preccurlyeq_k I$.

Example 11. Consider Example 6. It can be verified that the interpretation $\bar{I}$ such that $\bar{I}(A) = \langle 0.6, 0.7\rangle$, $\bar{I}(B) = \langle 0.7, 0.7\rangle$, $\bar{I}(C) = \langle 0.6, 0.9\rangle$, and $\bar{I}(C) = \langle 0.9, 0.9\rangle$ is a model of $\mathcal{P}$ such that its support is $\bar{J}$ and $\bar{J} \preccurlyeq_k \bar{I}$. Therefore, $\bar{I}$ is a H-founded model of $\mathcal{P}$.

If we consider the definition of the support in the classical setting, then H-founded models are such that [8] $\neg.U_{\mathcal{P}}(I) \subseteq I$, i.e. the false atoms provided by the unfounded set are already false in the interpretation I. Therefore, the CWA does not further contribute to improve the knowledge of I about the program $\mathcal{P}$.

H-founded models have interesting properties, as stated below.

Theorem 12. *Let $\mathcal{P}$, I and H be a logic program, an interpretation and a hypothesis, respectively. The following statements are equivalent*:
(1) *I is a H-founded model of $\mathcal{P}$,*
(2) $I = \Phi_{\mathcal{P}}(I) \oplus s^H_{\mathcal{P}}(I)$,
(3) $I \models \mathcal{P} \oplus s^H_{\mathcal{P}}(I)$,
(4) $I = \Phi_{\mathcal{P}}(I \oplus s^H_{\mathcal{P}}(I))$.

Note that, by definition, the Kripke–Kleene semantics of a program coincides with its $\preccurlyeq_k$-least model founded on the assumption that assigns the value `f` to the atoms that are in the head of no rule. So, given a logic program $\mathcal{P}$, with H_{KK} let us denote the hypothesis that assigns to the atoms that are in the head of no rule the value `f`, while assigns to the others $\bot$. Therefore:

Corollary 13. *Let $\mathcal{P}$, I and H be a logic program, an interpretation and a hypothesis, respectively. If $H = H_{KK}$ then I is the $\preccurlyeq_k$-least H-founded model of $\mathcal{P}$ iff I is the Kripke–Kleene model of $\mathcal{P}$.*

The above corollary follows from the fact that for $H = H_{KK}$, $s^H_{\mathcal{P}}(I) = H$ and, thus, by Theorem 12, $I \models \mathcal{P} \oplus H$ iff I is the Kripke–Kleene model of $\mathcal{P}$.

From a fixed-point characterization point of view, from Theorem 12 it follows immediately that the set of H-founded models can be identified by the fixed-points of at least two

[8] I is viewed as a set.

$\preccurlyeq_k$-monotone immediate consequence operators:

$$\Pi^H_{\mathcal{P}}(I) = \Phi_{\mathcal{P}}(I \oplus s^H_{\mathcal{P}}(I)), \tag{4}$$

$$\tilde{\Pi}^H_{\mathcal{P}}(I) = \Phi_{\mathcal{P}}(I) \oplus s^H_{\mathcal{P}}(I). \tag{5}$$

This also guarantees the existence and uniqueness of the $\preccurlyeq_k$-least H-founded model of a program $\mathcal{P}$. Furthermore, these operators have a quite interesting property, once we restrict our attention to the everywhere false hypothesis $H = \mathtt{I_f}$. Under this condition, it can be shown that the least fixed-point under $\preccurlyeq_k$ of $\Pi^H_{\mathcal{P}}$ and, thus, of $\tilde{\Pi}^H_{\mathcal{P}}$ coincides with the well-founded semantics. Therefore,

Theorem 14. *Consider a logic program $\mathcal{P}$. Then the $\preccurlyeq_k$-least H-founded model of $\mathcal{P}$ w.r.t. the hypothesis $H = \mathtt{I_f}$ is the well-founded semantics of $\mathcal{P}$.*

Note that the above theorem is not surprising in the light of the fact that the $\tilde{\Pi}^H_{\mathcal{P}}$ operator is quite similar to the $W_{\mathcal{P}}$ operator defined in Eq. (3) for classical logic programs and interpretations. The above theorem essentially extends the relationship to general logic programs interpreted over bilattices.

Summing up, as the support is $\preccurlyeq_k$-monotone in H, from $\mathtt{I}_\perp \preccurlyeq_k H_{KK} \preccurlyeq_k \mathtt{I_f}$ we have the expected result:

Corollary 15. *Given a logic program $\mathcal{P}$, the $\preccurlyeq_k$-least H-founded model, I_{OWA}, of $\mathcal{P}$ under $H = \mathtt{I}_\perp$, the $\preccurlyeq_k$-least H-founded model, I_{KK}, of $\mathcal{P}$ under $H = H_{KK}$ (i.e. the Kripke–Kleene model of $\mathcal{P}$), the $\preccurlyeq_k$-least H-founded model, I_{WF}, of $\mathcal{P}$ under $H = \mathtt{I_f}$ (i.e. the Well-Founded model of $\mathcal{P}$), it follows that $I_{OWA} \preccurlyeq_k I_{KK} \preccurlyeq_k I_{WF}$.*

While for classical logical programs and total interpretations, $\tilde{\Pi}^H_{\mathcal{P}}(I)$ (under $H = \mathtt{I_f}$) characterizes stable total models (as, $\tilde{\Pi}^H_{\mathcal{P}} = W_{\mathcal{P}}$), this is not true in the general case of interpretations over bilattices. In fact, from Table 1, we see that the partial classical interpretation I_2 is a H-founded model not being stable.

3.3. H-closed models

The problem mentioned above comes from the fact that some H-founded models contain knowledge that is neither entailed by the program nor by the safe part of the hypothesis. For instance, under $H = \mathtt{I_f}$, I_2 is a H-founded model that asserts that the truth value of p is $\mathtt{t}$, knowledge that can neither be entailed by the program nor by H. Consequently, we shall concentrate on those H-founded models that do not contain any extra knowledge other than the knowledge provided by the hypothesis. In doing this, we group H-founded models into sets of models having a given support and then take the least informative one from each group. Formally, for a given interpretation I, we will consider the class of all models of $\mathcal{P} \oplus s^H_{\mathcal{P}}(I)$, i.e. interpretations which contain the knowledge entailed by $\mathcal{P}$ and the support $s^H_{\mathcal{P}}(I)$, and then take the $\preccurlyeq_k$-least model of this class. If this $\preccurlyeq_k$-least model is I itself then I is a H-closed model.

Definition 16 (*H-closed model*). Let $\mathcal{P}$, I and H be a logic program, an interpretation and a hypothesis, respectively. Then I is a *H-closed model* of P iff $I = \min_{\preccurlyeq_k}\{J \vDash \mathcal{P} \oplus s^H_{\mathcal{P}}(I)\}$, i.e. the unique $\preccurlyeq_k$-least model of the program $\mathcal{P} \oplus s^H_{\mathcal{P}}(I)$ coincides with I.[9]

Therefore, if I is a H-closed model then $I \vDash \mathcal{P} \oplus s^H_{\mathcal{P}}(I)$, i.e. $I = \Phi_{\mathcal{P}\oplus s^H_{\mathcal{P}}(I)}(I) = \Phi_{\mathcal{P}}(I) \oplus s^H_{\mathcal{P}}(I)$. Therefore, by Theorem 12, any H-closed model is a H-founded model as well, i.e. $I \vDash \mathcal{P}$ and $s^H_{\mathcal{P}}(I) \preccurlyeq_k I$.

