Artificial Intelligence 174 (2010) 1285–1306

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Reducts of propositional theories, satisfiability relations, and generalizations of semantics of logic programs ☆

Miroslaw Truszczyński

Department of Computer Science, University of Kentucky, Lexington, KY 40506, USA

ARTICLE INFO

Article history:
Received 24 November 2009
Received in revised form 6 August 2010
Accepted 8 August 2010
Available online 12 August 2010

Keywords:
Logic programming
Stable models
Supported models
Reducts
Logic HT

ABSTRACT

Over the years, the stable-model semantics has gained a position of the correct (two-valued) interpretation of default negation in programs. However, for programs with aggregates (constraints), the stable-model semantics, in its broadly accepted generalization stemming from the work by Pearce, Ferraris and Lifschitz, has a competitor: the semantics proposed by Faber, Leone and Pfeifer, which seems to be *essentially* different. Our goal is to explain the relationship between the two semantics. Pearce, Ferraris and Lifschitz's extension of the stable-model semantics is best viewed in the setting of arbitrary propositional theories. We propose here an extension of the Faber–Leone–Pfeifer semantics, or *FLP semantics*, for short, to the full propositional language, which reveals both common threads and differences between the FLP and stable-model semantics. We use our characterizations of FLP-stable models to derive corresponding results on strong equivalence and on normal forms of theories under the FLP semantics. We apply a similar approach to define supported models for arbitrary propositional theories, and to study their properties.

1. Introduction

The stable-model semantics introduced by Gelfond and Lifschitz [23] is the foundation of answer-set programming [37, 41,22], a paradigm for modeling and solving search problems. Answer-set programming is broadly accepted as an effective knowledge representation tool for modeling intelligent agents and reasoning in complex domains [12,13,3]. In the last decade, it has been successfully applied in several areas of artificial intelligence such as product configuration [47], planning [50,48], reasoning about action [25], and diagnosis [43,2], with some of these applications concerning large-scale systems like the space shuttle flight controller [43]. Answer-set programming has also been applied beyond artificial intelligence for problems arising in bio-informatics [4,46], linguistics [7] and automated music generation [5].

The success of answer-set programming as a knowledge representation formalism and its applications in artificial intelligence and beyond make it essential that theoretical underpinnings of its semantics be established. Consequently, right from its inception, the stable-model semantics, has received much attention. The present paper contributes to this general line of research by extending the theoretical framework for the stable-model semantics based of the results and ideas proposed and developed by Pearce [44] and Ferraris [19] to two other closely related semantics that also play a major role in answer-set programming, the Faber–Leone–Pfeifer stable-model semantics [16] and the supported-model semantics [9,1,38].

A far-reaching contribution by Pearce [44] explained the stable-model semantics in terms of models of theories in the logic of *here-and-there* (*HT*, for short), introduced by Heyting [26]. It had two important consequences. First, it resulted in a generalization of the stable-model semantics, originally limited to a restricted syntax of program rules, to *arbitrary* theories

☆ An extended abstract of this paper appeared in the proceedings of the 2009 International Conference on Logic Programming.

E-mail address: mirek@cs.uky.edu.

0004-3702/$ – see front matter
doi:10.1016/j.artint.2010.08.004

in the language of propositional logic (we discuss the role of this development in more detail later). Second, it brought about the notion of *strong equivalence* of programs, fundamental to modular program development [32]. Strong equivalence has been extensively studied in the past decade. That research resulted in extensions and refinements of the original concept, in characterizations, and in complexity results [32,35,51,14,52,49].

The original definition of stable models [23] was based on the *reduct* of a program with respect to a set of atoms. The characterization in terms of the logic HT makes no reference to reducts but employs a form of model minimization. Ferraris [19] extended the notion of reduct to propositional theories, and developed the reduct-based definition of stable models equivalent to that provided by the logic HT (an exposition of the idea can also be found in the paper by Ferraris and Lifschitz [21]).

The papers by Pearce and Ferraris resulted in an elegant comprehensive treatment of the stable-model semantics. They also raise the question whether there are other generalizations of the stable-model semantics to the case of arbitrary logic theories. An indication that it might be so comes from the work by Faber et al. [16] on programs with aggregates. Aggregates, in the form of weight atoms, were introduced to answer-set programming by Niemelä and Simons [42], who extended the stable-model semantics to that class of programs. Ferraris [19] cast that generalization in terms of stable models of propositional theories. Stable models of programs with aggregates are no longer guaranteed to be minimal models. From the perspective of the Ferraris' result, it is not surprising. Stable models of propositional theories in general do not have the minimal-model property.

However, as minimization is an important knowledge-representation principle, Faber et al. [16] sought an alternative semantics for programs with constraints, one that would have the minimal-model property. Naturally, they also wanted it to coincide with the original semantics on the class of programs without aggregates. They came up with a solution that satisfied both requirements by modifying the concept of the reduct! In the setting with aggregates, the Faber–Leone–Pfeifer stable-model semantics, or *FLP semantics*, is different than the extension of the original stable-model semantics based on the logic HT (throughout the paper, whenever we speak about the stable-model semantics, we have the original semantics in mind). Thus, the question of alternative generalizations is relevant. The FLP semantics is steadily gaining on importance. It is now not only used as the basis for interpreting aggregates in the *dlv* system [17], but also in approaches aiming to integrate answer-set programming with other declarative programing paradigms [11].

A related question concerns a possibility of generalizing other semantics relevant to answer-set programming to the full propositional logic language. The one we consider here is the supported-model semantics. Its importance stems from two properties. First, the supported-model semantics is the key component of a characterization of stable models in terms of loop formulas [36], which gave rise to fast algorithms for computing stable models of programs [36,31,30]. Second, for a class of modal theories of some restricted syntax, it is a precise counterpart to the semantics of expansions of the autoepistemic logic [40,38], an important nonmonotonic logic for modeling belief sets of an agent with perfect introspection capabilities.

Given the applications of the Faber–Leone–Pfeifer stable-model semantics as an alternative to the standard Gelfond–Lifschitz one, and the role of the supported-model semantics in answer-set programming and nonmonotonic logics, our objective here is to investigate these semantics and show that they also can be studied by the means stemming from those developed by Pearce and Ferraris for the stable-model semantics. Specifically, we have the following goals:

(1) To extend the semantics of Faber et al. [16] to the language of propositional logic. We do so in two equivalent ways: by means of a generalization of the reduct introduced by Faber et al., as well as in terms of a certain satisfiability relation similar to the one that defines the logic HT. We show that the FLP semantics generalizes several properties of the stable-model semantics of logic programs and so, it can be regarded as its legitimate extension, alongside with the extension based on the logic HT. We derive several additional properties of the FLP semantics, including a characterization of strong equivalence under that semantics, and a normal-form result.
(2) To relate the FLP and stable-model semantics of propositional theories. We show that each can be expressed in each other in the sense that there are modular translations that do not use any auxiliary atoms and such that FLP-stable models of a theory are stable models of its image under the translation (and *vice versa*).
(3) To apply a similar two-pronged approach, exploiting both some notion of reduct and some satisfiability relation, to the supported model semantics. We show that also supported models can be defined for arbitrary propositional theories. We generalize to propositional language some well-known properties of supported models, as well as the results connecting stable and supported models of programs.

While most implemented answer-set programming systems [10] used in applications support only theories consisting of rules (we formally define rules in the next section), a generalization of answer-set programming to the full language of propositional logic is important. From the theoretical standpoint, it eliminates possible artifacts of syntactic restrictions and allows us to identify key principles behind the semantics of answer-set programming. In particular, considering answer set programming in the full language pinpoints the basic role of implication as a nonclassical connective, with the nonclassical behavior of the negation being a consequence of the fact that the negation can be expressed by means of the implication with the false consequent. The key role of implication is emphasized by the recursive definitions of the reducts — it is the only case that is treated in a nonstandard way. Moreover, it leads to the semantics of HT-interpretations, which paves the way to generalizations of answer-set programming and its semantics to the case of first-order logic theories [28,45].

From the practical standpoint, generalizations of the syntax of answer-set programming to the full language makes answer-set programming more flexible as a modeling formalism, and provides a basis for further extensions of the language, for instance, with aggregates [20].

Our paper demonstrates that the ideas originated by Pearce and Ferraris extend to two other semantics of logic programs: the FLP semantics and the supported-model semantics. The results concerning the FLP semantics have several potential implications and applications. They provide a certain normal-form result (cf. Section 4), which points to a possible extension of the syntax of disjunctive logic program rules currently supported by disjunctive logic programming systems such as *dlv*. The results on strong equivalence (cf. Section 3.5) lay the necessary foundation for the development of techniques and methods for modular program design under the FLP semantics. Finally, the extension of the FLP semantics to arbitrary rules demonstrates the feasibility of extending the present implementation of the *dlv* system to a richer input language not restricted to rules. Our results on the supported-model semantics are also of interest. As we observed above, logic programming with the supported-model semantics captures in a direct way a fragment of autoepistemic logic [38]. By extending the supported-model semantics to the entire language of propositional logic, we provide a way to expand the scope of this direct connection.

Our paper is organized as follows. In the next section, we recall two definitions of stable models of propositional theories. The first one is in terms of a reduct introduced by Ferraris [19]. It extends the original approach of Gelfond and Lifschitz. The second definition is in terms of HT-interpretations and is due to Pearce [44]. In Section 3, we discuss the approach by Faber et al. [16], extend it to arbitrary propositional theories, and study the properties of the resulting concepts. In particular, we characterize the general FLP-stable-model semantics in terms of the appropriately modified concept of the reduct, and in terms of a certain entailment relation based on HT-interpretations. We also discuss the question of the minimality of FLP-stable models, the complexity of reasoning with FLP-stable models, and the concept of strong equivalence with respect to FLP-stable models. Finally, we present a normal form theorem for that semantics. In Section 5, we show that techniques used in our paper can be applied to the supported-model semantics. Specifically, we define supported models for arbitrary propositional theories by modifying the notion of a reduct, and by introducing yet another entailment relation based on HT-interpretations. We derive several results for the generalized supported-model semantics and, in particular, we study the concept of strong equivalence for supported-model semantics, and the relationship of that semantics to those based on stable models and FLP-stable models.

2. Preliminaries

In this section we introduce basic terminology and describe the general definitions of the stable-model semantics in terms of the here-and-there models [44], and in terms of the *Ferraris* reduct, or *F*-reduct, for short [19], that generalizes the original Gelfond–Lifschitz reduct.

We consider the language of propositional logic determined by an infinite countable set *At* of atoms, and *boolean connectives* $\bot$, $\wedge$, $\vee$, and $\rightarrow$. A Backus–Naur Form expression $\varphi ::= \bot | A | (\varphi \wedge \varphi) | (\varphi \vee \varphi) | (\varphi \rightarrow \varphi)$, where $A \in At$, provides a concise definition of a formula. The parentheses are used only to disambiguate the order of binary operations. Whenever possible, we omit them. Generalizing the concept of the head of a program rule, we say that an occurrence of an atom is a *head* occurrence if it does not occur in the antecedent of any implication. Finally, when writing formulas, we often use the following shorthands:

$$\top = \bot \rightarrow \bot \quad \text{and} \quad \neg F = F \rightarrow \bot.$$

A set of formulas is a *theory*. In the case of all semantics we discuss here, there is no essential difference between *finite* theories and formulas. The former can be represented as the conjunctions of their elements. We often distinguish between formulas and theories as we want to address the case of infinite theories, too.

In the paper, we consider several special types of formulas and theories. A *rule* is a formula

$$A_1 \wedge \cdots \wedge A_m \wedge \neg B_1 \wedge \cdots \wedge \neg B_n \rightarrow C_1 \vee \cdots \vee C_s \vee \neg D_1 \vee \cdots \vee \neg D_t, \tag{1}$$

where A_i's, B_i's, C_i's and D_i's are atoms. If we use r to denote the rule (1), we say that the formulas $A_1 \wedge \cdots \wedge A_m \wedge \neg B_1 \wedge \cdots \wedge \neg B_n$ and $C_1 \vee \cdots \vee C_s \vee \neg D_1 \vee \cdots \vee \neg D_t$ are the *body* and the *head* of r, and denote them by $body(r)$ and $head(r)$, respectively. If $m = n = 0$, we represent the rule by its head. If $s = t = 0$, we write $\bot$ for the head of the rule. A *program* is a set of rules. We emphasize that the phrases "a head occurrence in a formula", discussed two paragraphs above, and "an element of the head of a rule" have a different meaning. In particular, each literal $\neg D_i$ is an element of the head of the rule (1), but the corresponding occurrence of D_i in the rule is *not* a head occurrence.

For consistency with the standard logic programming terminology, when referring to rules (1) with no negated atoms in the heads, we use the term *disjunctive program rule* or, simply, *disjunctive rule*. Further, we call disjunctive rules with at most one atom in the head *normal program rules* or, simply *normal rules*. By the convention above, disjunctive rules have no negated atoms in the head and so, this terminology agrees with the standard one.

Originally, the stable-model semantics was defined by Gelfond and Lifschitz [23] for *normal programs* (collections of normal rules). The definition was later extended to *disjunctive programs*, that is, collections of disjunctive rules also by Gelfond and Lifschitz [24], then to programs as understood here (collections of rules (1)) by Lifschitz and Woo [34], and to

a superclass of programs, *programs with nested expressions* by Lifschitz et al. [33]. Finally, the case of arbitrary theories was addressed by Pearce [44] and, later and in a different way, by Ferraris [19]. These last two approaches are equivalent. We will now discuss them, starting with the latter one.

For a formula F and a set of atoms Y, we define the *Ferraris reduct* (*F-reduct*) of F with respect to Y, written as F^Y, by induction:

R1. $\bot^Y = \bot$.
R2. If A is an atom:

$$A^Y = \begin{cases} A & \text{if } Y \models A, \\ \bot & \text{otherwise.} \end{cases}$$

R3. For $\circ = \wedge$ and $\vee$:

$$(G \circ H)^Y = \begin{cases} G^Y \circ H^Y & \text{if } Y \models G \circ H, \\ \bot & \text{otherwise.} \end{cases}$$

R4. For $\rightarrow$:

$$(G \rightarrow H)^Y = \begin{cases} G^Y \rightarrow H^Y & \text{if } Y \models G \rightarrow H, \\ \bot & \text{otherwise.} \end{cases}$$

We could have folded case (R4) into the case (R3). However, all concepts of reduct we consider later in the paper differ only in the way the implication is handled and so, we show this case separately.

For a theory $\mathcal{F}$, we define the F-reduct $\mathcal{F}^Y$ by setting $\mathcal{F}^Y = \{F^Y \mid F \in \mathcal{F}\}$. Next, we define $Y \subseteq At$ to be a *stable model* of $\mathcal{F}$ if Y is a minimal model of the theory $\mathcal{F}^Y$. One can show that stable models are models (hence, the term stable *model* is justified).

We will now illustrate the notions we just introduced. We will use the theories discussed below throughout the paper.

Example 1. Let $\mathcal{E}_1 = \{\neg\neg A \rightarrow A\}$. To compute the reduct $\mathcal{E}_1^{\emptyset}$, we note that $\emptyset \models \neg\neg A \rightarrow A$ (as the formula is a classical tautology). Thus, $\mathcal{E}_1^{\emptyset} = (\neg\neg A)^{\emptyset} \rightarrow A^{\emptyset}$. Since $\emptyset \not\models A$, $A^{\emptyset} = \bot$. Moreover, $\neg\neg A$ stands for $(A \rightarrow \bot) \rightarrow \bot$. Since $\emptyset \not\models (A \rightarrow \bot) \rightarrow \bot$, $(\neg\neg A)^{\emptyset} = ((A \rightarrow \bot) \rightarrow \bot)^{\emptyset} = \bot$. It follows that $\mathcal{E}_1^{\emptyset} = \bot \rightarrow \bot$. Clearly, $\emptyset \models \mathcal{E}_1^{\emptyset}$ and, trivially, there is no proper subset X of $\emptyset$ such that $X \models \mathcal{E}_1^{\emptyset}$. Thus, $\emptyset$ is a stable model of $\mathcal{E}_1$.

Similarly, as $\{A\} \models \neg\neg A \rightarrow A$, $\mathcal{E}_1^{\{A\}} = (\neg\neg A)^{\{A\}} \rightarrow A^{\{A\}}$. The definition implies that $A^{\{A\}} = A$. Moreover, as $\{A\} \models (A \rightarrow \bot) \rightarrow \bot$ (the expanded form of $\neg\neg A$), $((A \rightarrow \bot) \rightarrow \bot)^{\{A\}} = (A \rightarrow \bot)^{\{A\}} \rightarrow \bot^{\{A\}}$. Since $A \not\models A \rightarrow \bot$, $(A \rightarrow \bot)^{\{A\}} = \bot$. Also, by the definition, $\bot^{\{A\}} = \bot$. Thus, $\mathcal{E}_1^{\{A\}} = (\bot \rightarrow \bot) \rightarrow A$. Clearly, $\{A\} \models \mathcal{E}_1^{\{A\}}$. As there is no proper subset X of $\{A\}$ such that $X \models \mathcal{E}_1^{\{A\}}$, also $\{A\}$ is a stable model of $\mathcal{E}_1$.