Interestingly, H-closed models have also a different, equivalent and quite suggestive characterization. In fact, it follows immediately from Definition 16 that $\min_{\preccurlyeq_k}\{J\colon J \vDash \mathcal{P} \oplus s^H_{\mathcal{P}}(I)\} = KK(\mathcal{P} \oplus s^H_{\mathcal{P}}(I))$. Therefore,

Theorem 17. *Let $\mathcal{P}$, I and H be a logic program, an interpretation and a hypothesis, respectively. Then I is a H-closed model of $\mathcal{P}$ iff $I = KK(\mathcal{P} \oplus s^H_{\mathcal{P}}(I))$.*

That is, given an interpretation I, a logic program $\mathcal{P}$ and a hypothesis H, among all models of $\mathcal{P}$, we are looking for the $\preccurlyeq_k$-least models deductively closed under support k-completion (models that can be entailed by the program using their own support without any other extra knowledge). For instance, the interpretation $\bar{I}$ in Example 11 is a (the $\preccurlyeq_k$-least) H-closed model as well.

Finally, we may note that by Theorem 17, the fixed-points of the immediate consequence operator $KK(\mathcal{P} \oplus s^H_{\mathcal{P}}(\cdot))$ are exactly the H-closed models and, thus, the sequence of interpretations

$$\begin{aligned} I_0 &= \mathtt{I}_\perp, \\ I_{i+1} &= KK(\mathcal{P} \oplus s^H_{\mathcal{P}}(I_i)), \end{aligned}$$

converges to the $\preccurlyeq_k$-least H-closed model, which, thus, always exists and is unique.

We can device also an alternative immediate consequence operator. In the following we present the operator $\tilde{\Phi}^H_{\mathcal{P}}$, which coincides with $KK(\mathcal{P} \oplus s^H_{\mathcal{P}}(\cdot))$, i.e. $\tilde{\Phi}^H_{\mathcal{P}}(I) = KK(\mathcal{P} \oplus s^H_{\mathcal{P}}(I))$ for any interpretation I, but does not require any, even intuitive, program transformation like $\mathcal{P} \oplus s^H_{\mathcal{P}}(\cdot)$. This may be important in the classical logic programming case where $\mathcal{P} \oplus s^H_{\mathcal{P}}(\cdot)$ is not easy to define (as $\oplus$ does not belong to the language of classical logic programs). We show that the set of H-closed models coincides with the set of fixed-points of $\tilde{\Phi}^H_{\mathcal{P}}$.

Definition 18 (*immediate consequence operator* $\tilde{\Phi}^H_{\mathcal{P}}$). Consider a logic program $\mathcal{P}$, an interpretation I and a hypothesis H. Then $\tilde{\Phi}^H_{\mathcal{P}}(I)$ is defined as the following limit of the

[9] Uniqueness is guaranteed by the fact that the set of models J of $\mathcal{P} \oplus s^H_{\mathcal{P}}(I)$ coincides with the set of fixed-points of the $\preccurlyeq_k$-monotone operator $\Phi_{\mathcal{P}\oplus s^H_{\mathcal{P}}(I)}(J) = \Phi_{\mathcal{P}}(J) \oplus s^H_{\mathcal{P}}(I)$.

sequence of interpretations $J_i^{I,H}$: for any $i \geqslant 0$,

$$J_0^{I,H} = s_{\mathcal{P}}^H(I),$$

$$J_{i+1}^{I,H} = \Phi_{\mathcal{P}}(J_i^{I,H}) \oplus J_i^{I,H}.$$

It is easy to note that the sequence $J_i^{I,H}$ above is monotone increasing under $\preccurlyeq_k$ and, thus has a limit.

The following theorem follows directly from $\preccurlyeq_k$-monotonicity of $\Phi_{\mathcal{P}}$ and of the support, and from the Knaster–Tarski theorem.

Theorem 19. *The operator $\tilde{\Phi}_{\mathcal{P}}^H(I)$ is monotone in its arguments I and H w.r.t. $\preccurlyeq_k$.*

The following theorem characterizes the set of H-closed models in terms of fixed-points of $\tilde{\Phi}_{\mathcal{P}}^H$.

Theorem 20. *Let $\mathcal{P}$, I and H be a logic program, an interpretation and a hypothesis, respectively. Then $\tilde{\Phi}_{\mathcal{P}}^H(I) = KK(\mathcal{P} \oplus s_{\mathcal{P}}^H(I))$.*

It then follows immediately that

Corollary 21. *An interpretation I is a H-closed model of $\mathcal{P}$ iff I is a fixed-point of $\tilde{\Phi}_{\mathcal{P}}^H$.*

The following concluding theorem establishes that we have extended stable model semantics from the CWA to the AWA. Indeed, stable models of a logic program $\mathcal{P}$ coincide with fixed-points of $\tilde{\Phi}_{\mathcal{P}}^H$, under the closed world assumption (everywhere false hypothesis) $H = \mathtt{I_f}$.

Theorem 22. *Let $\mathcal{P}$, I and $H = \mathtt{I_f}$ be a logic program, an interpretation and the everywhere false hypothesis* (*CWA*), *respectively. The following statements are equivalent*:
(1) *I is a stable model of $\mathcal{P}$,*
(2) *I is a H-closed model of $\mathcal{P}$,*
(3) $I = \tilde{\Phi}_{\mathcal{P}}^H(I)$,
(4) $I = KK(\mathcal{P} \oplus s_{\mathcal{P}}^H(I))$.

Note that the above theorem gives new, both epistemic and fixed-point, characterizations of the stable models and emphasizes the role of the CWA. In particular, note the conceptual shift from the *truth-based* characterization of stable models to the *knowledge-based* one, i.e. from

$$I = \mathrm{lfp}_{\preccurlyeq_t}(\lambda x.\Psi_{\mathcal{P}}(x, I))$$

to

$$I = \min_{\preccurlyeq_k}\{J \vDash \mathcal{P} \oplus s_{\mathcal{P}}^H(I)\}.$$

Furthermore, unlike the Gelfond–Lifschitz transformation, it shows that no program transformation nor separation of the roles of positive and negative information are necessary in that semantics. In particular, the characterization of stable models through $\tilde{\Phi}^H_{\mathcal{P}}$, where $H = \mathtt{I_f}$, can be rewritten in the classical setting as follows. Consider a classical logic program $\mathcal{P}$, then a partial interpretation I is a stable model of $\mathcal{P}$, if and only if it is deductively closed under its greatest unfounded set completion, i.e. if and only if it coincides with the limit of the sequence:

$$J_0^{I,H} = \neg.U_{\mathcal{P}}(I),$$

$$J_{i+1}^{I,H} = \Phi_{\mathcal{P}}(J_i^{I,H}) \cup J_i^{I,H}.$$

Both Corollary 21 and Theorem 22 are illustrated in Tables 2 and 1, respectively.

4. Some applications of the AWA

Our approach allows deductive reasoning with default knowledge. That knowledge can be defined by or gathered in different ways. Of course, the programmer could define the AWA that should be associated with its program, but other scenarios could be developed as well. For instance, an assumption could be given by some knowledge, which would not be completely reliable, such as knowledge provided by some external source. In this scenario an assumption could be provided by the semantics of e.g. another logic program or by rules, which do hold *normally*, but admit exceptions. We illustrate some applications of our approach in the following examples.

4.1. Mixing OWA and CWA

Consider the following case, adapted from [32,36]. A judge is collecting information from two sources, the public prosecutor and the counsel for the defense, in order to decide whether to charge a person named Ted, accused of murder. The truth space is $\mathcal{FOUR}$, as the collection of information from different sources may easily lead to contradictory situations.

The judge first collects a set of facts

$$F = \{\ \mathtt{has\_witness(Ted)} \leftarrow \mathtt{f}$$
$$\mathtt{friends(John,\ Ted)} \leftarrow \mathtt{t}\}.$$

Then he combines them with his own set of rules R (described below) in order to make a decision.

$$\begin{aligned}
&\mathtt{is_suspect(x)} &&\leftarrow \mathtt{has_motive(x)} \vee \mathtt{has_witness(x)}\\
&\mathtt{is_cleared(x)} &&\leftarrow \exists y(\mathtt{has_alibi(x,y)} \wedge \neg\mathtt{friend(x,y)})\\
&\mathtt{is_cleared(x)} &&\leftarrow \mathtt{is_innocent(x)} \wedge \neg\mathtt{is_suspect(x)}\\
&\mathtt{friend(x,y)} &&\leftarrow \mathtt{friend(y,x)}\\
&\mathtt{friend(x,y)} &&\leftarrow \exists z(\mathtt{friend(x,z)} \wedge \mathtt{friend(z,y)})\\
&\mathtt{charge(x)} &&\leftarrow \mathtt{is_suspect(x)} \oplus \neg\mathtt{is_cleared(x)}.
\end{aligned}$$

The program is $\mathcal{P} = F \cup R$. The last rule is the judge's "decision making rule" and describes how the judge works: he joins the evidence of being suspect together with that of being not cleared to decide whether to charge Ted. The question is: what should the value of `charge(Ted)` be? Using F and R, then uniform default assumptions are not appropriate. Indeed, if the judge relies on the CWA, then he will decide that `Ted` is not cleared and must be charged despite the absence of proof. Relying on the OWA, the atoms `is_suspect(Ted)`, `is_cleared(Ted)` and `charge(Ted)` will be unknown, and the judge cannot take a decision (approaches based on OWA are often too weak). Assuming that by default the atoms `has_motive(Ted)`, `has_witness(Ted)` and `is_suspect(Ted)` are false, that the atom `is_innocent(Ted)` is true and that the others are unknown seems to be an appropriate assumption here. Relying on this assumption, the judge will infer that `Ted` is cleared, not suspect and should not be charged.

Related to the above example are those taken from the context of *Extended Logic Programs* (ELPs) (see, e.g. [2,3,21]). In ELP one can distinguish between *explicit negation*, $\neg A$, and *default negation*, $not(A)$. Informally, in the former case, to infer $\neg A$ we have to provide an explicit proof of $\neg A$, while in the latter case, we infer $not(A)$ if no proof of A exists. ELP offers a way of expressing explicitly a closed world assumption on an atom A by means of a rule of the form $\neg A \leftarrow not(A)$. That is A is false if no proof of A exists. In our approach, however, we do not rely, from a syntax point of view, on two different constructs $\neg A$ and $not(A)$, but have $\neg A$ only. The main difference to ELP is that in our approach we do not give a special meaning to $\neg A$, but rather manage it naturally: the evaluation of a negative literal $\neg A$ is given by the negation of the evaluation of A. The fact that A has a default value, $H(A)$, can in some special cases give to $\neg A$ a particular behavior. For instance, we may state a closed world assumption on an atom A by means of $H(A) = \mathtt{f}$ and, thus, $\neg A$ may behave like $not(A)$, while state an open world assumption on an atom A, simulating explicit negation, by means of $H(A) = \bot$. Also, by using the method described in [21] (and used in [2] as well) to remove expressions of the form $\neg A$ in ELPs, we may "translated" an ELP into our framework. Informally, given an ELP $\mathcal{P}$, we first replace all expressions $\neg A$ with a new atom $\bar{A}$ (in [21], $\bar{A}$ is called the *positive form* of $\neg A$) and, second, we then replace all expressions $not(A)$ with $\neg A$ and postulate $H(A) = \mathtt{f}$ for these atoms. All the other atoms have default value $\bot$ [10]. For instance, consider the ELP (see [21]) $\mathcal{P}_1 = \{(\neg P \leftarrow \mathtt{t}), (P \leftarrow \neg Q)\}$. Then according to the semantics of [21], $\mathcal{P}_1$ has unique answer set $\{\neg P\}$. In our setting, assuming $H = \mathtt{I}_\bot$, $\mathcal{P}_1$ is evaluated over $\mathcal{FOUR}$ and transformed into $\{(\bar{P} \leftarrow \mathtt{t}), (P \leftarrow \bar{Q})\}$. Therefore, the $\preccurlyeq_k$-least H-founded model I is such that $I(\bar{P}) = \mathtt{t}$ and $I(Q) = \bot$, which is one-to-one related to the answer set of $\mathcal{P}_1$.

However, in this paper we are not going to investigate further the relationship between ELPs (with their various semantics [2,3,21]) and our framework, but rather want to show how to use the feature of the any world assumption in some of the application

[10] We may also additionally relate $\bar{A}$ to A with some axioms, as in [2]. For instance we may add $\{\bar{A} \leftarrow \neg A\}$ (this is called the *strong negation axiom* in [2]), or add $\{A \leftarrow \neg\bar{A}\}$ (this is called the *weak negation axiom* or add $\{(A \leftarrow \neg\bar{A}), (\bar{A} \leftarrow \neg A)\}$ (this is called the *classical negation axiom*).

scenarios considered for ELPs:

- Consider, for instance, a rule expressing the fact that "if someone is innocent (s)he cannot be guilty". This may be represented by

  $$\texttt{Guilty(x)} \leftarrow \neg\texttt{Innocent(x)}.$$

  We assume (which is the rule in most countries) that anyone is by default innocent and not guilty i.e. $H(\texttt{Innocent(a)}) = \texttt{t}$, $H(\texttt{Guilty(a)}) = \texttt{f}$. Therefore, in order to charge someone, we have to provide a "proof" for being guilty (i.e. not innocent) from the facts.
- Similarly, a rule expressing the fact that "a car may cross railway tracks if there is no crossing train" may be represented by

  $$\texttt{Cross\_railway\_tracks} \leftarrow \neg\texttt{Train\_is\_comming}.$$

  In this situation, in order to safely cross the railway there should be explicit evidence that the train is not coming and, thus, we may assume by default that $H(\texttt{Train\_is\_comming}) = \bot$ (i.e. the atom is interpreted according to OWA) together with the default $H(\texttt{Cross\_railway\_tracks}) = \texttt{f}$, for safety.
- The set of terminal vertices (see [21]) of a directed graph can be defined by the following logic program:

  $$\texttt{Terminal(x)} \leftarrow \neg\texttt{Arc(x,y)}$$

  and hypothesis $H(\texttt{Arc(a,b)}) = \texttt{f}$, for all a,b of the Herbrand universe (the default for all instances of Terminal is not relevant).
- Suppose we have the logic program (see [21])

  $$\texttt{Employed(jack,stanford)} \leftarrow \texttt{t}$$
  $$\texttt{Employed(jane,sri)} \leftarrow \texttt{t}$$
  $$\texttt{AdequateIncome(x)} \leftarrow \texttt{Employed(x,y)}$$