Example 2. Let $\mathcal{E}_2 = \{(A \vee \neg A) \rightarrow A\}$. To compute the reduct $\mathcal{E}_2^{\emptyset}$, we note that $\emptyset \not\models (A \vee \neg A) \rightarrow A$. Thus, $\mathcal{E}_2^{\emptyset} = \bot$. It follows, in particular, that $\emptyset$ is not a stable model of $\mathcal{E}_2$.

Next, we observe that $A \models (A \vee \neg A) \rightarrow A$. Thus, $\mathcal{E}_2^{\{A\}} = (A \vee \neg A)^{\{A\}} \rightarrow A^{\{A\}}$. Since $(\neg A)^{\{A\}} = (A \rightarrow \bot)^{\{A\}} = \bot$, $\mathcal{E}_2^{\{A\}} = (A \vee \bot) \rightarrow A$. Clearly, we have $\emptyset \models \mathcal{E}_2^{\{A\}}$. Thus, $\{A\}$ is not a stable model of $\mathcal{E}_2$, either, and so, $\mathcal{E}_2$ has no stable models (as in the case of normal programs, we can restrict the search for stable models to subsets of the set of atoms that occur in the theory).

This notion of a stable model generalizes all earlier ones. It also coincides with the one proposed by Pearce [44]. The approach by Pearce is based on the logic HT [26], a logic located strictly between the intuitionistic and the propositional logics. Stable models are defined in terms of the satisfiability relation $\models_{ht}$ in the logic HT. A pair $\langle X, Y\rangle$, where $X, Y \subseteq At$, is an *HT-interpretation* if $X \subseteq Y$. The relation $\models_{ht}$, between HT-interpretations and formulas, is defined inductively as follows:

(1) $\langle X, Y\rangle \not\models_{ht} \bot$;
(2) $\langle X, Y\rangle \models_{ht} A$ if $X \models A$ (applies only if $A \in At$);
(3) $\langle X, Y\rangle \models_{ht} G \wedge H$ if $\langle X, Y\rangle \models_{ht} G$ and $\langle X, Y\rangle \models_{ht} H$;
(4) $\langle X, Y\rangle \models_{ht} G \vee H$ if $\langle X, Y\rangle \models_{ht} G$ or $\langle X, Y\rangle \models_{ht} H$;
(5) $\langle X, Y\rangle \models_{ht} G \rightarrow H$ if $Y \models G \rightarrow H$; and $\langle X, Y\rangle \not\models_{ht} G$, or $\langle X, Y\rangle \models_{ht} H$.

The relation extends in a standard way to theories. If for a theory $\mathcal{F}$, $\langle X, Y\rangle \models_{ht} \mathcal{F}$, then $\langle X, Y\rangle$ is an *HT-model* of $\mathcal{F}$. Some important properties of the relation $\models_{ht}$ are gathered below (cf. Ferraris and Lifschitz [21]).

Theorem 1. *For every formula F and every $X \subseteq Y \subseteq At$:*

(1) $\langle X, Y\rangle \models_{ht} F$ *implies* $Y \models F$;
(2) $\langle X, Y\rangle \models_{ht} \neg F$ *if and only if* $Y \models \neg F$;
(3) $\langle Y, Y\rangle \models_{ht} F$ *if and only if* $Y \models F$.

Pearce [44] defined Y to be a stable model of a theory $\mathcal{F}$ if and only if $\langle Y, Y\rangle \models_{ht} \mathcal{F}$ and for every $X \subseteq Y$ if $\langle X, Y\rangle \models_{ht} \mathcal{F}$, then $X = Y$ (a form of *minimality*). Ferraris and Lifschitz [21] proved that the two approaches are equivalent by showing the following two key results.

Theorem 2. *Let $\mathcal{F}$ be a theory*:

(1) *for every $Y \subseteq At$, $Y \models \mathcal{F}$ if and only if $Y \models \mathcal{F}^Y$*;
(2) *for every $X \subseteq Y \subseteq At$, $X \models \mathcal{F}^Y$ if and only if $\langle X, Y\rangle \models_{ht} \mathcal{F}$.*

Example 3. Let us consider the theory $\mathcal{E}_1 = \{\neg\neg A \to A\}$ from Example 1 and let $Y = \emptyset$. Since $Y \not\models (A \to \bot) \to \bot$, $\langle Y, Y\rangle \not\models_{ht} (A \to \bot) \to \bot$ and $Y \models ((A \to \bot) \to \bot) \to A$. Thus, $\langle Y, Y\rangle \models_{ht} ((A \to \bot) \to \bot) \to A$ or, in other words, $\langle Y, Y\rangle \models_{ht} \mathcal{E}_1$. Trivially, there is no proper subset X of Y such that $\langle X, Y\rangle \models_{ht} \mathcal{E}_1$. Thus, $Y = \emptyset$ is a stable model of $\mathcal{E}_1$, according to the definition by Pearce.

Next, let $Z = \{A\}$. Then $Z \models ((A \to \bot) \to \bot) \to A$ and $\langle Z, Z\rangle \models_{ht} A$. Thus, $\langle Z, Z\rangle \models_{ht} ((A \to \bot) \to \bot) \to A$ and, consequently, $\langle Z, Z\rangle \models_{ht} \mathcal{E}_1$. The only proper subset of Z is $X = \emptyset$. Clearly, $\langle X, Z\rangle \not\models_{ht} A$ (as $A \notin X$). Let us also observe that $Z \models (A \to \bot) \to \bot$, and $\langle X, Z\rangle \not\models_{ht} A \to \bot$ (as $Z \not\models A \to \bot$). Thus, $\langle X, Z\rangle \models_{ht} (A \to \bot) \to \bot$. It follows that $\langle X, Z\rangle \not\models_{ht} ((A \to \bot) \to \bot) \to A$. Thus, $\langle X, Z\rangle \not\models_{ht} \mathcal{E}_1$, and so $Z = \{A\}$ is a stable model of $\mathcal{E}_1$ according to the definition by Pearce.

Similarly, one can check that the theory $\mathcal{E}_2$ from Example 2 has no stable models according to the definition by Pearce.

Of course, these outcomes are only to be expected, given our discussion in Examples 1 and 2, and the equivalence of the definitions proposed by Ferraris and Pearce.

We conclude by noting that throughout the paper, we are only interested in the satisfiability of reducts (the one discussed in this section and two other types we introduce later) with respect to the standard propositional logic semantics. Thus, whenever we compute the reduct, we can simplify it by using propositional tautologies. Such simplifications have no effect on the concept of stability. For instance, we could simplify the reduct $\mathcal{E}_1^A = (\bot \to \bot) \to A$ to A.

3. FLP semantics

Faber et al. [16] based their work on a notion of reduct that differs from the one proposed by Gelfond and Lifschitz. Using our notation, it can be defined as follows. Let R be a disjunctive rule (that is, there are no negated atoms in the head)

$$A_1 \wedge \cdots \wedge A_m \wedge \neg B_1 \wedge \cdots \wedge \neg B_n \to C_1 \vee \cdots \vee C_s,$$

where A_i, B_i and C_i are atoms, and let Y be a set of atoms. The *FLP-reduct* $R^{\underline{Y}}$ (the notation we use is meant to distinguish between the FLP- and the F-reduct) is either R, if $Y \models A_1 \wedge \cdots \wedge A_m \wedge \neg B_1 \wedge \cdots \wedge \neg B_n$, or $\top$, otherwise. Given a disjunctive program $\mathcal{P}$, $\mathcal{P}^{\underline{Y}}$ is obtained by replacing each rule $R \in \mathcal{P}$ with $R^{\underline{Y}}$. Finally, Y is a *stable model* of $\mathcal{P}$ in the sense of Faber et al., if Y is a minimal model of $\mathcal{P}^{\underline{Y}}$. Faber et al. [16] proved that "their" stable models of disjunctive programs coincide with standard stable models. They also observed that the FLP-reduct does not depend on the syntactic form of the body of a rule. All that matters is whether the body is satisfied by Y. Thus, they extended the definition to more general formulas that are of the form

$$F \to C_1 \vee \cdots \vee C_s, \tag{2}$$

where C_i are atoms and F is a propositional formula.[1] That allowed them to extend the concept of a stable model to the class of theories that consist of such "generalized" disjunctive rules. Importantly, they proved that stable models, in their sense, of such theories are *minimal models*, while the stable-model semantics does not have that property. For instance, the program (theory) $\mathcal{E}_1 = \{\neg\neg A \to A\}$ has only one FLP-stable model, $\emptyset$, but two stable models, $\emptyset$ and $\{A\}$ (cf. Example 1 and Example 3, above).

3.1. General FLP semantics

To extend that approach to arbitrary propositional theories, we first generalize the notion of the FLP-reduct. To this end, we follow the inductive pattern of the definition of the F-reduct. There is no change for $F = \bot$, $F = A$, where $A \in At$, and

[1] Faber et al. used conjunctions of literals and aggregate atoms as F, but that detail is immaterial here.

$F = G \circ H$, where $\circ = \vee$ and $\wedge$. Indeed, there does not seem to be any other way, in which these cases could be handled. Thus, the only case that requires a discussion is that of $F = G \to H$. Once that case is settled, we will define Y to be an *FLP-stable model* of a theory $\mathcal{F}$ if Y is a minimal model of the FLP-reduct $\mathcal{F}^{\underline{Y}}$.

So, let us discuss the case of the implication. A literal reading of the FLP-reduct for rules suggests the following inductive definition for the case $F = G \to H$:

$$(G \to H)^{\underline{Y}} = G \to H, \quad \text{if } Y \models G; \qquad \text{otherwise}, \quad (G \to H)^{\underline{Y}} = \top.$$

However, under that choice, all occurrences of $\to$ (and so, also all occurrences of $\neg$) in the consequent of another implication would be interpreted in the classical way. While not a problem for formulas that do not have any implications occurring in the consequent of any "top-level" implication (and so, working correctly for the class of formulas considered by Faber et al.), in general it leads to some counterintuitive behavior.

For instance, let $\mathcal{F} = \{\neg\neg A\}$ and $\mathcal{G} = \{\neg B \to \neg\neg A\}$, where A and B are atoms. As B does not appear in the head of the rule of $\mathcal{G}$, it must be false in every reasonable generalization of the stable-model semantics. Consequently, $\mathcal{F}$ and $\mathcal{G}$ should have the same stable models. However, under the proposed definition it would not be so. Let $Y = \{A\}$. Since $\neg\neg A = (A \to \bot) \to \bot$ and $Y \not\models A \to \bot$, we would have $\mathcal{F}^{\underline{Y}} = \{\top\}$. Consequently, Y would not be a minimal model of $\mathcal{F}^{\underline{Y}} = \{\top\}$ (as $\emptyset$ is a model, too) and so, Y would not be a "stable" model of $\mathcal{F}$. On the other hand, as $Y \models \neg B$, $\mathcal{G}^{\underline{Y}} = \{\neg B \to \neg\neg A\}$. Thus, clearly, Y would be a minimal model of $\mathcal{G}^{\underline{Y}}$ and, consequently, a "stable" model of $\mathcal{G}$. A problem in itself, it also leads to another one. In $\mathcal{G}$, A has no head occurrence (informally, there is no "defining clause" for A in $\mathcal{G}$), yet $\mathcal{G}$ would have $\{A\}$ as a "stable" model.

Thus, we need to handle the case of $\to$ differently, but in such a way that under the restriction to theories consisting of formulas (2) we obtain the same concept of a stable model as the one proposed by Faber et al. In particular, we must ensure that all occurrences of $\to$ in the consequent of another occurrence of $\to$ are treated consistently in the same nonclassical way. In the remainder of this section we will argue that it can be accomplished by the following definition:

FLP4.

$$(G \to H)^{\underline{Y}} = \begin{cases} G \to H^{\underline{Y}} & \text{if } Y \models G \text{ and } Y \models H, \\ \top & \text{if } Y \not\models G, \\ \bot & \text{otherwise (that is, when } Y \not\models G \to H). \end{cases}$$

While it looks different than the original definition [16], it preserves its basic idea. Specifically, in the first case, when the implication is "strongly" satisfied (both its antecedent and consequent are satisfied by Y), we keep the antecedent unchanged, following the spirit of the original definition of Faber et al., but replace the consequent recursively with its reduct, to make sure the implications occurring in the antecedent are treated in a consistent way. The case when Y "weakly" satisfies the implication, that is, does not satisfy its antecedent, is dealt with as in the previous naive attempt (and as in the definition by Faber et al.). Namely, reflecting the principle that if the antecedent of an implication is false (informally, the implication "does not fire"), the implication is immaterial and can be replaced by $\top$ (effectively "removed"). In the case when the implication is not satisfied by Y, it can be replaced by $\bot$. Faber et al. do not distinguish this case and, in fact, proceed differently. They keep the rule in the program. However, they could have replaced it with $\bot$, as we propose (following the pattern for F-reduct), without affecting the resulting concept of a stable model. Indeed, if Y does not satisfy a rule in a program, Y cannot be a stable model of that program. Replacing a rule violated by Y with $\bot$ just makes that explicit.

To summarize, we define the *FLP-reduct* of the formula F with respect to Y, $F^{\underline{Y}}$, recursively, by using the clauses (R1)–(R3) of the definition of the F-reduct (adjusted to the notation $F^{\underline{Y}}$), as well as the clause (FLP4) for the implication $\to$. We extend the definition to theories in the standard way. With this definition in hand, we define next the notion of an FLP-stable model of a propositional theory (as announced above).

Definition 1. Let $\mathcal{F}$ be a theory. A set of atoms Y is an *FLP-stable model* of $\mathcal{F}$ if Y is a minimal model of $F^{\underline{Y}}$.

Example 4. We consider again the theory $\mathcal{E}_1 = \{\neg\neg A \to A\}$ from Example 1. To compute the reduct $\mathcal{E}_1^{\underline{\emptyset}}$, we note that $\emptyset \not\models \neg\neg A$. Thus, $\mathcal{E}_1^{\underline{\emptyset}} = \top$. Clearly, $\emptyset \models \mathcal{E}_1^{\underline{\emptyset}}$ and, trivially, there is no proper subset X of $\emptyset$ such that $X \models \mathcal{E}_1^{\underline{\emptyset}}$. Thus, $\emptyset$ is an FLP-stable model of $\mathcal{E}_1$.

On the other hand, we have $\{A\} \models \neg\neg A$ and $\{A\} \models A$. Thus, $\mathcal{E}_1^{\underline{\{A\}}} = \neg\neg A \to A^{\underline{\{A\}}}$. The definition implies that $A^{\underline{\{A\}}} = A$. Thus, $\mathcal{E}_1^{\underline{\{A\}}} = \neg\neg A \to A$. Clearly, $\{A\} \models \mathcal{E}_1^{\underline{\{A\}}}$. However, we also have that $\emptyset \models \mathcal{E}_1^{\underline{\{A\}}}$. Thus, $\{A\}$ is not an FLP-stable model of $\mathcal{E}_1$.

Example 5. Next, we reconsider the theory $\mathcal{E}_2 = \{(A \vee \neg A) \to A\}$ from Example 2. To compute the reduct $\mathcal{E}_2^{\underline{\emptyset}}$, we note that $\emptyset \not\models (A \vee \neg A) \to A$. Thus, $\mathcal{E}_2^{\underline{\emptyset}} = \bot$. It follows, as in Example 2, that $\emptyset$ is not an FLP-stable model of $\mathcal{E}_2$.

On the other hand, we observe that $\{A\} \models (A \vee \neg A)$ and $\{A\} \models A$. Thus, $\mathcal{E}_2^{\{A\}} = (A \vee \neg A) \rightarrow A^{\{A\}}$. Since $A^{\{A\}} = A$, it follows that $\mathcal{E}_2^{\{A\}} = (A \vee \neg A) \rightarrow A$. Clearly, we have $\{A\} \models \mathcal{E}_2^{\{A\}}$ and $\emptyset \not\models \mathcal{E}_2^{\{A\}}$. Thus, $\{A\}$ is an FLP-stable model of $\mathcal{E}_2$.

Examples 1, 2, 4 and 5 show that stable models need not be FLP-stable models and *vice versa*. Later, we provide a detailed comparison between the two semantics.

3.2. Basic properties

We start with a generalization of the well-known property of the standard F-reduct of disjunctive programs (cf. Theorem 2).