  Then under the (OWA) hypothesis $H = \mathtt{I}_{\bot}$, in the H-founded model we have that both AdequateIncome(jack) and AdequateIncome(jane) are evaluated to t. However, under the above hypothesis both Employed(jack,sri) and Employed(jane, stanford) are evaluated to $\bot$. If we further assume that the database is complete, then we can formulate this hypothesis by using the CWA, i.e. $H = \mathtt{I}_{\mathtt{f}}$. In this latter case the H-founded model evaluates both Employed(jack,sri) and Employed(jane,stanford) to f, as expected. If, however, the available employment information is complete for Stanford only then we have to consider the hypothesis $H(\texttt{Employed(a,stanford)}) = \texttt{f}$ and $H(\texttt{Employed(a,sri)}) = \bot$, for all individuals a of the Herbrand universe. Then the H-founded model evaluates Employed(jack,sri) to $\bot$, while evaluates Employed(jane,stanford) to f, as expected.

4.2. Information integration with uncertainty

Consider the following case, adapted from [35]. Consider an insurance company, which has information about its customers used to determine the risk coefficient of each customer. The company has a set $\mathcal{P}$ of rules to compute the risk coefficient. In our case, the value of the

risk coefficient has to be re-evaluated as the client is a new client and his risk coefficient is given by his precedent insurance company. The bilattice we consider is the interval bilattice over the real unit interval $[0, 1]$, i.e. the elements of the bilattice represents sub-intervals of $[0, 1]$. The client declares to be 28-years old, to be not that experienced, but fairly good driver and driving more or less a sport car. The risk coefficient of his precedent insurance company is in between 0.6 and 0.8. The client's declarations are considered as the hypothesis H:

$$\begin{aligned} H(\texttt{Young(john)}) &= \langle 0.7, 1\rangle \\ H(\texttt{Experience(john)}) &= \langle 0.1, 0.5\rangle \\ H(\texttt{Good\_driver(john)}) &= \langle 0.6, 1\rangle \\ H(\texttt{Sport\_car(john)}) &= \langle 0.6, 0.8\rangle \\ H(\texttt{Risk(John)}) &= \langle 0.6, 0.8\rangle. \end{aligned}$$

The rules of the company are the following:

$$\begin{aligned} \texttt{Good\_driver(X)} &\leftarrow \texttt{Experience(x)} \wedge \neg\texttt{Risk(x)} \\ \texttt{Risk(x)} &\leftarrow \texttt{Young(x)} \\ \texttt{Risk(x)} &\leftarrow \texttt{Sport\_car(x)} \\ \texttt{Risk(x)} &\leftarrow \texttt{Experience(x)} \wedge \neg\texttt{Good\_driver(x)}. \end{aligned}$$

The rules would be formulated more realistically, in case we couple our language with a framework capable of combining uncertainty values in a more flexible way, like for instance [27,28], in which computable functions are allowed to combine certainty values. In our case, these functions are restricted to be $\wedge$, $\vee$, $\otimes$, $\oplus$ and $\neg$. See, for instance, [36] as a first step in this direction.

Now, it can be verified that the k-least H-closed model is I, where

$$I(A) = \begin{cases} \langle 0.7, 1\rangle & \text{if } A = \texttt{Young(john)} \\ \langle 0.1, 0.5\rangle & \text{if } A = \texttt{Experience(john)} \\ \langle 0.3, 1\rangle & \text{if } A = \texttt{Good\_driver(john)} \\ \langle 0.6, 0.8\rangle & \text{if } A = \texttt{Sport\_car(john)} \\ \langle 0.7, 1\rangle & \text{if } A = \texttt{Risk(john)}. \end{cases}$$

Note that both the risk coefficient, `Risk(john)`, as well as the driver capabilities, `Good_driver(john)`, of the client have been revised.

4.3. Default rules

Another frequent source of defaults is the case where we also want to express *default statements* of the form

Normally, unless something abnormal holds, then φ implies A.

Such statements were the main motivation for non-monotonic logics like *Default Logic* [47], Autoepistemic Logic [14,39,42,44] and *Circumscription* [40,41] (see also [22]). We can formulate such a statement in a natural way, using *abnormality theories*, as

$$\begin{aligned} &A \leftarrow \varphi \wedge \neg Ab, \\ &Ab \leftarrow \neg A\ , \end{aligned} \tag{6}$$

where *Ab* stands for *abnormality*, and then consider the hypothesis $H(Ab) = \mathtt{f}$, i.e. by default there are no abnormal objects. McCarty [40,41] originated the concept of abnormality theory as a way to represent default reasoning in Circumscription. Defaults are represented with an introduced *abnormality* predicate: for instance, to say that normally birds fly, we would use the rule

$$\begin{aligned} &\mathtt{Flies(x)} \leftarrow \mathtt{Bird(x)} \wedge \neg \mathtt{Ab}_1\mathtt{(x)} \\ &\mathtt{Ab}_1\mathtt{(x)} \leftarrow \neg \mathtt{Flies(x)}. \end{aligned} \tag{7}$$

Here the meaning of $\mathtt{Ab}_1(x)$ is something like "x is abnormal with respect to flying birds". Note that there can be many different kinds of abnormality, and they are indexed according to kind. Likewise in Circumscription, where abnormality predicates are minimized (allowing relevant predicates to vary—e.g. `Flies`), we have to assert that no object is abnormal by default, i.e. $H(\mathtt{Ab}_1(t)) = \mathtt{f}$, for all terms t over the Herbrand universe.

Abnormality theories can represent two important types of *defeating* arguments. One usually distinguishes between defeaters that contradict outright a default (Type I defeaters), and those where only the justification for a default is undermined, without contradicting its conclusion (Type II defeaters). Here is some illustrating example. If it is known that `tweety` is a bird, which does not fly (Type I defeat) then we add to rule (7) the two facts

$$\begin{aligned} &\mathtt{Bird(tweety)} \leftarrow \mathtt{t} \\ &\mathtt{Flies(tweety)} \leftarrow \mathtt{f}. \end{aligned}$$

Here the conclusion $\mathtt{Ab}_1\mathtt{(tweety)}$ follows in all H-closed models. On the other hand, we infer that `Flies(a)` for any bird `a` such that $\mathtt{a} \neq \mathtt{tweety}$.

For Type II defeat, we simply assert that `tweety` is abnormal, without asserting that he does not fly, i.e. we add to rule (7) the fact

$$\mathtt{Ab}_1\mathtt{(tweety)} \leftarrow \mathtt{t}.$$

Then the conclusion $\neg\mathtt{Flies(tweety)}$ follows in all H-closed models.