Proposition 1. *For every theory $\mathcal{F}$ and for every set of atoms Y, $Y \models \mathcal{F}$ if and only if $Y \models \mathcal{F}^{\underline{Y}}$.*

Proof. It is enough to prove that for every formula F, we have $Y \models F$ if and only if $Y \models F^{\underline{Y}}$. We proceed by induction. The base cases of $F = \bot$ and $F = A$, where $A \in At$, are evident. Let $F = G \wedge H$. If $Y \not\models F$, then $F^{\underline{Y}} = \bot$. Thus, both sides of the equivalence are false, and the equivalence follows. If $Y \models F$ or $Y \models F^{\underline{Y}}$, then $F^{\underline{Y}} = G^{\underline{Y}} \wedge H^{\underline{Y}}$. Since

(1) $Y \models F$ if and only if $Y \models G$ and $Y \models H$, and
(2) $Y \models F^{\underline{Y}}$ if and only if $Y \models G^{\underline{Y}}$ and $Y \models H^{\underline{Y}}$,

the equivalence of $Y \models F$ and $Y \models F^{\underline{Y}}$ follows by the induction hypothesis. The argument for $\vee$ is similar. Thus, let $F = G \rightarrow H$. If $Y \not\models F$, then $F^{\underline{Y}} = \bot$ and the equivalence in the assertion holds. Similarly, if $Y \not\models G$, then $F^{\underline{Y}} = \top$, and both $Y \models F$ and $Y \models F^{\underline{Y}}$ hold. Finally, let $Y \models G$ and $Y \models H$. In this case, $F^{\underline{Y}} = G \rightarrow H^{\underline{Y}}$. By the inductive hypothesis, $Y \models H^{\underline{Y}}$ and so, $Y \models G \rightarrow H^{\underline{Y}}$. Thus, also in that case, both $Y \models F$ and $Y \models F^{\underline{Y}}$ hold. □

It follows that FLP-stable models are indeed models of formulas and theories.

Corollary 1. *Let $\mathcal{F}$ be a theory and Y a set of atoms. If Y is an FLP-stable model of $\mathcal{F}$, then Y is a model of $\mathcal{F}$.*

This result allows us to prove that on theories consisting of formulas of the form (2) FLP-stable models defined here and stable models of Faber et al. [16] coincide. Thus, our approach is a generalization of the one by Faber et al.

Theorem 3. *Let $\mathcal{P}$ be a theory consisting of formulas of type* (2)*. Then Y is a stable model of $\mathcal{P}$ according to the definition by Faber et al.* [16] *if and only if Y is the FLP-stable model, according to Definition* 1.

Proof. Let $\mathcal{P}$ be a theory consisting of formulas (2), and let Y be a set of atoms. For a formula $R = F \rightarrow C_1 \vee \cdots \vee C_s$ from $\mathcal{P}$, we denote by R' and R'' the reducts of R with respect to Y according to Faber et al., and according to our definition, respectively. Further, we write $\mathcal{P}'$ and $\mathcal{P}''$ for the reducts of a program $\mathcal{P}$ with respect to Y according to Faber et al., and according to our definition, respectively.

Reasoning in either direction we can assume that Y is a model of $\mathcal{P}$ (it is known that stable models according to Faber et al. [16] are models; for FLP-stable models, it follows from Corollary 1). Thus, $\mathcal{P}'$ consists of those rules $R = F \rightarrow C_1 \vee \cdots \vee C_s$, for which $Y \models F$. In addition, it might possibly contain $\top$. The reduct $\mathcal{P}''$ differs only in that each formula $R = F \rightarrow C_1 \vee \cdots \vee C_s$ from $\mathcal{P}$ that is retained in $\mathcal{P}'$, contributes to $\mathcal{P}''$ its reduct $R'' = F \rightarrow C'_1 \vee \cdots \vee C'_t$, where $C'_1, \ldots, C'_t$ are precisely those elements in $\{C_1, \ldots, C_s\}$ that hold in Y. In addition, as $\mathcal{P}'$, $\mathcal{P}''$ may also contain $\top$. It is evident, that for every $Z \subseteq Y$, $Z \models \mathcal{P}'$ if and only if $Z \models \mathcal{P}''$. Thus, Y is a minimal model of $\mathcal{P}'$ if and only if Y is a minimal model of $\mathcal{P}''$, and so, the result follows. □

One of the problematic properties of the literal attempt to generalize the approach by Faber et al. was that stable models of some theories contained atoms without head occurrences. We will now show that our generalization behaves properly in this respect.

Proposition 2. *Let $\mathcal{F}$ be a theory and Y an FLP-stable model of $\mathcal{F}$. Then every atom in Y has a head occurrence in $\mathcal{F}$.*

Proof. First, we prove by induction that for every set S of atoms containing all atoms with head occurrences in a formula F, and for every $Z \subseteq At$, if $Z \models F$ then $Z \cap S \models F^{\underline{Z}}$. It is a stronger property than what we need below, but it is the one for which the inductive argument can be made to work. If $F = \bot$, the claim is trivially true. If $F = A$, then $A \in S$. Let Z be any subset of At such that $Z \models F$. Then, it follows that $A \in Z$ and $F^{\underline{Z}} = A$. Thus, $Z \cap S \models F^{\underline{Z}}$ holds, as claimed. If $F = G \wedge H$ or $G \vee H$, then atoms with head occurrences in G (H, respectively) are contained in S. Thus, the induction hypothesis applies to G and H (G or H, respectively), and the claim follows.

Finally, let $F = G \to H$. Since $Z \models F$, $F^{\underline{Z}} = \top$, or $Z \models G$, $Z \models H$ and $F^{\underline{Z}} = G \to H^{\underline{Z}}$. In the first case, the assertion is evident. In the latter case, we have $Z \models H$. By the induction hypothesis (it can be used as all atoms with head occurrences in H have head occurrences in F, and so they belong to S), $Z \cap S \models H^{\underline{Z}}$. Thus, $Z \cap S \models F^{\underline{Z}}$ in that case, too.

Next, we prove the result. Let S be the set of atoms with head occurrences in $\mathcal{F}$. Since Y is an FLP-stable model of $\mathcal{F}$, $Y \models \mathcal{F}^{\underline{Y}}$. By Proposition 1, $Y \models \mathcal{F}$. By the claim we proved above, $Y \cap S \models \mathcal{F}^{\underline{Y}}$. Since Y is a minimal model of $F^{\underline{Y}}$ and $Y \cap S \subseteq Y$, $Y \cap S = Y$ and, consequently, $Y \subseteq S$. □

Finally, we state and prove two properties that we use later in the paper.

Proposition 3. *For every formulas F and G, and for every set of atoms Y:*

(1) $F^{\underline{Y}} \equiv \bot$ *if and only if* $Y \not\models F$;
(2) $(F \circ G)^{\underline{Y}} \equiv F^{\underline{Y}} \circ G^{\underline{Y}}$, *where* $\circ = \wedge$ *or* $\vee$.

Proof. (1) We proceed by induction. The cases $F = \bot$ and $F = A$, where $A \in At$, are straightforward. Thus, let us assume that $F = G \wedge H$. First, we consider the case when $Y \not\models F$. In that case, $F^{\underline{Y}} = \bot$ and the "if" part of the assertion follows. Conversely, let $Y \models F$. Then, $Y \models G$ and $Y \models H$. By Proposition 1, $Y \models G^{\underline{Y}}$ and $Y \models H^{\underline{Y}}$. Thus, $Y \models G^{\underline{Y}} \wedge H^{\underline{Y}}$. Since in such case $F^{\underline{Y}} = G^{\underline{Y}} \wedge H^{\underline{Y}}$, it follows that $Y \models F^{\underline{Y}}$ and so, $F^{\underline{Y}} \not\equiv \bot$. Thus, the "only if" part of the equivalence holds, too. The case of $\vee$ is essentially the same.

It remains to consider the case $F = G \to H$. For the "if" part, as before, it suffices to notice that if $Y \not\models F$, then $F^{\underline{Y}} = \bot$. Conversely, let $Y \models F$.

Case 1. $Y \not\models G$. Then, $F^{\underline{Y}} = \top$ and so, $F^{\underline{Y}} \not\equiv \bot$.

Case 2. $Y \models G$. Since $Y \models F$, $Y \models H$ follows. Consequently, $F^{\underline{Y}} = G \to H^{\underline{Y}}$. In addition, by Proposition 1, $Y \models H^{\underline{Y}}$. Thus, $Y \models F^{\underline{Y}}$ and $F^{\underline{Y}} \not\equiv \bot$.

(2) We consider the case $\circ = \wedge$ only. The case $\circ = \vee$ is similar. If $Y \models F \wedge G$ then $(F \wedge G)^{\underline{Y}}$ and $F^{\underline{Y}} \wedge G^{\underline{Y}}$ are equal! Thus, let us assume that $Y \not\models F \wedge G$. It follows that $(F \wedge G)^{\underline{Y}} = \bot$. Moreover, we have $Y \not\models F$ or $Y \not\models G$. By (1), $F^{\underline{Y}} \equiv \bot$ or $G^{\underline{Y}} \equiv \bot$. Thus, $F^{\underline{Y}} \wedge G^{\underline{Y}} \equiv \bot$, and the claim follows. □

3.3. Minimal-model property

The main objective of Faber et al. [16] was to generalize the stable-model semantics to the class of theories consisting of rules of the form (2) so that stable models would be minimal models. Faber et al. proved that their generalization indeed has that property.

The extended FLP semantics has the minimal-model property for a broad class of theories, including those consisting of rules (2), but not in general.

Example 6. Let $F = \neg A \vee A$ and $Y = \emptyset$. Since $Y \models A \to \bot$ and $Y \not\models A$, $(\neg A)^{\underline{Y}} = (A \to \bot)^{\underline{Y}} = \top$. Moreover, $A^{\underline{Y}} = \bot$. Thus, $F^{\underline{Y}} \equiv (\neg A)^{\underline{Y}} \vee A^{\underline{Y}} \equiv \top$. Clearly, Y is a minimal model of $F^{\underline{Y}}$ and so, an FLP-stable model of F. Next, let us consider $Z = \{A\}$. We now have $(\neg A)^{\underline{Z}} = (A \to \bot)^{\underline{Z}} = \bot$ and $A^{\underline{Z}} = A$. Thus, $F^{\underline{Z}} \equiv (\neg A)^{\underline{Z}} \vee A^{\underline{Z}} \equiv A$. Again, Z is a minimal model of $F^{\underline{Z}}$ and so, an FLP-stable model of F. Thus, FLP-stable models of F do not form an antichain and Z is not a minimal model of F.

To describe a broad class of theories for which FLP-stable models are minimal models, we introduce monotone and disjunctive-monotone formulas.

Definition 2. A formula F is *monotone* if for every $X \subseteq Y \subseteq At$, $X \models F$ implies $Y \models F$. A formula F is *disjunctive-monotone* if every occurrence of $\vee$ in F operates on monotone formulas.

We note that a disjunctive-monotone formula does not have to be monotone. For instance, $A \wedge \neg C$ is disjunctive-monotone (as it contains no occurrence of $\vee$) but not monotone. Similarly, a monotone formula is not necessarily disjunctive-monotone. For instance, $\top \vee \neg A$ is monotone but not disjunctive monotone.

Proposition 4. *For every disjunctive-monotone formula F and every sets of atoms X and Y such that $X \subseteq Y$, if $X \models F$ and $Y \models F$ then $X \models F^{\underline{Y}}$.*

Proof. We proceed by induction. The case of $F = \bot$ is vacuously true. If $F = A$, where A is an atom, then $F^{\underline{Y}} = A = F$ (it follows from the assumption that $Y \models A$). Thus, $X \models F^{\underline{Y}}$ (as $X \models F$). For the inductive step, there are three cases to consider.

Case 1. $F = G \wedge H$. Since F is disjunctive-monotone, G and H are disjunctive-monotone, too, and we also have $X \models G$, $X \models H$, $Y \models G$ and $Y \models H$. By the induction hypothesis, $X \models G^{\underline{Y}}$ and $X \models H^{\underline{Y}}$. Consequently, $X \models G^{\underline{Y}} \wedge H^{\underline{Y}} = (G \wedge H)^{\underline{Y}} = F^{\underline{Y}}$.

Case 2. $F = G \vee H$. Since $X \models F$, $X \models G$ or $X \models H$. Without loss of generality, we may assume that $X \models G$. Since F is disjunctive-monotone, G is disjunctive-monotone. Moreover, G is monotone. Thus, $Y \models G$. By the induction hypothesis, $X \models G^{\underline{Y}}$. Since $F^{\underline{Y}} = G^{\underline{Y}} \vee H^{\underline{Y}}$, $X \models F^{\underline{Y}}$.

Case 3. $F = G \to H$. Since $Y \models F$, $F^{\underline{Y}} \neq \bot$. If $F^{\underline{Y}} = \top$ then $X \models F^{\underline{Y}} = \top$. If, on the other hand, $Y \models G$, $Y \models H$ and $F^{\underline{Y}} = G \to H^{\underline{Y}}$, then there are two cases to consider. If $X \not\models G$, then $X \models F^{\underline{Y}}$. If $X \models G$, then $X \models H$. Since H is disjunctive-monotone (as F is), by induction it holds that $X \models H^{\underline{Y}}$. Thus, $X \models F^{\underline{Y}}$ in that case, too. □

Corollary 2. *Let $\mathcal{F}$ be a theory such that every formula in $\mathcal{F}$ is of the form H or $G \to H$, where H is disjunctive-monotone. For every $X \subseteq Y \subseteq At$, if $X \models \mathcal{F}$ and $Y \models \mathcal{F}$, then $X \models \mathcal{F}^{\underline{Y}}$.*

Proof. To prove the result, it suffices to prove it for each formula F in $\mathcal{F}$. If F is disjunctive-monotone, then the result follows from Proposition 4. If $F = G \to H$, where H is disjunctive-monotone, we reason as follows. Since $Y \models F$, we have $F^{\underline{Y}} = \top$; or $Y \models G$, $Y \models H$ and $F^{\underline{Y}} = G \to H^{\underline{Y}}$. In the first case, $X \models F^{\underline{Y}}$ is evident. In the second case, if $X \not\models G$, the assertion follows. Otherwise, since $X \models F$, $X \models H$. By Proposition 4, $X \models H^{\underline{Y}}$ follows. Consequently, $X \models F^{\underline{Y}}$ follows, as well. □

Proposition 4 and Corollary 2 imply that for the class of theories of the type considered in Corollary 2, FLP-stable models are minimal.

Corollary 3. *Let $\mathcal{F}$ be a theory such that every formula in $\mathcal{F}$ is of the form H or $G \to H$, where H is disjunctive-monotone. If Y is an FLP-stable model of $\mathcal{F}$ then Y is a minimal model of $\mathcal{F}$.*

Proof. Since Y is a model of $\mathcal{F}^{\underline{Y}}$, Y is a model of $\mathcal{F}$ (Proposition 1). Let us assume that $X \models \mathcal{F}$ and $X \subseteq Y$. By Corollary 2, $X \models \mathcal{F}^{\underline{Y}}$. Since Y is a minimal model of $\mathcal{F}^{\underline{Y}}$, $X = Y$. Thus, Y is a minimal model of $\mathcal{F}$. □

Corollary 3 extends the result by Faber et al., as it applies in particular to theories consisting of formulas of type (2). It can be generalized further to the case, where each formula in a theory is of the form $H_k \to (H_{k-1} \to (\cdots \to (H_1 \to H_0)\cdots))$, where $k \geqslant 0$ and H_0 is disjunctive monotone. The argument is essentially the same.

3.4. Computational complexity for FLP semantics

It is well known that the truth value of a formula in an interpretation can be found in polynomial time. It follows that given a formula and a set of atoms Y, one can compute $F^{\underline{Y}}$ in polynomial time by means of a simple recursive algorithm that directly follows the definition of the reduct. Further, we have that the problem to decide whether a model of a formula is a minimal model is in the class coNP (in fact, one can show it is coNP-complete). Indeed, the complementary problem, to decide whether a given model of a formula is *not* a minimal one is in NP (a model properly contained in the given one serves as a witness). Thus, the problem to decide whether a formula has an FLP-stable model is in the class Σ_2^P. The completeness of the problem for the class Σ_2^P follows from the fact that on disjunctive programs FLP-stable models coincide with stable models [16], and the existence problem for stable models is Σ_2^P-complete [15]. Consequently, deciding the existence of an FLP-stable model is Σ_2^P-complete, too. We state that result below, together with two other related results that can be proved by similar arguments.

Theorem 4. *The problem of the existence of an FLP-stable model is Σ_2^P-complete. Skeptical reasoning with FLP-stable models (is a given atom a member of every FLP-stable model) is Π_2^P-complete. Brave reasoning with FLP-stable models (is a given atom a member of some FLP-stable model) is Σ_2^P-complete.*

3.5. HT-interpretations and FLP semantics — strong equivalence

We now describe FLP-stable models in terms of HT-interpretations, and apply that result to characterize strong equivalence with respect to the FLP semantics. First, we define a certain satisfiability relation $\models_{flp}$ between HT-interpretations and formulas. The definition is inductive and follows the same pattern as that for $\models_{ht}$. The cases $\langle X, Y\rangle \models_{flp} F$ for $F = \bot$, $F = A$, where $A \in At$, $F = G \wedge H$ and $G \vee H$, are handled as in the case of $\models_{ht}$. For the implication we have the following clause:

5′. $\langle X, Y\rangle \models_{flp} G \to H$ if $Y \models G \to H$; and $Y \not\models G$, or $X \not\models G$, or $\langle X, Y\rangle \models_{flp} H$.

The relation $\models_{flp}$ extends in a standard way to HT-interpretations and *sets* of formulas. If $\mathcal{F}$ is a theory and $\langle X, Y\rangle \models_{flp} \mathcal{F}$, we say that $\langle X, Y\rangle$ is an *FLP-model* of $\mathcal{F}$ (not to be confused with an FLP-*stable* model).

We have the following simple property of $\models_{flp}$, mirroring a similar one for $\models_{ht}$ (cf. Theorem 1).

Theorem 5. *For every formula F and every sets $X \subseteq Y \subseteq At$:*

(1) $\langle X, Y\rangle \models_{flp} F$ *implies* $Y \models F$;
(2) $\langle X, Y\rangle \models_{flp} \neg F$ *if and only if* $Y \not\models F$;
(3) $\langle Y, Y\rangle \models_{flp} F$ *if and only if* $Y \models F$.

Proof. (1) The case $F = \bot$ is evident. If $F = A$, where $A \in At$, and $\langle X, Y\rangle \models_{flp} F$, then $A \in X$. Thus, $A \in Y$ and $Y \models F$. The inductive step for $F = G \wedge H$ and $F = G \vee H$ is standard. If $F = G \to H$ and $\langle X, Y\rangle \models F$ then, in particular, $Y \models F$ (by the definition of $\models_{flp}$ for the case of implication). Thus, the claim follows.

(2) By the definition, $\langle X, Y\rangle \models_{flp} \neg F$ if and only if $Y \models \neg F$, and $Y \not\models F$ or $X \not\models F$ or $\langle X, Y\rangle \models_{flp} \bot$. Thus, $\langle X, Y\rangle \models_{flp} \neg F$ if and only if $Y \models \neg F$ and the claim follows.

(3) We show only the argument in the inductive step for the case $F = G \to H$ (the basis and all other cases are straightforward). First, by the definition, if $\langle Y, Y\rangle \models_{flp} F$ then $Y \models F$. Conversely, if $Y \models F$, then $Y \not\models G$ or $Y \models H$. By the induction hypothesis, $Y \models H$ is equivalent to $\langle Y, Y\rangle \models_{flp} H$. Thus, $\langle Y, Y\rangle \models_{flp} F$. □

Similarly as the relation $\models_{ht}$ and the F-reduct, the $\models_{flp}$ relation and the FLP-reduct are closely connected (cf. Theorem 2).

Theorem 6. *For every formula F and for every two sets of atoms $X \subseteq Y$, $X \models F^{\underline{Y}}$ if and only if $\langle X, Y\rangle \models_{flp} F$.*

Proof. We proceed by induction. The case when $F = \bot$ is straightforward. Let $F = A$, where $A \in At$. If $X \models A^{\underline{Y}}$, then $A^{\underline{Y}} \neq \bot$. Thus, $A^{\underline{Y}} = A$. It follows that $X \models A$ and so, $\langle X, Y\rangle \models A$. Conversely, if $\langle X, Y\rangle \models A$, then $X \models A$. Since $X \subseteq Y$, $Y \models A$. Thus, $A^{\underline{Y}} = A$ and $X \models A^{\underline{Y}}$ as required.

Next, let $F = G \wedge H$. If $X \models (G \wedge H)^{\underline{Y}}$, then $(G \wedge H)^{\underline{Y}} = G^{\underline{Y}} \wedge H^{\underline{Y}}$. Thus, $X \models G^{\underline{Y}}$ and $X \models H^{\underline{Y}}$. By the inductive hypothesis, $\langle X, Y\rangle \models G$ and $\langle X, Y\rangle \models H$. Thus, $\langle X, Y\rangle \models G \wedge H$, as needed. Conversely, let $\langle X, Y\rangle \models G \wedge H$. Then $\langle X, Y\rangle \models G$ and $\langle X, Y\rangle \models H$ and, by the inductive hypothesis, $X \models G^{\underline{Y}}$ and $X \models H^{\underline{Y}}$. Thus, $X \models G^{\underline{Y}} \wedge H^{\underline{Y}}$. By Proposition 3, $G^{\underline{Y}} \wedge H^{\underline{Y}} \equiv (G \wedge H)^{\underline{Y}}$. Thus, $X \models (G \wedge H)^{\underline{Y}}$.

The argument for the case $F = G \vee H$ is similar. Thus, we move on to the case $F = G \to H$. We have the following equivalences:

(1) $X \models (G \to H)^{\underline{Y}}$;
(2) $Y \not\models G$; or $Y \models G$ and $Y \models H$, and $X \models G \to H^{\underline{Y}}$;
(3) $Y \not\models G$; or $Y \models H$ and $X \models G \to H^{\underline{Y}}$;
(4) $Y \not\models G$ or $Y \models H$; and $Y \not\models G$ or $X \models G \to H^{\underline{Y}}$;
(5) $Y \models G \to H$; and $Y \not\models G$ or $X \not\models G$, or $X \models H^{\underline{Y}}$;
(6) $Y \models G \to H$; and $Y \not\models G$ or $X \not\models G$, or $\langle X, Y\rangle \models_{flp} H$.

The last statement is equivalent to $\langle X, Y\rangle \models_{flp} F$ and the result follows. □

Theorem 6 is the key to a characterization of FLP-stable models in terms of the relation $\models_{flp}$.

Corollary 4. *Let $\mathcal{F}$ be a theory and Y a set of atoms. Then Y is an FLP-stable model of $\mathcal{F}$ if and only if $\langle Y, Y\rangle \models_{flp} \mathcal{F}$ and for every $X \subset Y$, $\langle X, Y\rangle \not\models_{flp} \mathcal{F}$.*

Proof. By the definition, Y is an FLP-stable model of $\mathcal{F}$ if and only if $Y \models \mathcal{F}^{\underline{Y}}$ and, for every $X \subset Y$, $X \not\models \mathcal{F}^{\underline{Y}}$. We apply Theorem 6. The former condition is equivalent to $\langle Y, Y\rangle \models_{flp} \mathcal{F}$. The latter one can be stated equivalently as: for every $X \subset Y$, $\langle X, Y\rangle \not\models_{flp} \mathcal{F}$. Thus, the assertion follows. □

Example 7. Let us consider the theory $\mathcal{E}_2 = \{(A \vee \neg A) \to A\}$ from Example 2. Let $Y = \emptyset$. Then $Y \not\models (A \vee \neg A) \to A$, and so, $\langle Y, Y\rangle \not\models_{flp} \mathcal{E}_2$. Thus, in agreement with our results, $Y = \emptyset$ is not an FLP-stable model of $\mathcal{E}_2$.

For $Z = \{A\}$, the situation is different. First, we note that now $Z \models (A \vee \neg A) \to A$ and $\langle Z, Z\rangle \models_{flp} A$ (as $A \in Z$). Thus, $\langle Z, Z\rangle \models_{flp} (A \vee \neg A) \to A$ and so, $\langle Z, Z\rangle \models_{flp} \mathcal{E}_2$. Clearly, $X = \emptyset$ is the only proper subset of Z and $\langle X, Z\rangle \not\models_{flp} (A \vee \neg A) \to A$ (indeed, we have $Z \models A \vee \neg A$, $X \models A \vee \neg A$ and $\langle X, Z\rangle \not\models_{flp} A$ (as $A \notin X$)). According to Corollary 4, $\{A\}$ is an FLP-stable model of $\mathcal{E}_2$ (as we already established by means of the reduct-based definition in Example 2).

We conclude this section with a discussion of the notion of strong FLP-equivalence. Theories $\mathcal{F}$ and $\mathcal{G}$ are *strongly FLP-equivalent* if for every theory $\mathcal{H}$, the theories $\mathcal{F}\cup\mathcal{H}$ and $\mathcal{G}\cup\mathcal{H}$ have the same FLP-stable models. This is a literal adaptation of the standard definition of strong equivalence [32] to the case of FLP-stable models.

Theorem 7. *Let $\mathcal{F}$ and $\mathcal{G}$ be two formulas. Then, $\mathcal{F}$ and $\mathcal{G}$ are strongly FLP-equivalent if and only if $\mathcal{F}$ and $\mathcal{G}$ have the same FLP-models.*

Proof. ($\Leftarrow$) For every theory $\mathcal{H}$, $\langle X,Y\rangle \models_{flp} \mathcal{F}\cup\mathcal{H}$ if and only if $\langle X,Y\rangle \models_{flp} \mathcal{G}\cup\mathcal{H}$. By Corollary 4, $\mathcal{F}\cup\mathcal{H}$ and $\mathcal{G}\cup\mathcal{H}$ have the same FLP-stable models.

($\Rightarrow$) Let us assume that there are $X\subseteq Y\subseteq At$ such that $\langle X,Y\rangle$ satisfies one of $\mathcal{F}$ and $\mathcal{G}$ but not the other. Without loss of generality, we may assume that $\langle X,Y\rangle \models_{flp}\mathcal{F}$ and $\langle X,Y\rangle \not\models_{flp}\mathcal{G}$. By Theorem 6, it follows that $X\models \mathcal{F}^{\underline{Y}}$ and $X\not\models \mathcal{G}^{\underline{Y}}$. The first property implies that $\mathcal{F}^{\underline{Y}}\not\equiv\bot$. Consequently, by Proposition 3, $Y\models\mathcal{F}$. By Proposition 1, $Y\models\mathcal{F}^{\underline{Y}}$.

Case 1. $Y\not\models\mathcal{G}^{\underline{Y}}$. It follows that $\langle Y,Y\rangle\not\models_{flp}\mathcal{G}$. Thus, $Y\not\models\mathcal{G}$ and for every $\mathcal{H}$, $Y\not\models\mathcal{G}\cup\mathcal{H}$. Thus, Y is not an FLP-stable model of $\mathcal{G}\cup\mathcal{H}$. Let us now define $\mathcal{H}=Y$. We have $(\mathcal{F}\cup\mathcal{H})^{\underline{Y}}\equiv\mathcal{F}^{\underline{Y}}\cup\mathcal{H}^{\underline{Y}}$. Moreover, $\mathcal{H}^{\underline{Y}}=\mathcal{H}$. Thus, $(\mathcal{F}\cup\mathcal{H})^{\underline{Y}}\equiv\mathcal{F}^{\underline{Y}}\cup\mathcal{H}$. It follows that (a) $Y\models(\mathcal{F}\cup\mathcal{H})^{\underline{Y}}$, and (b) there is no $X\subset Y$ such that $X\models(\mathcal{F}\cup\mathcal{H})^{\underline{Y}}$. Thus, Y is an FLP-stable model of $\mathcal{F}\cup\mathcal{H}$. As we noted, Y is not an FLP-stable model of $\mathcal{G}\cup\mathcal{H}$. Thus, $\mathcal{F}$ and $\mathcal{G}$ are not strongly FLP-equivalent, a contradiction.

Case 2. $Y\models\mathcal{G}^{\underline{Y}}$. We recall that $X\not\models\mathcal{G}^{\underline{Y}}$. Thus, $X\subset Y$. We define

$$\mathcal{H}=X\cup\{A\rightarrow B\mid A,B\in Y\setminus X\}.$$

We have $(\mathcal{F}\cup\mathcal{H})^{\underline{Y}}\equiv\mathcal{F}^{\underline{Y}}\cup\mathcal{H}^{\underline{Y}}$. Moreover, it is easy to check that $\mathcal{H}^{\underline{Y}}=\mathcal{H}$. Thus, $(\mathcal{F}\cup\mathcal{H})^{\underline{Y}}\equiv\mathcal{F}^{\underline{Y}}\cup\mathcal{H}$. We recall that $X\models\mathcal{F}^{\underline{Y}}$. We also have $X\models\mathcal{H}$. Thus, $X\models(\mathcal{F}\cup\mathcal{H})^{\underline{Y}}$ and so, Y is not an FLP-model of $\mathcal{F}\cup\mathcal{H}$. Since $(\mathcal{G}\cup\mathcal{H})^{\underline{Y}}\equiv\mathcal{G}^{\underline{Y}}\cup\mathcal{H}$, $Y\models\mathcal{G}^{\underline{Y}}$, and $Y\models\mathcal{H}$, we have $Y\models(\mathcal{G}\cup\mathcal{H})^{\underline{Y}}$. Let $Z\subset Y$ be such that $Z\models\mathcal{G}^{\underline{Y}}\cup\mathcal{H}$. Since $Z\models\mathcal{H}$, $Z=X$. However, $X\not\models\mathcal{G}^{\underline{Y}}$, a contradiction. Thus, Y is a minimal model of $(\mathcal{G}\cup\mathcal{H})^{\underline{Y}}$ and so a stable model of $\mathcal{G}\cup\mathcal{H}$. This contradicts our assumption that $\mathcal{F}$ and $\mathcal{G}$ are FLP-equivalent. Consequently, $\mathcal{F}$ and $\mathcal{G}$ have the same FLP-models. □

4. Normal forms and a comparison with stable-model semantics

The following result was obtained by Cabalar and Ferraris [8]. It concerns representing theories by programs — theories consisting of rules (formulas of the form (1)).

Theorem 8. *For every theory $\mathcal{F}$ there is a program $\mathcal{G}$ (in the same language) such that $\mathcal{F}$ and $\mathcal{G}$ have the same HT-models (are equivalent in the logic HT).*

In other words, every theory $\mathcal{F}$ is strongly equivalent to some program $\mathcal{G}$. A similar result holds for the FLP-models and, in fact, it can be obtained by means of a very similar argument to that used by Cabalar and Ferraris [8]. In what follows we write $\neg Y$ for $\{\neg y\mid y\in Y\}$. We first state and prove three auxiliary results (analogous to results proved by Cabalar and Ferraris [8] for HT-countermodels).

Proposition 5. *Let $X\subset Y\subseteq Z$ be finite. Then $\langle U,V\rangle$, where $U\subseteq V\subseteq Z$, is an FLP-countermodel of $\bigwedge X\wedge\bigwedge\neg\overline{Y}\rightarrow\bigvee\overline{X}\vee\bigvee\neg Y$ (where the set complements $\overline{X}$ and $\overline{Y}$ are defined with respect to Z) if and only if $U=X$ and $V=Y$.*

Proof. Let us denote $\bigwedge X\wedge\bigwedge\neg\overline{Y}\rightarrow\bigvee\overline{X}\vee\bigvee\neg Y$ by F. Since $Y\setminus X\neq\emptyset$, there is $a\in Y\setminus X$. It follows that $a\vee\neg a$ is a subformula of $\bigvee\overline{X}\vee\bigvee\neg Y$. Thus, F is a propositional tautology.

Let us consider a pair $\langle U,V\rangle$, where $U\subseteq V$. Due to the observation above, $\langle U,V\rangle$ is an FLP-countermodel of F if and only if

$$V\models\bigwedge X\wedge\bigwedge\neg\overline{Y},\quad U\models\bigwedge X\wedge\bigwedge\neg\overline{Y},\quad\text{and}\quad\langle U,V\rangle\not\models_{flp}\bigvee\overline{X}\vee\bigvee\neg Y.$$

Moreover, from the definition, one can check that $\langle U,V\rangle\not\models_{flp}\bigvee\overline{X}\vee\bigvee\neg Y$ if and only if $U\not\models\bigvee\overline{X}$ and $V\not\models\bigvee\neg Y$.

Now, since $U\models\bigwedge X$ and $U\not\models\bigvee\overline{X}$, $X\subseteq U$ and $U\cap\overline{X}=\emptyset$. Thus, $U=X$. Since $V\models\bigwedge\neg\overline{Y}$ and $V\not\models\bigvee\neg Y$, $V\subseteq Y$ and $Y\subseteq V$. Thus, $V=Y$.