Note that, due to the generality of the notion of hypotheses, we may also deal with "(default) degrees of abnormality", rather than just consider the classical $\{\mathtt{f}, \mathtt{t}\}$ setting. So, for instance, we may take advantage of some statistical indication on the number of abnormal birds w.r.t. the property of flying: we may state that $H(\mathtt{Ab}_1(t)) = \langle 0, 0.2\rangle$ in the interval bilattice, indicating that the degree of being a bird t a non-flyer is quite low. So, for instance if we add to rule (7) the two facts

$$\begin{aligned} &\mathtt{Bird(polly)} \leftarrow \langle 1, 1\rangle \\ &\mathtt{Ab}_1\mathtt{(tweety)} \leftarrow \langle 0.9, 1\rangle, \end{aligned}$$

dictating that `polly` is a bird and that `tweety` is likely abnormal w.r.t. flying, then[11] the conclusions `Flies(polly)` and `Flies(tweety)` follow in all H-closed models with degree $\langle 0.8, 1\rangle$ and $\langle 0, 0.1\rangle$, respectively. Therefore, `polly` "likely" flies, whereas `tweety` does "unlikely" fly.

[11] $H(\mathtt{Flies}(t)) = H(\mathtt{Bird}(t)) = \langle 0, 1\rangle$.

Interestingly, default rules like (7) or variants may automatically be generated within the context of *Inductive Logic Programming* (ILP) (see, e.g. [29,45,48]) and *Data Mining* (DM) (see, e.g. [25]). In particular, if we are interested in the predicate `Flies`, informally one of the tasks in ILP and in DM, more precisely in the task of mining association rules, is to induce/mine a rule "explaining (to some degree)" the observations about the predicate `Flies`. A candidate rule may then have the form of (7) and is usually accompanied by some measures indicating an "appropriateness" degree of the rule. For instance, in mining association rules [25] (ILP has different measures), these values are usually the *support*, roughly indicating the probability of being $\texttt{Flies(x)} \wedge \texttt{Bird(x)}$ true, and the *confidence*, indicating the conditional probability $P(\texttt{Flies(x)}|\texttt{Bird(x)})$. Anyway, the important point here is that both ILP as well as DM can be considered as two similar ways to generate default rules and assumptions about abnormality predicates associated to those rules which may then be coupled together to some sure knowledge in order to make some new inferences, as we have seen above.

5. Conclusions and related work

In this paper we introduced the notion of Any-World Assumption (AWA). It generalizes the well-known notions of OWA and CWA. The former dictates that the default truth of the atoms is 'unknown', while the latter establishes that the default truth value is 'false'. These are two, though important, extreme assumptions of a large variety of possible assumptions. The AWA is a generalization of the above concepts as any assignment over any truth value set can be used as an assumption, i.e. as default knowledge. Our formalization assumes a monotone operator over the space of interpretations over bilattices, whose fixed-points are assumed to be the intended models of the world being represented by a set of formulae under the OWA. The particular instantiation we have chosen is that of logic programs, and the monotone operator is $\Phi_{\mathcal{P}}$, whose fixed-points are the so-called Kripke–Kleene models of a program. We have then introduced the notion of support, which regards the AWA as an additional source of information to be used to complete the knowledge provided by a logic program. The way we couple the support to the implicit knowledge provided by a logic program generalizes the well-known and long studied notion of stable model semantics for logic programs, which is based on the CWA. In fact, if we restrict our attention to the everywhere false assumption, then the usual stable models semantics is obtained and the support is one-to-one related with the greatest unfounded set.

Our work generalizes related work such as [2,26,31,32,36]. In [31] it has been shown that the usual semantics of logic programs can be obtained through a unique computation method, but using different uniform assumptions, i.e. assumptions that assign the same default truth-value to all the atoms. In [2,26] some atoms are allowed to be interpreted according to the OWA, while others are allowed to be interpreted according to the CWA and, thus, roughly the choice of the default is restricted to the value *unknown* and/or *false* (additionally, [26] allows to manage default rules as well). Closer to our approach are [32,36]. But, in [36], the extension is confined to particular logic programs for the management of certainty values and limited to the well-founded semantics [50] only and, thus, is just a particular case of our framework. Finally some concepts presented here were first sketched

in [32] but relying on some particular mode of evaluation of formulas that weakens the approach. As a consequence, the semantics proposed in [32] captured the well-founded semantics for Datalog programs with negation in three-valued logics, but fails for general logic programs over arbitrary truth spaces. Moreover no relationship to stable models was established.

There are still several issues open. A main objective is to extend the framework to the case where any set of formulae, not just logic programs are considered. Towards this direction it would be interesting, as first step, to extend the AWA to disjunctive logic programs, where rules are of the form $L_1 \vee \ldots \vee L_n \leftarrow \varphi$, and then generalize it to arbitrary formulae of the form $\varphi_1 \leftarrow \varphi_2$.

Acknowledgements

We would like to thank the anonymous reviewers for their valuable comments on the early version of this paper.

Appendix. Some proofs

Theorem 4. *Let $\mathcal{P}$ and I be a classical logic program and a partial classical interpretation, respectively. Consider the CWA $H = \mathtt{I}_{\mathtt{f}}$. Let X be a subset of $B_{\mathcal{P}}$. Then X is an unfounded set*[12] *of $\mathcal{P}$ w.r.t. I iff $\neg.X \preccurlyeq_k \Phi_{\mathcal{P}}(I \oplus \neg.X)$, i.e., $\neg.X$ is safe w.r.t. $\mathcal{P}$, I and H.*

Proof. Suppose X is unfounded w.r.t. $\mathcal{P}$ and I. Assume $A \in X$. Therefore, by definition of unfounded sets, if $A \leftarrow \varphi \in \mathcal{P}^*$, where $\varphi = \varphi_1 \vee \ldots \vee \varphi_n$ and $\varphi_i = L_{i_1} \wedge \ldots \wedge L_{i_n}$, then for all $i \in \{0, \ldots, n\}$, $I(\varphi_i) = \mathtt{f}$ or $\neg.X(\varphi_i) = \mathtt{f}$. Therefore, by monotonicity of $\wedge$ and $\vee$ w.r.t. $\preccurlyeq_k, \mathtt{f} \preccurlyeq_k (I \oplus \neg.X)(\varphi)$, i.e. $\neg.X(A) \preccurlyeq_k \Phi_{\mathcal{P}}(I \oplus \neg.X)(A)$. That relation holds for all $A \in X$ but also for the other atoms (in that case value is $\bot$ in X), thus $\neg.X \preccurlyeq_k \Phi_{\mathcal{P}}(I \oplus \neg.X)$. The other direction can be shown similarly using the fact that I is a partial classical interpretation, i.e. a interpretation over $\{\mathtt{f}, \bot, \mathtt{t}\}$. □

Theorem 8. *Let $\mathcal{P}$, I and H be a logic program, an interpretation and a hypothesis, respectively. Consider the iterated sequence of interpretations $F_i^{I,H}$ defined as follows: for any $i \geqslant 0$,*

$$F_0^{I,H} = H,$$

$$F_{i+1}^{I,H} = \sigma_{\mathcal{P}}^{I,H}(F_i^{I,H}).$$

The sequence $F_i^{I,H}$ is monotone decreasing under $\preccurlyeq_k$ and, thus, reaches a fixed-point $F_\lambda^{I,H}$, for a limit ordinal λ. Furthermore, $s_{\mathcal{P}}^H(I) = F_\lambda^{I,H}$ holds.