Conversely, if $U=X$ and $V=Y$ then $V\models\bigwedge X\wedge\bigwedge\neg\overline{Y}$, $U\models\bigwedge X\wedge\bigwedge\neg\overline{Y}$, $U\not\models\bigvee\overline{X}$ and $V\not\models\bigvee\neg Y$. Thus, $\langle U,V\rangle$ is an FLP-countermodel of F. □

Proposition 6. *Let $Y\subseteq Z$ be finite. Then $\langle U,V\rangle$, where $U\subseteq V\subseteq Z$, is an FLP-countermodel to $\bigwedge Y\wedge\bigwedge\neg\overline{Y}\rightarrow\bot$ (where the set complement $\overline{Y}$ is defined with respect to Z) if and only if $V=Y$.*

Proof. By the definition, $\langle U, V\rangle$ is an FLP-countermodel to $Y \wedge \bigwedge \neg\overline{Y} \rightarrow \bot$ if and only if (1) $V \models Y \wedge \bigwedge \neg\overline{Y}$, or (2) $V \models Y \wedge \bigwedge \neg\overline{Y}$ and $U \models Y \wedge \bigwedge \neg\overline{Y}$. The condition (1) is equivalent to $V = Y$. The condition (2) is equivalent to $U = V = Y$. Thus the disjunction of the two conditions is equivalent to $V = Y$, as required. □

Proposition 7. *Let F be a formula. If $\langle Y, Y\rangle$ is an FLP-countermodel of F, then for every $X \subseteq Y$, $\langle X, Y\rangle$ is an FLP-countermodel of F.*

Proof. If $\langle Y, Y\rangle$ is an FLP-countermodel of F, then $Y \not\models F^{\underline{Y}}$ (Theorem 6) and so, $Y \not\models F$ (Proposition 1). Consequently, by Theorem 5, $\langle X, Y\rangle$ is an FLP-countermodel of F. □

Theorem 9. *Let $\mathcal{F}$ be a theory. There exists a program $\mathcal{G}$ such that $\mathcal{F}$ and $\mathcal{G}$ have the same FLP-models.*

Proof. For $F \in \mathcal{F}$, we consider FLP-countermodels $\langle X, Y\rangle$ of F such that $Y \subseteq At(F)$. For each FLP-countermodel $\langle X, Y\rangle$ with $X \neq Y$, we take the formula defined in Proposition 5 (with $Z = At(F)$). For each countermodel $\langle X, Y\rangle$ such that $X = Y$, we take the formula from Proposition 6. We take for $\mathcal{G}$ the set of all rules constructed in that way from countermodels of formulas in $\mathcal{F}$. By Proposition 7, $\mathcal{F}$ and $\mathcal{G}$ have the same FLP-countermodels consisting of atoms in $At(\mathcal{F})$ and so, the same FLP-models consisting of atoms in $At(\mathcal{F})$. Thus, they have the same FLP-models. □

We saw (Examples 1–2 and 4–5) that the semantics of stable and FLP-stable models are different and neither is stronger than the other one. However, each can be expressed in terms of the other one. To see that, we first observe that HT- and FLP-models of rules coincide.

Proposition 8. *Let R be a rule. Then, R has the same HT- and FLP-models.*

Proof. Let $R = \bigwedge A \wedge \bigwedge \neg B \rightarrow \bigvee C \vee \bigvee \neg D$. Directly from the definition, it follows that $\langle X, Y\rangle \models_{ht} R$ if and only if $Y \models R$; and $X \cap A = \emptyset$ or $B \subseteq Y$ or $C \subseteq X$ or $Y \cap D = \emptyset$.

Similarly, $\langle X, Y\rangle \models_{flp} R$ if and only if $Y \models R$; and $X \cap A = \emptyset$ or $B \subseteq X$ or $Y \cap A = \emptyset$ or $B \subseteq Y$ or $C \subseteq X$ or $Y \cap D = \emptyset$. Since $X \cap A = \emptyset$ or $Y \cap A = \emptyset$ if and only if $X \cap A = \emptyset$, and $B \subseteq X$ or $B \subseteq Y$ if and only if $B \subseteq Y$, the result follows. □

Theorems 8 and 9 yield now the following two corollaries relating the two semantics.

Corollary 5. *For every theory $\mathcal{F}$ there are programs $\mathcal{F}_{ht}$ and $\mathcal{F}_{flp}$ such that*

(1) *$\langle X, Y\rangle$ is an HT-model of $\mathcal{F}$ if and only if $\langle X, Y\rangle$ is an FLP-model of $\mathcal{F}_{ht}$;*
(2) *$\langle X, Y\rangle$ is an FLP-model of $\mathcal{F}$ if and only if $\langle X, Y\rangle$ is an HT-model of $\mathcal{F}_{flp}$.*

Proof. It is enough to take for $\mathcal{F}_{ht}$ and $\mathcal{F}_{flp}$ programs guaranteed by Theorems 8 and 9, respectively. □

Thus the meaning of any theory $\mathcal{F}$ under HT-models is captured by FLP-models of the normal form $\mathcal{F}_{ht}$ of $\mathcal{F}$ that is assured by Theorem 8. Similarly, the meaning of any theory $\mathcal{F}$ under FLP-models is captured by HT-models of the normal form $\mathcal{F}_{flp}$ of $\mathcal{F}$ given by Theorem 9. As another corollary we state a result relating stable and FLP-stable models of $\mathcal{F}$, $\mathcal{F}_{ht}$ and $\mathcal{F}_{flp}$.

Corollary 6. *For every theory $\mathcal{F}$:*

(1) *Y is a stable model of $\mathcal{F}$ if and only if Y is an FLP-stable model of $\mathcal{F}_{ht}$;*
(2) *Y is an FLP-stable model of $\mathcal{F}$ if and only if Y is a stable model of $\mathcal{F}_{flp}$.*

We mention that recently Lee and Meng [29] obtained a result related to Corollary 6(2) but restricted to formulas that are programs with aggregates. Namely, they showed that a *program*, say $\mathcal{F}$, as considered by Faber et al. [16] (and so, possibly with aggregates in the bodies of rules), can be compiled into propositional formula $FLP(\mathcal{F})$ so that FLP-stable answer sets of P correspond to stable models of $FLP(\mathcal{F})$.

We close by pointing out a drawback of the FLP semantics. Namely, the operator of $\leftrightarrow$ does not function, in general, as the operator of explicit definition.[2] For instance, if we introduce a new atom B, then the theory $\mathcal{E}_1' = \{\neg B \rightarrow A, \neg A \leftrightarrow B\}$ (we obtain it from $\mathcal{E}_1$ by replacing $\neg A$ with B and adding the "definition" $\neg A \leftrightarrow B$) has two FLP-stable models, $\{B\}$ and $\{A\}$, while the original theory $\mathcal{E}_1$ has only one FLP-stable model, $\{A\}$ (and, in particular, no counterpart to the stable model $\{B\}$).

[2] This aspect of the FLP semantics was pointed out by one of the reviewers.

5. Supported models

The approach that yielded generalizations of stable and FLP-stable model semantics for arbitrary propositional theories can also be applied to the supported-model semantics.

5.1. The reduct for the supported-model semantics

For a formula F and a set of atoms X, we define the *SPP-reduct* of F with respect to Y, written as $F^{\underline{Y}}_{=}$, by adapting to the new notation the inductive clauses (R1)–(R3), and using the following definition for the implication:

SPP4.

$$(G \to H)^{\underline{Y}}_{=} = \begin{cases} H^{\underline{Y}}_{=} & \text{if } Y \models G \text{ and } Y \models H, \\ \top & \text{if } Y \not\models G, \\ \bot & \text{otherwise.} \end{cases}$$

This notion of reduct is motivated by the definition of supported models in the case of programs with disjunctive rules [6,27]. We recall that definition. Let P be a disjunctive program. The *supp-reduct* of P with respect to a set of atoms Y, $P(Y)$, is the set of the heads of those rules in P whose body is satisfied by Y. A set of atoms Y is a *supported model* of P if Y is a minimal model of $P(Y)$.

The basic idea behind $P(Y)$ is to drop the antecedent of a rule if the antecedent is satisfied by Y. We adopt that idea in the definition of the SPP-reduct. However, in the first case of the definition, we set the reduct $(G \to H)^{\underline{Y}}_{=}$ to be $H^{\underline{Y}}_{=}$ rather than H due to the same reasons we discussed when generalizing the FLP-reduct. With the definition of the reduct in hand, the definition of a supported model is standard.

Definition 3. Let $\mathcal{F}$ be a theory. A set of atoms Y is a *supported model* of $\mathcal{F}$ if Y is a minimal model of $\mathcal{F}^{\underline{Y}}_{=}$.

Example 8. We will one more time consider theories $\mathcal{E}_1 = \{\neg\neg A \to A\}$ and $\mathcal{E}_2 = \{(A \vee \neg A) \to A\}$ from Examples 1 and 2, respectively. First, let $Y = \emptyset$. Clearly, we have $Y \not\models \neg\neg A$. Thus, $\mathcal{E}_1^{\underline{Y}}{}_{=} = \top$. It follows that $Y \models \mathcal{E}_1$ and, as $Y = \emptyset$, Y is a supported model of $\mathcal{E}_1$ being trivially a minimal model of $\mathcal{E}_1^{\underline{Y}}{}_{=}$. On the other hand, $Y \not\models (A \vee \neg A) \to A$. Thus, $\mathcal{E}_2^{\underline{Y}}{}_{=} = \bot$ and Y is not a supported model of $\mathcal{E}_2$.

Next, let $Z = \{A\}$. Then, $Z \models \neg\neg A$ and $Z \models A$. Thus, $\mathcal{E}_1^{\underline{Z}}{}_{=} = (\neg\neg A \to A)^{\underline{Z}}_{=} = A^{\underline{Z}}_{=} = A$. It follows that Z is a supported model of $\mathcal{E}_1$. Similarly, $Z \models (A \vee \neg A)$ and $Z \models A$. Thus, $\mathcal{E}_2^{\underline{Z}}{}_{=} = ((A \vee \neg A) \to A)^{\underline{Z}}_{=} = A^{\underline{Z}}_{=} = A$, and Z is a supported model of $\mathcal{E}_2$, too.

The results and the proofs that worked in the case of stable and FLP semantics work, with only minor changes (and with one exception) in the case of supported models, too. We start by gathering in one result several basic properties of the SPP-reduct and supported models.

Proposition 9. *For every theory $\mathcal{F}$ and every set of atoms Y:*

(1) *$Y \models \mathcal{F}$ if and only if $Y \models \mathcal{F}^{\underline{Y}}_{=}$;*
(2) *$Y \models \mathcal{F}$ if and only if $\mathcal{F}^{\underline{Y}}_{=} \not\equiv \bot$;*
(3) *If Y is a supported model of $\mathcal{F}$, then Y is a model of $\mathcal{F}$;*
(4) *If Y is a supported model of $\mathcal{F}$, then every atom in Y has a head occurrence in $\mathcal{F}$.*

Proof. (1) It is enough to consider the case when $\mathcal{F}$ consists of a single formula F. We proceed by structural induction. The base cases of $F = \bot$ and $F = A$, where $A \in At$, are straightforward. Let us consider $F = G \wedge H$. If $Y \not\models F$, then $F^{\underline{Y}}_{=} = \bot$. Thus, both sides of the equivalence are false, and the equivalence follows. If $Y \models F$, then $F^{\underline{Y}}_{=} = G^{\underline{Y}}_{=} \wedge H^{\underline{Y}}_{=}$. By the definition and the inductive hypothesis, the following statements are equivalent:

$$Y \models F,$$
$$Y \models G \wedge H,$$
$$Y \models G, \quad \text{and} \quad Y \models H,$$
$$Y \models G^{\underline{Y}}_{=} \quad \text{and} \quad Y \models H^{\underline{Y}}_{=},$$
$$Y \models G^{\underline{Y}}_{=} \wedge H^{\underline{Y}}_{=},$$
$$Y \models F^{\underline{Y}}_{=}.$$

Thus, again the required equivalence holds. The argument for $\vee$ is similar. Let then $F = G \to H$. If $Y \not\models F$, then $F^{\underline{Y}} = \bot$ and the equivalence in the assertion holds. Similarly, if $Y \not\models G$, then $F^{\underline{Y}} = \top$, and both $Y \models F$ and $Y \models F^{\underline{Y}}$ hold. Finally, let us assume that $Y \models G$ and $Y \models H$. In this case, $F^{\underline{Y}} = H^{\underline{Y}}$. By the inductive hypothesis, $Y \models H^{\underline{Y}}$ and so, $Y \models F^{\underline{Y}}$. Thus, also in that case, both $Y \models F$ and $Y \models F^{\underline{Y}}$ hold.

(2) As before, it is enough to consider the case when $\mathcal{F}$ consists of a single formula F. If $Y \models F$ then (1) implies that $F^{\underline{Y}} \not\equiv \bot$. We prove the converse implication by induction. If $F = \bot$, then the implication is trivially true. If $F = A$, where A is an atom, then $F^{\underline{Y}} = A$ (as the assumption excludes the only other possibility that $F^{\underline{Y}} \neq \bot$). It follows that $A \in Y$ and so, $Y \models F$. Next, let $F = G \wedge H$. Since $F^{\underline{Y}} \not\equiv \bot$, $F^{\underline{Y}} = G^{\underline{Y}} \wedge H^{\underline{Y}}$. Moreover, $G^{\underline{Y}} \not\equiv \bot$ and $H^{\underline{Y}} \not\equiv \bot$. By the induction hypothesis, $Y \models G$ and $Y \models H$. Thus, $Y \models F$. The case $F = G \vee H$ is similar. Finally, if $F = G \to H$, we have that either $Y \not\models G$, or $Y \models G$, $Y \models H$ and $F^{\underline{Y}} = H^{\underline{Y}}$. In either case, $Y \models F$.

(3) Follows from (1) and from the definition of a supported model.

(4) We first show that for every formula F and every set of atoms S containing all atoms with head occurrences in F, if $Y \subseteq At$ and $Y \models F$ then $Y \cap S \models F^{\underline{Y}}$. We proceed by induction. If $F = \bot$, the claim is trivially true. If $F = A$, then $A \in S$. If $Y \models F$, then $A \in Y$ and $F^{\underline{Y}} = A$. Thus, the claim follows. If $F = G \wedge H$ or $G \vee H$, then $Y \models G$ *and* (*or*, respectively) $Y \models H$. Moreover, atoms with head occurrences in G are contained in S and the same holds for H. Thus, the induction hypothesis applies to G and H. Consequently, we have $Y \cap S \models G$ *and* (*or*, respectively) $Y \cap S \models H$, and the claim follows.

Finally, let $F = G \to H$. Since $Y \models F$, $F^{\underline{Y}} = \top$, or $Y \models G$, $Y \models H$ and $F^{\underline{Y}} = H^{\underline{Y}}$. In the first case, the assertion is evident. In the latter case, we have $Y \models H$. By the induction hypothesis (it applies, as every atom with a head occurrence in H has a head occurrence in F and so, it belongs to S), $Y \cap S \models H^{\underline{Y}}$. Thus, $Y \cap S \models F^{\underline{Y}}$ in that case, too.

We are ready to prove the assertion (4). Let S be the set of atoms with head occurrences in $\mathcal{F}$. Since Y is a supported model of $\mathcal{F}$, $Y \models F^{\underline{Y}}$. By (1), $Y \models \mathcal{F}$. Thus, by the property proved above, $Y \cap S \models \mathcal{F}^{\underline{Y}}$. Since Y is a minimal model of $F^{\underline{Y}}$, and $Y \cap S \subseteq Y$, $Y \cap S = Y$. Consequently, $Y \subseteq S$. □

We now observe that our concept of a supported model indeed generalizes that of a disjunctive program [6].

Theorem 10. *Let P be a disjunctive program. Then, $Y \subseteq At$ is a supported model of P according to our definition if and only if Y is a supported model according to the original definition of supported models of disjunctive logic programs.*

Proof. If Y is a supported model according to either definition, Y is a model of P. Using this observation, as well as the definitions of the corresponding reducts, one can show that $H_1 \vee \cdots \vee H_k \in P^{\underline{Y}}$ if and only if $k \geqslant 1$ and there are atoms $H_{k+1}, \ldots, H_m$ such that $H_i \notin Y$, $k+1 \leqslant i \leqslant m$, and $H_1 \vee \cdots \vee H_k \vee H_{k+1} \vee \cdots \vee H_m \in P(Y)$. Let us assume that Y is a supported model of P according to the original definition. It follows that Y is a minimal model of $P(Y)$. By the observation above, Y is a model of $P^{\underline{Y}}$. If $Z \subseteq Y$ is a model of $P^{\underline{Y}}$, then Z is a model of $P(Y)$ (again, by the observation above, we have that $P^{\underline{Y}}$ classically entails $P(Y)$). Thus, $Z = Y$ and Y is a minimal model of $P^{\underline{Y}}$. Consequently, Y is a supported model of P according to our definition.

Conversely, let Y be a supported model of P according to our definition. Then, Y is a minimal model of $P^{\underline{Y}}$. Since $P^{\underline{Y}}$ classically entails $P(Y)$, Y is a model of $P(Y)$. Let $Z \subseteq Y$ be a model of $P(Y)$. Let $H_1 \vee \cdots \vee H_k \in P^{\underline{Y}}$. By the observation above, there are atoms $H_{k+1}, \ldots, H_m$ such that $H_i \notin Y$, $k+1 \leqslant i \leqslant m$, and $H_1 \vee \cdots \vee H_k \vee H_{k+1} \vee \cdots \vee H_m \in P(Y)$. It follows that $H_i \notin Z$, $k+1 \leqslant i \leqslant m$. Since Z is a model of $P(Y)$, it follows that Z is a model of $H_1 \vee \cdots \vee H_k$. Thus, Z is a model of $P^{\underline{Y}}$ and, consequently, $Z = Y$. Thus, Y is a minimal model of $P(Y)$ and so, a supported model of P according to the original definition. □

5.2. HT-interpretations and supported models

Next, we characterize supported models in terms of a certain satisfiability relation that connects HT-interpretations and formulas. It follows closely the definitions of $\models_{ht}$ and $\models_{flp}$ but is modified for the case of the implication.

5″. $\langle X, Y\rangle \models_{spp} G \to H$ if $Y \models G \to H$, and $Y \not\models G$ or $\langle X, Y\rangle \models_{spp} H$.

If $\langle X, Y\rangle \models_{spp} \mathcal{F}$, we say that $\langle X, Y\rangle$ is an *SPP-model* of $\mathcal{F}$.

Our next result gathers together some simple properties of the relation $\models_{spp}$ that mirror those of $\models_{ht}$ and $\models_{flp}$.

Theorem 11. *For every formula F and every sets $X \subseteq Y \subseteq At$:*

(1) $\langle X, Y\rangle \models_{spp} F$ *implies* $Y \models F$;
(2) $\langle X, Y\rangle \models_{spp} \neg F$ *if and only if* $Y \not\models F$;
(3) $\langle Y, Y\rangle \models_{spp} F$ *if and only if* $Y \models F$.

Proof. (1) We proceed by induction. The claim is evident for $F = \bot$. If $F = A$, where A is an atom, $\langle X, Y\rangle \models_{spp} F$ implies that $A \in X$. Thus, $A \in Y$ and $Y \models F$. If $F = G \wedge H$, then $\langle X, Y\rangle \models_{spp} F$ implies $\langle X, Y\rangle \models_{spp} G$ and $\langle X, Y\rangle \models_{spp} H$. By the

induction hypothesis, $Y \models G$ and $Y \models H$. Thus, $Y \models G \wedge H$ and, consequently, $Y \models F$. The case $F = G \vee H$ is similar. Finally, let $F = G \rightarrow H$. By the definition, if $\langle X, Y\rangle \models_{spp} F$ then $Y \models F$. Thus, the result follows.

(2) By the definition, $\langle X, Y\rangle \models_{spp} \neg F$ if and only if $Y \models \neg F$ and $Y \not\models F$. Thus, the claim follows.

(3) The argument is similar. The inductive step for the case $F = G \rightarrow H$ (the basis and all other cases are straightforward) is as follows. First, by the definition, if $\langle Y, Y\rangle \models_{spp} F$ then $Y \models F$. Conversely, if $Y \models F$, then $Y \not\models G$ or $Y \models H$. By the induction hypothesis, $Y \not\models G$ or $\langle Y, Y\rangle \models_{spp} H$. Since $Y \models F (= G \rightarrow H)$, $\langle Y, Y\rangle \models_{spp} F$ follows. □

The following result is analogous to similar results for $\models_{ht}$ and $\models_{flp}$, and ties the SPP-reduct and the relation $\models_{spp}$ (cf. Theorems 2 and 6).

Theorem 12. *For every theory $\mathcal{F}$ and for every two sets of atoms $X \subseteq Y$, $X \models \mathcal{F}^{\underline{Y}}_{=}$ if and only if $\langle X, Y\rangle \models_{spp} \mathcal{F}$.*

Proof. It is enough to prove the result for a single formula F. We proceed by induction. The case when $F = \bot$ is straightforward. Let $F = A$, where $A \in At$. If $X \models A^{\underline{Y}}_{=}$, then $A^{\underline{Y}}_{=} \neq \bot$. Thus, $A^{\underline{Y}}_{=} = A$. It follows that $X \models A$ and so, $\langle X, Y\rangle \models_{spp} A$. Conversely, if $\langle X, Y\rangle \models_{spp} A$, then $X \models A$. Since $X \subseteq Y$, $Y \models A$. Thus, $A^{\underline{Y}}_{=} = A$ and $X \models A^{\underline{Y}}_{=}$ as required.

Next, let $F = G \wedge H$. If $X \models (G \wedge H)^{\underline{Y}}_{=}$, then $(G \wedge H)^{\underline{Y}}_{=} = G^{\underline{Y}}_{=} \wedge H^{\underline{X}}_{=}$. Thus, $X \models G^{\underline{Y}}_{=}$ and $X \models H^{\underline{Y}}_{=}$. By the inductive hypothesis, $\langle X, Y\rangle \models_{spp} G$ and $\langle X, Y\rangle \models_{spp} H$. Thus, $\langle X, Y\rangle \models_{spp} G \wedge H$, as needed. Conversely, let $\langle X, Y\rangle \models_{spp} G \wedge H$. Then $\langle X, Y\rangle \models_{spp} G$ and $\langle X, Y\rangle \models_{spp} H$. By the inductive hypothesis, we have $X \models G^{\underline{Y}}_{=}$ and $X \models H^{\underline{Y}}_{=}$. Thus, $X \models G^{\underline{Y}}_{=} \wedge H^{\underline{Y}}_{=}$. Clearly, $G^{\underline{Y}}_{=} \not\equiv \bot$ and $H^{\underline{Y}}_{=} \not\equiv \bot$. By Proposition 9(2), $Y \models G$ and $Y \models H$. Thus, $Y \models G \wedge H$ and so, $(G \wedge H)^{\underline{Y}}_{=} = G^{\underline{Y}}_{=} \wedge H^{\underline{Y}}_{=}$. Hence, $X \models (F \wedge G)^{\underline{Y}}_{=}$.

The argument for the case $F = G \vee H$ is similar. And so, let us move on to the case $F = G \rightarrow H$. We have the following equivalences:

$$
\begin{aligned}
&X \models (G \rightarrow H)^{\underline{Y}}_{=},\\
&Y \not\models G; \quad \text{or} \quad Y \models G \quad \text{and} \quad Y \models H, \quad \text{and} \quad X \models H^{\underline{Y}}_{=},\\
&Y \not\models G; \quad \text{or} \quad Y \models H \quad \text{and} \quad X \models H^{\underline{Y}}_{=},\\
&Y \not\models G \quad \text{or} \quad Y \models H; \quad \text{and} \quad Y \not\models G \quad \text{or} \quad X \models H^{\underline{Y}}_{=},\\
&Y \models G \rightarrow H; \quad \text{and} \quad Y \not\models G \quad \text{or} \quad \langle X, Y\rangle \models_{spp} H,\\
&\langle X, Y\rangle \models_{spp} F.
\end{aligned}
$$

Thus, the result follows. □

The main consequence of Theorem 12 is a characterization of supported models in terms of the relation $\models_{spp}$.

Corollary 7. *Let $\mathcal{F}$ be a theory and Y a set of atoms. Then Y is a supported model of $\mathcal{F}$ if and only if $\langle Y, Y\rangle \models_{spp} \mathcal{F}$ and for every $X \subset Y$, $\langle X, Y\rangle \not\models_{spp} \mathcal{F}$.*

Proof. By the definition, Y is a supported model of $\mathcal{F}$ if and only if $Y \models F^{\underline{Y}}_{=}$, and for every $X \subset Y$, $X \not\models F^{\underline{Y}}_{=}$. We now apply Theorem 12. The former condition is equivalent to $\langle Y, Y\rangle \models_{spp} F$. The latter one is equivalent to the property that for every $X \subset Y$, $\langle X, Y\rangle \not\models_{spp} F$. Thus, the assertion follows. □

5.3. Strong equivalence with respect to supported models

We will now study the strong equivalence of theories with respect to the supported-model semantics. Two theories $\mathcal{F}$ and $\mathcal{G}$ are *strongly SPP-equivalent* if for every theory $\mathcal{H}$, $\mathcal{F} \cup \mathcal{H}$ and $\mathcal{G} \cup \mathcal{H}$ have the same supported models. Corollary 7 implies that if $\mathcal{F}$ and $\mathcal{G}$ have the same SPP-models then they are strongly SPP-equivalent. Unlike in the case of stable or FLP-stable semantics, though, that condition is not necessary. A weaker condition provides a characterization of strong SPP-equivalence. An SPP-model is *essential* if it is of the form $\langle Y, Y\rangle$ or $\langle Y \setminus \{A\}, Y\rangle$, where $A \in At$. Having the same essential SPP-models is sufficient and necessary for $\mathcal{F}$ and $\mathcal{G}$ to be strongly SPP-equivalent. To prove it, we need a simple auxiliary property.

Proposition 10. *For every formula F and for every interpretation Y, $F^{\underline{Y}}_{=}$ is monotone.*

Proof. Clearly, if $F = \bot$, $F^{\underline{Y}}_{=}$ is trivially monotone. If $F = A$, where $A \in At$ and $X \models F^{\underline{Y}}_{=}$, then $F^{\underline{Y}}_{=} = A$ and $A \in X$. Thus, for every Z, if $X \subseteq Z$, $Z \models F^{\underline{Y}}_{=}$.

Let us assume that $F = G \wedge H$ and that $X \models F^{\underline{Y}}_{=}$. Since $F^{\underline{Y}}_{=} = G^{\underline{Y}}_{=} \wedge H^{\underline{Y}}_{=}$ (we note that $F^{\underline{Y}}_{=} \neq \bot$), $X \models G^{\underline{Y}}_{=}$ and $X \models H^{\underline{Y}}_{=}$. Let Z be an interpretation such that $X \subseteq Z$. By the induction hypothesis, $Z \models G^{\underline{Y}}_{=}$ and $Z \models H^{\underline{Y}}_{=}$. Thus, $Z \models G^{\underline{Y}}_{=} \wedge H^{\underline{Y}}_{=}$ and so, $Z \models F^{\underline{Y}}_{=}$. The case of $F = G \vee H$ is similar.

Finally, let us assume that $F = G \to H$ and that $X \models F^{\underline{Y}}_{=}$. It follows that $F^{\underline{Y}}_{=} = \top$ or $F^{\underline{Y}}_{=} = H^{\underline{Y}}_{=}$. Clearly, if $F^{\underline{Y}}_{=} = \top$ then for every interpretation Z such that $X \subseteq Z$, $Z \models F^{\underline{Y}}_{=}$. If $F^{\underline{Y}}_{=} = H^{\underline{Y}}_{=}$, then $X \models H^{\underline{Y}}_{=}$ and, by the induction hypothesis, for every interpretation Z such that $X \subseteq Z$, $Z \models H^{\underline{Y}}_{=}$. Thus, in either case, if $X \subseteq Z$, then $Z \models F^{\underline{Y}}_{=}$. □

We now have the following characterization of strong SPP-equivalence. Unlike the one developed for programs [49], where the general case is established through a certain reduction to normal programs, the present characterization is direct.

Theorem 13. *Let $\mathcal{F}$ and $\mathcal{G}$ be two theories. Then, $\mathcal{F}$ and $\mathcal{G}$ are strongly SPP-equivalent if and only if $\mathcal{F}$ and $\mathcal{G}$ have the same essential SPP-models.*

Proof. ($\Rightarrow$) Let $\langle Y, Y\rangle$ be an essential SPP-model of $\mathcal{F}$. It follows that $Y \models \mathcal{F}^{\underline{Y}}_{=} \cup Y = (\mathcal{F} \cup Y)^{\underline{Y}}_{=}$. Moreover, Y is a minimal model of $\mathcal{F}^{\underline{Y}}_{=} \cup Y = (\mathcal{F} \cup Y)^{\underline{Y}}_{=}$. Consequently, Y is a supported model of $\mathcal{F} \cup Y$. By the assumption, Y is a supported model of $\mathcal{G} \cup Y$. Thus, Y is a model of $\mathcal{G}$ and so, by Proposition 9, $Y \models \mathcal{G}^{\underline{Y}}_{=}$. By Theorem 12, it follows that $\langle Y, Y\rangle$ is an SPP-model of $\mathcal{G}$.

Next, let $\langle Y \setminus \{a\}, Y\rangle$ be an essential SPP-model of $\mathcal{F}$. It follows that $Y \setminus \{a\} \models \mathcal{F}^{\underline{Y}}_{=}$. By Proposition 10, $Y \models \mathcal{F}^{\underline{Y}}_{=}$. Thus, $\langle Y, Y\rangle$ is an SPP-model of $\mathcal{F}$. By the argument given above, $\langle Y, Y\rangle$ is an SPP-model of $\mathcal{G}$. By Proposition 9, $Y \models \mathcal{G}^{\underline{Y}}_{=}$. Thus, $Y \models \mathcal{G}^{\underline{Y}}_{=} \cup (Y \setminus \{a\}) = (\mathcal{G} \cup (Y \setminus \{a\}))^{\underline{Y}}_{=}$. Let us assume that $\langle Y \setminus \{a\}, Y\rangle$ is not an SPP-model of $\mathcal{G}$. Then, $Y \setminus \{a\} \not\models \mathcal{G}^{\underline{Y}}_{=}$. Since every model of $(\mathcal{G} \cup (Y \setminus \{a\}))^{\underline{Y}}_{=} = \mathcal{G}^{\underline{Y}}_{=} \cup (Y \setminus \{a\})$ contains $Y \setminus \{a\}$, and $Y \setminus \{a\} \not\models \mathcal{G}^{\underline{Y}}_{=}$, it follows that Y is a minimal model of $(\mathcal{G} \cup (Y \setminus \{a\}))^{\underline{Y}}_{=}$. Thus, Y is a supported model of $\mathcal{G} \cup (Y \setminus \{a\})$. Consequently, it is a supported model of $\mathcal{F} \cup (Y \setminus \{a\})$. But $Y \setminus \{a\} \models (\mathcal{F} \cup (Y \setminus \{a\}))^{\underline{Y}}_{=}$, a contradiction. Thus, $\langle Y \setminus \{a\}, Y\rangle$ is an SPP-model of $\mathcal{G}$. By symmetry, it follows that essential SPP-models of $\mathcal{F}$ and $\mathcal{G}$ coincide.

($\Leftarrow$) Let $\mathcal{H}$ be any theory and let Y be a supported model of $\mathcal{F} \cup \mathcal{H}$. It follows that $\langle Y, Y\rangle$ is an SPP-model of $\mathcal{F}$ and of $\mathcal{H}$. By the assumption, $\langle Y, Y\rangle$ is an SPP-model of $\mathcal{G}$ and of $\mathcal{H}$. Thus, $\langle Y, Y\rangle \models_{spp} \mathcal{G} \cup \mathcal{H}$. Let $X \subset Y$ be such that $X \models (\mathcal{G} \cup \mathcal{H})^{\underline{Y}}_{=}$. It follows that $X \models \mathcal{G}^{\underline{Y}}_{=}$ and $X \models \mathcal{H}^{\underline{Y}}_{=}$. Let $a \in Y \setminus X$ (such an a exists). Then, by Proposition 10, $Y \setminus \{a\} \models \mathcal{G}^{\underline{Y}}_{=}$ and $Y \setminus \{a\} \models \mathcal{H}^{\underline{Y}}_{=}$. Thus, $\langle Y \setminus \{a\}, Y\rangle$ is an SPP-model of $\mathcal{G}$ and so, of $\mathcal{F}$. It follows that $Y \setminus \{a\} \models \mathcal{F}^{\underline{Y}}_{=}$ and, consequently, that $Y \setminus \{a\} \models \mathcal{F}^{\underline{Y}}_{=} \cup \mathcal{H}^{\underline{Y}}_{=} \equiv (\mathcal{F} \wedge \mathcal{H})^{\underline{Y}}_{=}$. This is a contradiction with Y being a supported model of $\mathcal{F} \cup \mathcal{H}$. Thus, Y is a minimal model of $(\mathcal{G} \cup \mathcal{H})^{\underline{Y}}_{=}$ and so, Y is a supported model of $\mathcal{G} \cup \mathcal{H}$. By symmetry, $\mathcal{F} \cup \mathcal{H}$ and $\mathcal{G} \cup \mathcal{H}$ have the same supported models. Thus, they are strongly SPP-equivalent. □

5.4. Complexity of reasoning with supported models

It turns out that, as in the case of disjunctive logic programs, reasoning with supported models is easier (assuming the polynomial hierarchy does not collapse) than reasoning with stable or FLP-stable models. Namely, we have the following result.

Theorem 14. *The problem of the existence of a supported model is NP-complete. Skeptical reasoning with supported models is coNP-complete. Brave reasoning with supported models is NP-complete.*

Proof. Given a theory $\mathcal{F}$ and a set of atoms Y, Y is a minimal model of $F^{\underline{Y}}_{=}$ if and only if $Y \models F^{\underline{Y}}_{=}$ and for every $a \in Y$, $Y \setminus \{a\} \not\models F^{\underline{Y}}_{=}$ (by the monotonicity of $F^{\underline{Y}}_{=}$, cf. Proposition 10). Since $F^{\underline{Y}}_{=}$ can be computed in polynomial time, it follows that the problem of the existence of a supported model of a theory $\mathcal{F}$ is in the class NP. The membership of the other two problems in their respective classes can also be established as a consequence of the observation that the problem of checking whether a set of atoms Y is a supported model of a theory $\mathcal{F}$ is in the class P. The hardness part follows in each case from the well-known hardness of the corresponding problems under the restriction to normal programs. For the sake of completeness, we will outline here a proof that the problem of the existence of a supported model of a normal program is NP-hard. To this end, we note that by the Fages Lemma [18], for normal programs without positive atoms in the bodies of rules, stable and supported models coincide. The problem of the existence of a stable model for such programs is NP-complete (the construction used by Marek and Truszczyński [39], demonstrates that). Thus, the problem of the existence of a supported model for such programs is NP-complete, too and, consequently, the NP-hardness of the problem for arbitrary normal programs follows. □

5.5. Normal form result for the supported-model semantics

As in the other two cases, also for the supported-model semantics every theory has a normal form. However, now it is given by even simpler formulas — conjunctions of normal rules. We start by stating several simple properties concerning the equivalence of formulas with respect to SPP-models. We say that two formulas F and G are *SPP-equivalent*, denoted $F \equiv_{spp} G$, if they have the same SPP-models.