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

Proof. As $F_1^{I,H} \preccurlyeq_k F_0^{I,H}$ and $\sigma_{\mathcal{P}}^{I,H}$ is monotone under $\preccurlyeq_k$, so the sequence is decreasing under $\preccurlyeq_k$, i.e. $F_{i+1}^{I,H} \preccurlyeq_k F_i^{I,H} \preccurlyeq_k H$. Therefore, the sequence has a fixed-point at the limit, say $F_\lambda^{I,H}$.

Let us show that $F_\lambda^{I,H}$ is safe and $\preccurlyeq_k$-greatest. $F_\lambda^{I,H} = \sigma_{\mathcal{P}}^{I,H}(F_\lambda^{I,H}) = H \otimes \Phi_{\mathcal{P}}(I \oplus F_\lambda^{I,H})$. Therefore, $F_\lambda^{I,H} \preccurlyeq_k H$ and $F_\lambda^{I,H} \preccurlyeq_k \Phi_{\mathcal{P}}(I \oplus F_\lambda^{I,H})$, so $F_\lambda^{I,H}$ is safe w.r.t. $\mathcal{P}$ and H.

Consider any X safe w.r.t. $\mathcal{P}$ and H. We show by induction on i that $X \preccurlyeq_k F_i^{I,H}$ and, thus, at the limit $X \preccurlyeq_k F_\lambda^{I,H}$, so $F_\lambda^{I,H}$ is $\preccurlyeq_k$-greatest.

(i) Case $i = 0$. By definition, $X \preccurlyeq_k H = F_0^{I,H}$.

(ii) Induction step: suppose $X \preccurlyeq_k F_i^{I,H}$. Since X is safe, we have $X \preccurlyeq_k X \otimes X \preccurlyeq_k H \otimes \Phi_{\mathcal{P}}(I \oplus X)$. By induction, $X \preccurlyeq_k H \otimes \Phi_{\mathcal{P}}(I \oplus F_i^{I,H}) = F_{i+1}^{I,H}$. □

In the following, with $F_i^{I,H}$ we will always indicate the ith iteration of the computation of the support of $\mathcal{P}$ w.r.t. I, H, according to Theorem 8.

Theorem 9. *Let $\mathcal{P}$ be a logic program. The support operator $s_{\mathcal{P}}^H(I)$ is monotone in its arguments I and H w.r.t. $\preccurlyeq_k$.*

Proof. Consider two interpretations I and J, where $I \preccurlyeq_k J$, and two hypotheses H and H', where $H \preccurlyeq_k H'$. Consider the two sequences $F_i^{I,H}$ and $F_i^{J,H'}$. We show by induction on i that $F_i^{I,H} \preccurlyeq_k F_i^{J,H'}$ and, thus, at the limit $s_{\mathcal{P}}^H(I) \preccurlyeq_k s_{\mathcal{P}}^{H'}(J)$.

(i) Case $i = 0$. By definition, $F_0^{I,H} = H \preccurlyeq_k H' = F_0^{J,H}$.

(ii) Induction step: suppose $F_i^{I,H} \preccurlyeq_k F_i^{J,H'}$. By monotonicity under $\preccurlyeq_k$ of $\Phi_{\mathcal{P}}$ and the induction hypothesis, $F_{i+1}^{I,H} = H \otimes \Phi_{\mathcal{P}}(I \oplus F_i^{I,H}) \preccurlyeq_k H' \otimes \Phi_{\mathcal{P}}(J \oplus F_i^{J,H'}) = F_{i+1}^{J,H'}$, which concludes. □

Theorem 12. *Let $\mathcal{P}$, I and H be a logic program, an interpretation and a hypothesis, respectively. The following statements are equivalent*:

(1) *I is a H-founded model of $\mathcal{P}$,*

(2) $I = \Phi_{\mathcal{P}}(I) \oplus s_{\mathcal{P}}^H(I)$,

(3) $I \vDash \mathcal{P} \oplus s_{\mathcal{P}}^H(I)$,

(4) $I = \Phi_{\mathcal{P}}(I \oplus s_{\mathcal{P}}^H(I))$.

Proof. Assume Point 1. holds, i.e. $I \vDash \mathcal{P}$ and $s_{\mathcal{P}}^H(I) \preccurlyeq_k I$. Then, $I = \Phi_{\mathcal{P}}(I) = \Phi_{\mathcal{P}}(I) \oplus s_{\mathcal{P}}^H(I)$, so Point 2 holds. Assume Point 2 holds. Then, by Eq. (2), $I = \Phi_{\mathcal{P}}(I) \oplus s_{\mathcal{P}}^H(I) = \Phi_{\mathcal{P} \oplus s_{\mathcal{P}}^H(I)}(I)$, i.e. $I \vDash \mathcal{P} \oplus s_{\mathcal{P}}^H(I)$, so Point 3 holds. Assume Point 3 holds. So, $s_{\mathcal{P}}^H(I) \preccurlyeq_k I$ and from the safeness of $s_{\mathcal{P}}^H(I)$, it follows that $s_{\mathcal{P}}^H(I) \preccurlyeq_k \Phi_{\mathcal{P}}(I \oplus s_{\mathcal{P}}^H(I)) = \Phi_{\mathcal{P}}(I)$ and, thus, $I = \Phi_{\mathcal{P} \oplus s_{\mathcal{P}}^H(I)}(I) = \Phi_{\mathcal{P}}(I) \oplus s_{\mathcal{P}}^H(I) = \Phi_{\mathcal{P}}(I)$. Therefore, $\Phi_{\mathcal{P}}(I \oplus s_{\mathcal{P}}^H(I)) = \Phi_{\mathcal{P}}(I) = I$, so Point 4 holds. Finally, assume Point 4 holds. From the safeness of $s_{\mathcal{P}}^H(I)$, it follows that $s_{\mathcal{P}}^H(I) \preccurlyeq_k \Phi_{\mathcal{P}}(I \oplus s_{\mathcal{P}}^H(I)) = I$. Therefore, $I = \Phi_{\mathcal{P}}(I \oplus s_{\mathcal{P}}^H(I)) = \Phi_{\mathcal{P}}(I)$ (i.e. $I \vDash \mathcal{P}$) and, thus I is a H-founded model of $\mathcal{P}$. So, Point 1 holds, which concludes the proof. □

Concerning Theorem 14, the operator $\Pi^H_{\mathcal{P}}$ in Eq. (4), with $H = \mathtt{I_f}$, has been defined first in [38], without recognizing it to characterize H-founded models. But, it has been shown in [38] that the least fixed-point under $\preccurlyeq_k$ coincides with the well-founded semantics. Therefore, from [38] and Theorem 12 it follows immediately that:

Theorem 14. *Consider a logic program $\mathcal{P}$. Then the $\preccurlyeq_k$-least H-founded model of $\mathcal{P}$ w.r.t. the hypothesis $H = \mathtt{I_f}$ is the well-founded semantics of $\mathcal{P}$.*

Note that
- by definition $\tilde{\Phi}^H_{\mathcal{P}}(I) = \Phi_{\mathcal{P}}(\tilde{\Phi}^H_{\mathcal{P}}(I)) \oplus \tilde{\Phi}^H_{\mathcal{P}}(I)$, and thus $\Phi_{\mathcal{P}}(\tilde{\Phi}^H_{\mathcal{P}}(I)) \preccurlyeq_k \tilde{\Phi}^H_{\mathcal{P}}(I)$; and
- for fixed-points of $\tilde{\Phi}^H_{\mathcal{P}}$ we have that $I = \Phi_{\mathcal{P}}(I) \oplus I$ and, thus, $\Phi_{\mathcal{P}}(I) \preccurlyeq_k I$.