Proposition 11. *For every formulas F, G and H:*

(1) $F \to G \equiv_{spp} \neg F \vee G$;
(2) $\neg(F \vee G) \equiv_{spp} \neg F \wedge \neg G$;
(3) $\neg(F \wedge G) \equiv_{spp} \neg F \vee \neg G$;
(4) $F \wedge (G \vee H) \equiv_{spp} (F \wedge G) \vee (F \wedge H)$;
(5) $F \vee (G \wedge H) \equiv_{spp} (F \vee G) \wedge (F \vee H)$;
(6) $\neg\neg\neg F \equiv_{spp} \neg F$;
(7) $\neg\top \equiv_{spp} \bot$ *and* $\neg\bot \equiv_{spp} \top$;
(8) $F \wedge \bot \equiv_{spp} \bot$ *and* $F \vee \bot \equiv_{spp} F$;
(9) $F \wedge \top \equiv_{spp} F$ *and* $F \vee \top \equiv_{spp} \top$;
(10) $F \circ F \equiv_{spp} F$, *for* $\circ = \wedge$ *and* $\vee$.

Proof. (1) If $\langle X, Y\rangle \models_{spp} F \to G$ then $Y \not\models F$ or $\langle X, Y\rangle \models_{spp} G$. We recall that $\langle X, Y\rangle \models_{spp} \neg F$ if and only if $Y \not\models F$ (Theorem 11(2)). Thus, $\langle X, Y\rangle \models_{spp} \neg F$ or $\langle X, Y\rangle \models_{spp} G$ and so, $\langle X, Y\rangle \models_{spp} \neg F \vee G$. Conversely, if $\langle X, Y\rangle \models_{spp} \neg F \vee G$ then $Y \models \neg F \vee G$ (Theorem 11(1)) and, consequently, $Y \models F \to G$. Moreover, $\langle X, Y\rangle \models_{spp} \neg F$ or $\langle X, Y\rangle \models_{spp} G$. Thus, $Y \not\models F$ or $\langle X, Y\rangle \models_{spp} G$ (again by Theorem 11(2)), and the claim follows.

The properties (2) and (3) follow by Theorem 11(2) and the corresponding properties of equivalence in propositional logic. For instance, $\langle X, Y\rangle \models_{spp} \neg(F \vee G)$ if and only if $Y \not\models F \vee G$ (Theorem 11(2)). The latter condition is equivalent to $Y \not\models F$ and $Y \not\models G$. This conjunction, in turn, is equivalent to $\langle X, Y\rangle \models_{spp} \neg F$ and $\langle X, Y\rangle \models_{spp} \neg G$ (Theorem 11(2), again). Thus, $\langle X, Y\rangle \models_{spp} \neg(F \vee G)$ if and only if $\langle X, Y\rangle \models_{spp} \neg F \wedge \neg G$, which implies (2).

The properties (4) and (5) follow from the inductive definition of $\models_{spp}$ for the connectives $\wedge$ and $\vee$. For instance, $\langle X, Y\rangle \models_{spp} F \wedge (G \vee H)$ if and only if $\langle X, Y\rangle \models_{spp} F$ and $\langle X, Y\rangle \models_{spp} G \vee H$. The latter condition can be equivalently restated as $\langle X, Y\rangle \models_{spp} F$, and $\langle X, Y\rangle \models_{spp} G$ or $\langle X, Y\rangle \models_{spp} H$, or as $\langle X, Y\rangle \models_{spp} F$ and $\langle X, Y\rangle \models_{spp} G$, or $\langle X, Y\rangle \models_{spp} F$ and $\langle X, Y\rangle \models_{spp} H$. Using the inductive definition of $\models_{spp}$ for the connectives $\wedge$ and $\vee$, we get that $\langle X, Y\rangle \models_{spp} F \wedge (G \vee H)$ if and only if $\langle X, Y\rangle \models_{spp} (F \wedge G) \vee (F \wedge H)$, thus proving (4).

The property (6) follows from Theorem 11(2), and the property (7) from the definitions of the shorthands $\top$ and $\neg$, and from the definition of $\models_{spp}$. The remaining three properties follow from the inductive definition of $\models_{spp}$ for the connectives $\wedge$ and $\vee$ and the facts that $\bot$ has no SPP-models and $\top$ is satisfied by every SPP-model. □

The normal form result is now a consequence of Proposition 11 and Theorem 13.

Theorem 15. *Let $\mathcal{F}$ be a theory. Then there is a program $\mathcal{G}$ consisting of normal rules such that $\mathcal{F}$ and $\mathcal{G}$ have the same essential SPP-models* (*and so, are strongly SPP-equivalent and have the same supported models*).

Proof. It is enough to prove that for every formula F there is a program consisting of normal rules that has the same essential SPP-models as F. Let us consider the following transformation of F. First, we replace in F every subformula $G \to H$ with $\neg G \vee H$ (Proposition 11(1)). Then, we proceed as in the case of propositional logic when constructing a CNF representation of the formula, using De Morgan Laws (Proposition 11(2) and (3)) to move negation in, then using the "triple negation" law (Proposition 11(4)), distributivity properties (Proposition 11(5) and (6)), and simplification rules (Proposition 11(7)–(10)). When that process ends, we split the resulting conjunction into the set of its conjuncts — disjunctions that are of the form

$$C_1 \vee \cdots \vee C_k \vee \neg A_1 \vee \cdots \vee \neg A_m \vee \neg\neg B_1 \vee \cdots \vee \neg\neg B_n, \tag{3}$$

where A_i, B_i, and C_i are atoms. By Proposition 11, the set of these formulas has the same SPP-models as the original formula F. Thus, it is strongly SPP-equivalent to F.

Next, we note that if $k \geqslant 2$, then formulas

$$C_1 \vee \cdots \vee C_k \vee G$$

and

$$\bigwedge_{i=1}^{k} (C_i \vee \neg\neg C_1 \vee \cdots \vee \neg\neg C_{i-1} \vee \neg\neg C_{i+1} \vee \cdots \vee \neg\neg C_k \vee G),$$

where C_i are atoms and G is a formula, have the same essential models and, consequently, they are strongly SPP-equivalent. Thus, each disjunction (3) can be rewritten into a strongly SPP-equivalent set of disjunctions:

$$C_i \vee \neg\neg C_1 \vee \cdots \vee \neg\neg C_{i-1} \vee \neg\neg C_{i+1} \vee \cdots \vee \neg\neg C_k \vee \neg A_1 \vee \cdots \vee \neg A_m \vee \neg\neg B_1 \vee \cdots \vee \neg\neg B_n,$$

where $1 \leqslant i \leqslant k$.

By Proposition 11(3) and (1), each such disjunction can be written as an SPP-equivalent normal rule

$$A_1 \wedge \cdots \wedge A_m \wedge \neg B_1 \wedge \cdots \wedge \neg B_n \wedge \neg C_1 \wedge \cdots \wedge \neg C_{i-1} \wedge \neg C_{i+1} \wedge \cdots \wedge \neg C_k \rightarrow C_i,$$

$1 \leqslant i \leqslant k$. Thus, the assertion follows. □

5.6. Relationships

Finally, we will study the relationship between the semantics given by relations $\models_{ht}$ and $\models_{flp}$, on the one hand, and $\models_{spp}$, on the other. To this end, we observe first that SPP-models are HT-models and FLP-models.

Proposition 12. *For every formula F and every $X \subseteq Y \subseteq At$:*

(1) $\langle X, Y\rangle \models_{spp} F$ *implies* $\langle X, Y\rangle \models_{ht} F$;
(2) $\langle X, Y\rangle \models_{spp} F$ *implies* $\langle X, Y\rangle \models_{flp} F$.

Proof. We proceed by induction. Both for (1) and (2), the base case and the inductive step for $F = G \circ H$, where $\circ = \vee$ and $\wedge$, follow from the observation that the corresponding conditions defining the relations $\models_{spp}$, $\models_{ht}$ and $\models_{flp}$ are the same.

Let us consider $F = G \rightarrow H$. Let us assume that $\langle X, Y\rangle \models_{spp} F$. It follows that

$$Y \models G \rightarrow H; \quad \text{and} \quad Y \not\models G \quad \text{or} \quad \langle X, Y\rangle \models_{spp} H. \tag{4}$$

We know that $Y \not\models G$ implies $\langle X, Y\rangle \not\models_{ht} G$ [21]. Combining that with the induction hypothesis, we obtain from (4) that $Y \models G \rightarrow H$; and $\langle X, Y\rangle \not\models_{ht} G$ or $\langle X, Y\rangle \models_{ht} H$. Thus, $\langle X, Y\rangle \models_{ht} G \rightarrow H$ follows.

It also follows from (4) that $Y \models G \rightarrow H$; and $Y \not\models G$ or $\langle X, Y\rangle \models_{spp} H$ or $X \not\models G$ (we have added one more disjunct to the second conjunct). By the induction hypothesis, that new disjunction implies $Y \not\models G$ or $\langle X, Y\rangle \models_{flp} H$ or $X \not\models G$. Consequently, $\langle X, Y\rangle \models_{flp} G \rightarrow H$ follows. □

As a corollary, we now obtain that, as in the case of disjunctive logic programming, stable models (both under the standard and FLP generalizations) are supported models.

Theorem 16. *For every theory $\mathcal{F}$ and every set of atoms Y, if Y is a stable model of $\mathcal{F}$ or Y is an FLP-stable model of $\mathcal{F}$, then Y is a supported model of $\mathcal{F}$.*

Proof. The proof is the same in both cases. Thus, we show the argument in the first case only. Let Y be a stable model of $\mathcal{F}$. It follows that $\langle Y, Y\rangle \models_{ht} \mathcal{F}$. Consequently, $Y \models \mathcal{F}$ (cf. Theorem 1) and so, $\langle Y, Y\rangle \models_{spp} \mathcal{F}$ (by Theorem 11). Let us assume that $X \subseteq Y$ and $\langle X, Y\rangle \models_{spp} \mathcal{F}$. Then $\langle X, Y\rangle \models_{ht} \mathcal{F}$ (Proposition 12). Since Y is a stable model of $\mathcal{F}$, $X = Y$. Thus, Y is a supported model of $\mathcal{F}$. □

In general, the implications in Proposition 12 cannot be reversed. However, in the case of the relation $\models_{ht}$, we can find a broad class of formulas for which the converse implication holds, too. The key is the following result that exhibits a class of formulas, for which the relation $\models_{ht}$ reduces to the standard propositional entailment. Before we state it, we recall that an occurrence of an atom A in a formula is directly under the scope of $\neg$, if it is the antecedent in the subformula $A \rightarrow \bot$.

Proposition 13. *Let F be a formula such that every occurrence of an atom in F is directly under the scope of $\neg$. Then, for every $X \subseteq Y \subseteq At$, $\langle X, Y\rangle \models_{ht} F$ if and only if $Y \models F$.*

Proof. The claim is evident if $F = \bot$. Otherwise, F is of the form $F = G \circ H$, where $\circ = \vee, \wedge$ or $\rightarrow$. First, let us assume that $\circ = \rightarrow$, G is an atom and $H = \bot$. Then, the claim follows from Theorem 1(2). Otherwise, the induction hypothesis applies to G and H (every occurrence of an atom in G and H is directly under the scope of $\neg$) and again, the inductive step is easy to verify. □

This result has several consequences. First, we show that we can express the relation $\models_{spp}$ in terms of the relation $\models_{ht}$. Moreover, it does not require any extension of the language. Given a formula F, we define $\overline{F}$ to be the formula obtained by replacing every nonhead occurrence of an atom A that is not directly negated with $\neg\neg A$.

Proposition 14. *For every formula F and every $X \subseteq Y \subseteq At$, $\langle X, Y\rangle \models_{spp} F$ if and only if $\langle X, Y\rangle \models_{ht} \overline{F}$.*

Proof. The claim is evident if $F = \bot$. If $F = A$, where $A \in At$, then $\overline{F} = A = F$ and the result holds.

Since $\overline{G \circ H} = \overline{G} \circ \overline{H}$, for $\circ = \wedge$ and $\vee$, for these two connectives, the induction step is easy to verify. Thus, let $F = G \to H$. Clearly, $\overline{F} = G' \to \overline{H}$, where G' is obtained from G by replacing every occurrence of an atom A that is not directly negated with $\neg\neg A$. Proposition 13 applies to G'. Thus, $\langle X, Y\rangle \models_{ht} G'$ if and only if $Y \models G'$. By the definitions of G' and of the classical entailment relation $\models$, $Y \models G'$ if and only if $Y \models G$. Thus, $\langle X, Y\rangle \models_{ht} G'$ if and only if $Y \models G$. Finally, by the induction hypothesis, $\langle X, Y\rangle \models_{ht} \overline{H}$ if and only if $\langle X, Y\rangle \models_{spp} H$. Consequently, the following statements are equivalent:

(1) $\langle X, Y\rangle \models_{spp} G \to H$;
(2) $Y \models G \to H$; and $Y \not\models G$ or $\langle X, Y\rangle \models_{spp} H$;
(3) $Y \models G' \to H$; and $\langle X, Y\rangle \not\models_{ht} G'$ or $\langle X, Y\rangle \models_{ht} \overline{H}$;
(4) $\langle X, Y\rangle \models_{ht} G' \to \overline{H}$;
(5) $\langle X, Y\rangle \models_{ht} \overline{F}$.

Thus, the claim for $F = G \to H$ follows. □

Corollary 8. *For every formula F and every interpretation Y, Y is a supported model of F if and only if it is a stable model of $\overline{F}$.*

Next, we observe that it is impossible to express $\models_{ht}$ in terms of $\models_{spp}$. The following corollary makes the meaning of that statement precise.

Corollary 9. *Let A and B be atoms. There is no formula F such that $\langle X, Y\rangle \models_{ht} A \to B \vee \neg B \vee \neg A$ if and only if $\langle X, Y\rangle \models_{spp} F$.*

Proof. Let $H = A \to B \vee \neg B \vee \neg A$. It is easy to verify that $\langle \emptyset, \{A, B\}\rangle \models_{ht} H$ and $\langle \{A\}, \{A, B\}\rangle \not\models_{ht} H$. Let us assume that there is a formula F such that for every $X \subseteq Y$, $\langle X, Y\rangle \models_{ht} H$ if and only if $\langle X, Y\rangle \models_{spp} F$. Then, $\langle \emptyset, \{A, B\}\rangle \models_{ht} F$. By Theorem 12, $\emptyset \models F^{\underline{\{A,B\}}}$. By Proposition 10, $F^{\underline{Y}}$ is monotone. Thus, $\{A\} \models F^{\underline{\{A,B\}}}$ or, equivalently, $\langle \{A\}, \{A, B\}\rangle \models_{ht} F$, a contradiction. □

Given our result on the biexpressibility of stable and FLP-stable semantics, it follows that for every theory $\mathcal{F}$ there is a theory $\mathcal{F}'$ such that Y is a supported model of $\mathcal{F}$ if and only if Y is an FLP-stable model of $\mathcal{F}'$. In other words, we can express supported models in terms of FLP-stable models without the need to expand the language. On the other hand, there is a theory $\mathcal{F}$ such that for no theory $\mathcal{F}'$, FLP-stable models of $\mathcal{F}$ are exactly supported models for $\mathcal{F}'$. Thus, FLP-stable models cannot be (in general) expressed as supported models, unless we extend the language.

As a corollary to Proposition 14 we obtain the promised result showing a class of formulas, which have the same HT- and SPP-models.

Proposition 15. *Let F be a formula such that every occurrence of an atom in F is either a head occurrence or falls directly under the scope of $\neg$. For every $X \subseteq Y \subseteq At$, $\langle X, Y\rangle \models_{ht} F$ if and only if $\langle X, Y\rangle \models_{spp} F$.*

Proof. Since under the assumptions about F, we have $F = \overline{F}$, the claim follows from Proposition 14. □

In turn, this result implies the following generalization of the property that stable and supported models of purely negative disjunctive programs coincide.

Corollary 10. *Let $\mathcal{F}$ be a theory such that every occurrence of an atom in $\mathcal{F}$ is either a head occurrence or falls directly under the scope of $\neg$. Then Y is a stable model of $\mathcal{F}$ if and only if it is a supported model of $\mathcal{F}$.*

There is an alternative argument that shows that the $\models_{spp}$ relation can be expressed by means of the $\models_{ht}$ and $\models_{flp}$ relations.[3] It is based on the following simple observation that identifies a class of formulas on which *all* three semantics coincide.