Before proving Theorem 20, we need the following lemma.

Lemma 1. *Let $\mathcal{P}$, H, I and K be a logic program, a hypothesis and two interpretations, respectively. If $K \vDash \mathcal{P} \oplus s^H_{\mathcal{P}}(I)$ then $\tilde{\Phi}^H_{\mathcal{P}}(I) \preccurlyeq_k K$.*

Proof. Assume $K \vDash \mathcal{P} \oplus s^H_{\mathcal{P}}(I)$, i.e. by Eq. (2), $K = \Phi_{\mathcal{P} \oplus s^H_{\mathcal{P}}(I)}(K) = \Phi_{\mathcal{P}}(K) \oplus s^H_{\mathcal{P}}(I)$. Therefore, $s^H_{\mathcal{P}}(I) \preccurlyeq_k K$. We show by induction on i that $J^{I,H}_i \preccurlyeq_k K$ and, thus, at the limit $\tilde{\Phi}^H_{\mathcal{P}}(I) \preccurlyeq_k K$.

(i) Case $i = 0$. By definition, $J^{I,H}_0 = s^H_{\mathcal{P}}(I) \preccurlyeq_k K$.

(ii) Induction step: suppose $J^{I,H}_i \preccurlyeq_k K$. Then by assumption and by induction we have that $J^{I,H}_{i+1} = \Phi_{\mathcal{P}}(J^{I,H}_i) \oplus J^{I,H}_i \preccurlyeq_k \Phi_{\mathcal{P}}(K) \oplus K = \Phi_{\mathcal{P}}(K) \oplus \Phi_{\mathcal{P}}(K) \oplus s^H_{\mathcal{P}}(I) = \Phi_{\mathcal{P}}(K) \oplus s^H_{\mathcal{P}}(I) = K$, which concludes. □

Theorem 20. *Let $\mathcal{P}$, I and H be a logic program, an interpretation and a hypothesis, respectively. Then $\tilde{\Phi}^H_{\mathcal{P}}(I) = KK(\mathcal{P} \oplus s^H_{\mathcal{P}}(I))$.*

Proof. Let K be the Kripke–Kleene model of $\mathcal{P} \oplus s^H_{\mathcal{P}}(I)$ under $\preccurlyeq_k$. K is the limit of the sequence

$$K_0 = \mathtt{I}_{\perp},$$

$$K_{i+1} = \Phi_{\mathcal{P} \oplus s^H_{\mathcal{P}}(I)}(K_i).$$

As $K \vDash \mathcal{P} \oplus s^H_{\mathcal{P}}(I)$, by Lemma 1, $\tilde{\Phi}^H_{\mathcal{P}}(I) \preccurlyeq_k K$. Now we show that $K \preccurlyeq_k \tilde{\Phi}^H_{\mathcal{P}}(I)$, by proving by induction on i that $K_i \preccurlyeq_k \tilde{\Phi}^H_{\mathcal{P}}(I)$ and, thus, at the limit $K \preccurlyeq_k \tilde{\Phi}^H_{\mathcal{P}}(I)$.

(i) Case $i = 0$. We have $K_0 = \mathtt{I}_{\perp} \preccurlyeq_k \tilde{\Phi}^H_{\mathcal{P}}(I)$.

(ii) Induction step: suppose $K_i \preccurlyeq_k \tilde{\Phi}^H_{\mathcal{P}}(I)$. Then, by induction we have $K_{i+1} = \Phi_{\mathcal{P} \oplus s^H_{\mathcal{P}}(I)}(K_i) \preccurlyeq_k \Phi_{\mathcal{P} \oplus s^H_{\mathcal{P}}(I)}(\tilde{\Phi}^H_{\mathcal{P}}(I))$. As $s^H_{\mathcal{P}}(I) \preccurlyeq_k \tilde{\Phi}^H_{\mathcal{P}}(I)$, by Eq. (2) it follows that $K_{i+1} \preccurlyeq_k$

$\Phi_{\mathcal{P}\oplus s^H_{\mathcal{P}}(I)}(\tilde{\Phi}^H_{\mathcal{P}}(I)) = \Phi_{\mathcal{P}}(\tilde{\Phi}^H_{\mathcal{P}}(I)) \oplus s^H_{\mathcal{P}}(I) \preccurlyeq_k \Phi_{\mathcal{P}}(\tilde{\Phi}^H_{\mathcal{P}}(I)) \oplus \tilde{\Phi}^H_{\mathcal{P}}(I) = \tilde{\Phi}^H_{\mathcal{P}}(I)$, which concludes. □

The concluding Theorem 22, follows immediately from the work in [37], where the hypothesis is "hardwired" to the everywhere false hypothesis $H = \mathtt{I}_{\mathtt{f}}$.

Theorem 22 (*Loyer and Straccia [37]*). *Let $\mathcal{P}$, I and $H = \mathtt{I}_{\mathtt{f}}$ be a logic program, an interpretation and the everywhere false hypothesis* (*CWA*), *respectively. The following statements are equivalent*:
(1) *I is a stable model of $\mathcal{P}$,*
(2) *I is a H-closed model of $\mathcal{P}$,*
(3) $I = \tilde{\Phi}^H_{\mathcal{P}}(I)$,
(4) $I = KK(\mathcal{P} \oplus s^H_{\mathcal{P}}(I))$.

References

[1] J. Alcantâra, C.V. Damásio, L.M. Pereira, Paraconsistent logic programs, in: Proc. of the 8th European Conf. on Logics in Artificial Intelligence (JELIA-02), Lecture Notes in Computer Science, Vol. 2424, Cosenza, Italy, 2002, Springer, pp. 345–356.
[2] J.J. Alferes, L.M. Pereira, On logic program semantics with two kinds of negation, in: Proc. of the Joint Internat. Conf. and Symp. on Logic Programming, Washington, USA, 1992, The MIT Press, Cambridge, MA, pp. 574–588.
[3] O. Arieli, Paraconsistent declarative semantics for extended logic programs, Ann. Math. Artificial Intelligence 36 (4) (2002) 381–417.
[4] O. Arieli, A. Avron, Reasoning with logical bilattices, J. Logic, Language and Information 5 (1) (1996) 25–63.
[5] O. Arieli, A. Avron, The value of the four values, Artificial Intelligence J. 102 (1) (1998) 97–141.
[6] N.D. Belnap, A useful four-valued logic, in: G. Epstein, J.M. Dunn (Eds.), Modern Uses of Multiple-valued Logic, Reidel, Dordrecht, NL, 1977, pp. 5–37.
[7] H. Blair, V.S. Subrahmanian, Paraconsistent logic programming, Theoret. Comput. Sci. 68 (1989) 135–154.
[8] K.L. Clark, Negation as failure, in: H. Gallaire, J. Minker (Eds.), Logic and Data Bases, Plenum Press, New York, NY, 1978, pp. 293–322.
[9] C.V. Damásio, L.M. Pereira, A survey of paraconsistent semantics for logic programs, in: D. Gabbay, P. Smets (Eds.), Handbook of Defeasible Reasoning and Uncertainty Management Systems, Kluwer, 1998, pp. 241–320.
[10] C.V. Damásio, L.M. Pereira, Antitonic logic programs, in: Proc. of the 6th European Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR-01), Lecture Notes in Computer Science, Vol. 2173, Springer, 2001.
[11] M. Denecker, V.W. Marek, M. Truszczyński, Approximating operators, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning, in: J. Minker, (Ed.), NFS-workshop on Logic-based Artificial Intelligence, 1999, pp. 1–26.
[12] M. Denecker, V.W. Marek, M. Truszczyński, Ultimate approximations in nonmonotonic knowledge representation systems, in: D. Fensel, F. Giunchiglia, D. McGuinness, M. Williams (Eds.), Principles of Knowledge Representation and Reasoning: Proc. of the 8th Internat. Conf. Morgan Kaufmann, 2002, pp. 177–188.
[13] M. Denecker, V.W. Marek, M. Truszczyński, Uniform semantic treatment of default and autoepistemic logics, Artificial Intelligence J. 143 (2003) 79–122.
[14] J. Doyle, D. McDermott, Nonmonotonic logic I, Artificial Intelligence 13 (1980) 41–72.
[15] M.C. Fitting, The family of stable models, J. Logic Programming 17 (1993) 197–225.

[16] M.C. Fitting, Fixpoint semantics for logic programming—a survey, Theoret. Comput. Sci. 21 (3) (2002) 25 –51.
[17] M. Fitting, A Kripke–Kleene-semantics for general logic programs, J. Logic Programming 2 (1985) 295–312.
[18] M. Fitting, Bilattices and the semantics of logic programming, J. Logic Programming 11 (1991) 91–116.
[19] M. Fitting, Kleene's logic, generalized, J. Logic and Computation 1 (6) (1992) 797–810.
[20] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in: R.A. Kowalski, K. Bowen (Eds.), Proc. of the 5th Internat. Conf. on Logic Programming, Cambridge, Massachusetts, The MIT Press, Cambridge, 1988, pp. 1070–1080.
[21] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Generation Comput. 9 (3/4) (1991) 365–386.
[22] M.L. Ginsberg, (Eds.), Readings in Nonmonotonic Reasoning, Morgan Kaufmann, Los Altos, CA, 1987.
[23] M.L. Ginsberg, Multi-valued logics: a uniform approach to reasoning in artificial intelligence, Computational Intelligence 4 (1988) 265–316.
[24] R. Hähnle, G. Escalada-Imaz, Deduction in many-valued logics: a survey, Mathware and Soft Comput. IV (2) (1997) 69–97.
[25] J. Han, M. Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, San Francisco, USA, 2000.
[26] K. Inoue, Hypothetical reasoning in logic programs, J. Logic Programming 18 (1994) 191–227.
[27] M. Kifer, V.S. Subrahmanian, Theory of generalized annotated logic programming and its applications, J. Logic Programming 12 (1992) 335–367.
[28] L.V.S. Lakshmanan, N. Shiri, A parametric approach to deductive databases with uncertainty, IEEE Trans. Knowledge and Data Engng 13 (4) (2001) 554–570.
[29] N. Lavrac, S. Dzeroski, Inductive Logic Programming: Techniques and Applications, Ellis Horwood, New York, 1994.
[30] N. Leone, P. Rullo, F. Scarcello, Disjunctive stable models: unfounded sets, fixpoint semantics, and computation, Information and Comput. 135 (2) (1997) 69–112.
[31] N. Loyer, Y. Spyratos, D. Stamate, Parametrized semantics of logic programs—a unifying framework, Theoret. Comput. Sci. 308 (1–3) (2003) 429–447.
[32] Y. Loyer, N. Spyratos, D. Stamate, Hypotheses-based semantics of logic programs in multi-valued logics, ACM Trans. Computational Logic 15 (3) (2004) 508–527.
[33] Y. Loyer, 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), Lecture Notes in Computer Science, Vol. 2424, Cosenza, Italy, Springer, 2002, pp. 271–282.
[34] Y. Loyer, U. Straccia, The well-founded semantics in normal logic programs with uncertainty, in: Proc. of the 6th Internat. Symp. on Functional and Logic Programming (FLOPS-2002), Lecture Notes in Computer Science, Vol. 2441, Aizu, Japan, 2002, Springer, pp. 152–166.
[35] Y. Loyer, U. Straccia, The approximate well-founded semantics for logic programs with uncertainty, in: 28th Internat. Symp. on Mathematical Foundations of Computer Science (MFCS-2003), Lecture Notes in Computer Science, Vol. 2747, Bratislava, Slovak Republic, 2003, Springer, pp. 541–550.
[36] Y. Loyer, U. Straccia, Default knowledge in logic programs with uncertainty, in: Proc. of the 19th Internat. Conf. on Logic Programming (ICLP-03), Lecture Notes in Computer Science, Vol. 2916, Mumbai, India, 2003, Springer, pp. 466–480.
[37] Y. Loyer, U. Straccia, An epistemic foundation of stable model semantics, Technical Report ISTI-2003-TR-11, Istituto di Scienza e Tecnologie dell'Informazione, Consiglio Nazionale delle Ricerche, Pisa, Italy, 2003.
[38] Y. Loyer, U. Straccia, Epistemic foundation of the well-founded semantics over bilattices, in: 29th Internat. Symp. on Mathematical Foundations of Computer Science (MFCS-2004), Lecture Notes in Computer Science, Vol. 3153, Bratislava, Slovak Republic, 2004, Springer, pp. 513–524.
[39] V.W. Marek, M. Truszczyński, Autoepistemic logic, J. ACM 38 (3) (1991) 587–618.
[40] J. McCarthy, Circumscription—a form of nonmonotonic reasoning, Artificial Intelligence 13 (1980) 27–39.
[41] J. McCarthy, Applications of circumscription to formalizing commonsense knowledge, Artificial Intelligence 28 (1986) 89–116.
[42] D. McDermott, Nonmonotonic logic II, J. ACM 22 (1982) 33–57.
[43] R.C. Moore, Possible-world semantics for autoepistemic logic, in: Proc. of the 1st Internat. Workshop on Nonmonotonic Reasoning, New Paltz, NY, 1984, pp. 344–354.

[44] R.C. Moore, Semantical considerations on nonmonotonic logic, Artificial Intelligence 25 (1985) 75–94.
[45] S. Muggleton, L. De Raedt, Inductive logic programming: theory and methods, J. Logic Programming 19/20 (1994) 629–679.
[46] R. Reiter, On closed world data bases, in: H. Gallaire, J. Minker (Eds.), Logic and Data Bases, Plenum Press, New York, NY, 1978, pp. 55–76.
[47] R. Reiter, A logic for default reasoning, Artificial Intelligence 13 (1980) 81–132.
[48] C. Sakama, Induction from answer sets in nonmonotonic logic programs, ACM Trans. Computational Logic (TOCL), 2004.
[49] A. Tarski, A lattice-theoretical fixpoint theorem and its applications, Pacific J. Math. (5) (1955) 285–309.
[50] A. van Gelder, K.A. Ross, J.S. Schlimpf, The well-founded semantics for general logic programs, J. ACM 38 (3) (1991) 620–650.