Proposition 16. *Let F be a formula in which every occurrence of $\to$ has $\bot$ in the consequent (in other words, formulas that can be written by means of the connectives $\wedge$, $\vee$ and $\neg$). For every HT-interpretation $\langle X, Y\rangle$, the following conditions are equivalent*:

(1) $\langle X, Y\rangle \models_{ht} F$;
(2) $\langle X, Y\rangle \models_{flp} F$;
(3) $\langle X, Y\rangle \models_{spp} F$.

[3] This argument is due to one of the anonymous reviewers.

Proof. The clauses defining each of the relations in the case when $F = \bot$, $F = A$, where A is an atom, $F = G \wedge H$ and $F = G \vee H$ are the same. Moreover, for each of the three satisfiability relations, $\langle X, Y\rangle$ satisfies (in the corresponding sense) $\neg G$ if and only if $Y \not\models G$ (and so, for such formulas, the three satisfiability relations also coincide). Thus, the result follows by induction. □

For every formula F, we denote by $\overline{\overline{F}}$ the formula obtained from F by replacing each subformula $G \to H$, where $H \neq \bot$, with $(G \to \bot) \vee H$ (or $\neg G \vee H$, for short).

Corollary 11. *For every formula F, SPP-models of F coincide with HT-models of $\overline{\overline{F}}$, with FLP-models of $\overline{\overline{F}}$ and with SPP-models of $\overline{\overline{F}}$.*

Proof. By Proposition 11(1), F and $\overline{\overline{F}}$ have the same SPP-models. Thus, the assertion follows from Proposition 16. □

The value of our first approach to show that the SPP semantics can be expressed in terms of HT and FLP semantics, which uses $F \mapsto \overline{F}$ transformation, is in that it gives Corollary 10, a generalization of the property that in purely negative disjunctive programs stable and supported models coincide. On the other hand, the value of the approach based on the $F \mapsto \overline{\overline{F}}$ transformation lies in the fact that it shows a class of formulas (theories), on which all three semantics coincide.

6. Discussion and conclusions

In this paper, we developed generalizations of the FLP semantics and the supported-model semantics to the full language of propositional logics using appropriate variants of methods developed earlier by Pearce and Ferraris for the stable-model semantics. In this way, all three semantics are cast in the same uniform framework, which facilitates comparisons and offers insights into their properties. Our results contribute to the theoretical foundations of answer-set programming, a major knowledge representation formalism.

More specifically, Ferraris [19] showed that the stable-model semantics can be extended to the language of propositional logic by means of an appropriate generalization of the notion of the F-reduct. We showed that the approach by Ferraris can be adapted to two other semantics of programs: the FLP and supported-model semantics. Moreover, the generalizations require only small changes in the definition of the reduct that concern how the implication is handled in the case both its antecedent and consequent are satisfied by the context. In the case of the FLP-reduct, we keep the antecedent of the implication unchanged, in the case of the SPP-reduct, we drop it.

Not only the definitions follow the same pattern. The theories of the three semantics are quite similar, too, both in the way the results are stated as well as proved. In particular, in each case, we have a corresponding characterization of the semantics in terms of a satisfiability relation between HT-interpretations and formulas similar to the characterization of the stable-model semantics of arbitrary propositional theory by Pearce [44]. As before, what differentiates between the relations is the way the implication is handled.

The uniformity with which the three semantics can be defined and studied is striking. It suggests that considering the reduct-based approach in the general language of logic, may reveal new insights into the phenomenon of nonmonotonicity. A related question is whether any other semantics can be defined in this way, that is, whether there are any other notions of reduct that might lead to useful formalisms. As there seem to be no simple ways to modify the reduct left, the uniform approach presented here suggests that the realm of nonmonotonic semantics of programs and theories may essentially boil down to the three ones discussed in the paper.

The uniformity notwithstanding, there are also differences. We saw that the relation $\models_{spp}$ is, in some sense, weaker than the other two. Further, the relation $\models_{ht}$ that captures the stable-models semantics defines a logic, namely the logic HT. To the contrary, the relation $\models_{flp}$ does not: the set of formulas F such that for every $\langle X, Y\rangle$, $\langle X, Y\rangle \models_{flp} F$ while closed under modus ponens, is not closed under substitution.[4] Also, while stable and FLP semantics are closely related, supported-model semantics is essentially different (cf. the characterization of strong SPP-equivalence, and the normal form theorem). A detailed comparison of the semantics is beyond the scope of this paper. We leave it for future work.

We note here that our argument for the normal form result with respect to SPP-models can be adjusted to show that the set of theorems with respect to $\models_{spp}$ is closed under substitution and so, is a logic. The question of axiomatization of that logic is for now open.

Acknowledgements

The present text reflects many corrections and suggestions offered by the anonymous reviewers. The author gratefully acknowledges their effort. The work was partially supported by the NSF grant IIS-0913459.

[4] Indeed, for every HT-interpretation we have $\langle X, Y\rangle$, $\langle X, Y\rangle \models_{flp} p \to p$, yet $\langle\{p\}, \{p, q\}\rangle \not\models_{flp} (p \to q) \to (p \to q)$. This behavior of the FLP semantics is a drawback. In particular, it is responsible for the difficulty of using explicit definitions under the FLP semantics, which we discussed at the end of Section 3.

References

[1] K. Apt, Logic programming, in: J. van Leeuven (Ed.), Handbook of Theoretical Computer Science, Elsevier, Amsterdam, 1990, pp. 493–574.
[2] M. Balduccini, M. Gelfond, M. Nogueira, Answer set based design of knowledge systems, Annals of Mathematics and Artificial Intelligence 47 (1–2) (2006) 183–219.
[3] C. Baral, From knowledge to intelligence – building blocks and applications, invited talk, AAAI 2005 (2005), http://www.public.asu.edu/~cbaral/aaai05-invited-talk.ppt.
[4] C. Baral, K. Chancellor, N. Tran, N.L. Tran, A. Joy, M. Berens, A knowledge based approach for representing and reasoning about signaling networks, in: Proceedings of the 12th International Conference on Intelligent Systems for Molecular Biology/3rd European Conference on Computational Biology (Supplement of Bioinformatics), 2004, pp. 15–22.
[5] G. Boenn, M. Brain, M.D. Vos, J. Fitch, Anton: Composing logic and logic composing, in: E. Erdem, F. Lin, T. Schaub (Eds.), Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2009, in: Lecture Notes in Computer Science, vol. 5753, Springer, 2009, pp. 542–547.
[6] S. Brass, J. Dix, Characterizations of the disjunctive stable semantics by partial evaluation, Journal of Logic Programming 32 (3) (1997) 207–228.
[7] D. Brooks, E. Erdem, J. Minett, D. Ringe, Character-based cladistics and answer set programming, in: M. Hermenegildo, D. Cabeza (Eds.), Proceedings of the 7th International Symposium on Practical Aspects of Declarative Languages, PADL 2005, in: Lecture Notes in Computer Science, vol. 3350, Springer, 2005, pp. 37–51.
[8] P. Cabalar, P. Ferraris, Propositional theories are strongly equivalent to logic programs, Theory and Practice of Logic Programming 7 (6) (2007) 745–759.
[9] K. Clark, Negation as failure, in: H. Gallaire, J. Minker (Eds.), Logic and Data Bases, Plenum Press, New York–London, 1978, pp. 293–322.
[10] M. Denecker, J. Vennekens, S. Bond, M. Gebser, M. Truszczyński, The second answer set programming competition, in: E. Erdem, F. Lin, T. Schaub (Eds.), Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2009, in: Lecture Notes in Computer Science, vol. 5753, Springer, 2009, pp. 637–654.
[11] T. Eiter, G. Brewka, M. Dao-Tran, M. Fink, G. Ianni, T. Krennwallner, Combining nonmonotonic knowledge bases with external sources, in: S. Ghilardi, R. Sebastiani (Eds.), Proceedings of the 7th International Symposium on Frontiers of Combining Systems, FroCoS 2009, in: Lecture Notes in Computer Science, vol. 5749, Springer, 2009, pp. 18–42.
[12] T. Eiter, W. Faber, N. Leone, G. Pfeifer, A. Polleres, Answer set planning under action costs, Journal of Artificial Intelligence Research (JAIR) 19 (2003) 25–71.
[13] T. Eiter, W. Faber, N. Leone, G. Pfeifer, A. Polleres, A logic programming approach to knowledge-state planning, II: The dlv^k system, Artificial Intelligence 144 (1–2) (2003) 157–211.
[14] T. Eiter, M. Fink, S. Woltran, Semantical characterizations and complexity of equivalences in answer set programming, ACM Transactions on Computational Logic 8 (3) (July 2007) 53.
[15] T. Eiter, G. Gottlob, On the computational cost of disjunctive logic programming: Propositional case, Annals of Mathematics and Artificial Intelligence 15 (3–4) (1995) 289–323.
[16] W. Faber, N. Leone, G. Pfeifer, Recursive aggregates in disjunctive logic programs: semantics and complexity, in: Proceedings of the 9th European Conference on Artificial Intelligence, JELIA 2004, in: Lecture Notes in Artificial Intelligence, vol. 3229, Springer, 2004, pp. 200–212.
[17] W. Faber, G. Pfeifer, N. Leone, T. Dell'Armi, G. Ielpa, Design and implementation of aggregate functions in the dlv system, Theory and Practice of Logic Programming 8 (5–6) (2008) 545–580.
[18] F. Fages, Consistency of Clark's completion and existence of stable models, Journal of Methods of Logic in Computer Science 1 (1994) 51–60.
[19] P. Ferraris, Answer sets for propositional theories, in: Logic Programming and Nonmonotonic Reasoning, 8th International Conference, LPNMR 2005, in: Lecture Notes in Artificial Intelligence, vol. 3662, Springer, 2005, pp. 119–131.
[20] P. Ferraris, V. Lifschitz, Weight constraints and nested expressions, Theory and Practice of Logic Programming 5 (2004) 45–74.
[21] P. Ferraris, V. Lifschitz, Mathematical foundations of answer set programming, in: S. Artëmov, H. Barringer, A. d'Avila Garcez, L.C. Lamb, J. Woods (Eds.), We Will Show Them! Essays in Honour of Dov Gabbay, College Publications, 2005, pp. 615–664.
[22] M. Gelfond, N. Leone, Logic programming and knowledge representation – the A-prolog perspective, Artificial Intelligence 138 (2002) 3–38.
[23] M. Gelfond, V. Lifschitz, The stable semantics for logic programs, in: Proceedings of the 5th International Conference on Logic Programming, ICLP 1988, MIT Press, 1988, pp. 1070–1080.
[24] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Generation Computing 9 (1991) 365–385.
[25] E. Giunchiglia, J. Lee, V. Lifschitz, N. McCain, H. Turner, Nonmonotonic causal theories, Artificial Intelligence 153 (1–2) (2004) 49–104.
[26] A. Heyting, Die formalen Regeln der intuitionistischen Logik, Sitzungsberichte der Preussischen Akademie von Wissenschaften, Physikalisch-mathematische Klasse, 1930, pp. 42–56.
[27] K. Inoue, C. Sakama, Negation as failure in the head, Journal of Logic Programming 35 (1998) 39–78.
[28] J. Lee, V. Lifschitz, R. Palla, A reductive semantics for counting and choice in answer set programming, in: Proceedings of the 23rd AAAI Conference on Artificial Intelligence, AAAI 2008, AAAI Press, 2008, pp. 472–479.
[29] J. Lee, Y. Meng, On reductive semantics of aggregates in answer set programming, in: Proceedings of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR 2009, in: Lecture Notes in Computer Science, vol. 5753, Springer, 2009, pp. 182–195.
[30] Y. Lierler, Abstract answer set solvers, in: M.G. de la Banda, E. Pontelli (Eds.), Proceedings of the 24th International Conference on Logic Programming, ICLP 2008, in: Lecture Notes in Computer Science, vol. 5366, Springer, 2008, pp. 377–391.
[31] Y. Lierler, M. Maratea, Cmodels-2: SAT-based answer set solver enhanced to non-tight programs, in: Proceedings of LPNMR-04, in: Lecture Notes in Computer Science, vol. 2923, Springer, 2004, pp. 346–350.
[32] V. Lifschitz, D. Pearce, A. Valverde, Strongly equivalent logic programs, ACM Transactions on Computational Logic 2 (4) (2001) 526–541.
[33] V. Lifschitz, L.R. Tang, H. Turner, Nested expressions in logic programs, Annals of Mathematics and Artificial Intelligence (1999) 369–389.
[34] V. Lifschitz, T. Woo, Answer sets in general nonmonotonic reasoning, in: Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning, KR '92, Morgan Kaufmann, San Mateo, CA, 1992, pp. 603–614.
[35] F. Lin, Reducing strong equivalence of logic programs to entailment in classical propositional logic, in: Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning, KR 2002, Morgan Kaufmann, 2002, pp. 170–176.
[36] F. Lin, Y. Zhao, ASSAT: Computing answer sets of a logic program by SAT solvers, in: Proceedings of the 18th National Conference on Artificial Intelligence, AAAI 2002, AAAI Press, 2002, pp. 112–117.
[37] V. Marek, M. Truszczyński, Stable models and an alternative logic programming paradigm, in: K. Apt, W. Marek, M. Truszczyński, D. Warren (Eds.), The Logic Programming Paradigm: A 25-Year Perspective, Springer, Berlin, 1999, pp. 375–398.
[38] W. Marek, V. Subrahmanian, The relationship between stable, supported, default and autoepistemic semantics for general logic programs, Theoretical Computer Science 103 (2) (1992) 365–386.
[39] W. Marek, M. Truszczyński, Autoepistemic logic, Journal of the ACM 38 (3) (1991) 588–619.
[40] R. Moore, Semantical considerations on nonmonotonic logic, Artificial Intelligence 25 (1) (1985) 75–94.
[41] I. Niemelä, Logic programming with stable model semantics as a constraint programming paradigm, Annals of Mathematics and Artificial Intelligence 25 (3–4) (1999) 241–273.

[42] I. Niemelä, P. Simons, Extending the smodels system with cardinality and weight constraints, in: J. Minker (Ed.), Logic-Based Artificial Intelligence, Kluwer Academic Publishers, 2000, pp. 491–521.
[43] M. Nogueira, M. Balduccini, M. Gelfond, R. Watson, M. Barry, An a-prolog decision support system for the space shuttle, in: I.V. Ramakrishnan (Ed.), Proceedings of the 3rd International Symposium on Practical Aspects of Declarative Languages, PADL 2001, in: Lecture Notes in Computer Science, vol. 1990, Springer, 2001, pp. 169–183.
[44] D. Pearce, A new logical characterisation of stable models and answer sets, in: J. Dix, L.M. Pereira, T.C. Przymusinski (Eds.), Non-Monotonic Extensions of Logic Programming, NMELP '96, in: Lecture Notes in Computer Science, vol. 1216, Springer, 1997, pp. 57–70.
[45] D. Pearce, A. Valverde, Quantified equilibrium logic and foundations for answer set programs, in: Proceedings of the 24th International Conference on Logic Programming, ICLP 2008, 2008, pp. 546–560.
[46] T. Schaub, S. Thiele, Metabolic network expansion with answer set programming, in: P. Hill, D. Warren (Eds.), Proceedings of the 25th International Conference on Logic Programming, ICLP 2009, in: Lecture Notes in Computer Science, vol. 5649, Springer, 2009, pp. 312–326.
[47] T. Soininen, I. Niemelä, Developing a declarative rule language for applications in product configuration, in: G. Gupta (Ed.), Proceedings of the 1st International Workshop on Practical Aspects of Declarative Languages, PADL 1999, in: Lecture Notes in Computer Science, vol. 1551, Springer, 1998, pp. 305–319.
[48] T. Son, E. Pontelli, C. Sakama, Logic programming for multiagent planning with negotiation, in: P. Hill, D. Warren (Eds.), Proceedings of the 25th International Conference on Logic Programming, ICLP 2009, in: Lecture Notes in Computer Science, vol. 5649, Springer, 2009, pp. 99–114.
[49] M. Truszczyński, S. Woltran, Hyperequivalence of logic programs with respect to supported models, in: Proceedings of the 23rd National Conference on Artificial Intelligence, AAAI 2008, AAAI Press, 2008, pp. 560–565.
[50] P. Tu, T. Son, C. Baral, Reasoning and planning with sensing actions, incomplete information, and static causal laws using answer set programming, Theory and Practice of Logic Programming 7 (4) (2007) 377–450.
[51] H. Turner, Strong equivalence made easy: Nested expressions and weight constraints, Theory and Practice of Logic Programming 3 (2003) 609–622.
[52] S. Woltran, A common view on strong, uniform, and other notions of equivalence in answer-set programming, Theory and Practice of Logic Programming 8 (2) (2008) 217–234.