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

Mirosław Truszczyński

Department of Computer Science, University of Kentucky, Lexington, KY 40506-0046, USA
mirek@cs.uky.edu

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 [1], is the foundation of answer-set programming [2–4], a paradigm for modeling and solving search problems. From its inception, developing theoretical underpinnings of the stable-model semantics has been a major research objective. In particular, a contribution by Pearce [5] explained the stable-model semantics in terms of models of theories in the logic *here-and-there* (*HT*, for short), introduced by Heyting [6].

Pearce's work 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 in the language of propositional logic. Second, it brought about the notion of *strong equivalence* of programs, fundamental to modular program development [7].

The original definition of stable models [1] 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. Lifschitz and Ferraris [8, 9] 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.

The question motivating the present work is 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, Leone and Pfeifer [10] on programs with aggregates. Aggregates, in the form of weight aggregates, were introduced to answer-set programming by Niemelä and Simons [11], who extended the stable-model semantics to that class of programs. Ferraris and Lifschitz [9] cast that generalization in terms of stable models of propositional theories. Stable models of programs with weight constraints are no longer guaranteed to be minimal models. From the perspective of the Ferraris and Lifschitz's result, it is not unexpected. 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. [10] 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 this specific semantics in mind). Thus, the question we raised earlier is relevant.

In this paper, we have the following goals: (1) To extend the semantics of Faber et al. [10] 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 generalization, 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 a certain 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.

2 Preliminaries

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 $\varphi ::= \bot \mid A \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi)$, 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 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

$$A_1 \wedge \ldots \wedge A_m \wedge \neg B_1 \wedge \ldots \wedge \neg B_n \rightarrow C_1 \vee \ldots \vee C_r \vee \neg D_1 \vee \ldots \vee \neg D_s, \qquad (1)$$

where A_i's, B_i's C_i's and D_i's are atoms. We call $A_1 \wedge \ldots \wedge A_m \wedge \neg B_1 \wedge \ldots \wedge \neg B_n$ and $C_1 \vee \ldots \vee C_r \vee \neg D_1 \vee \ldots \vee \neg D_s$ the *body* and the *head* of the rule, respectively. If $m = n = 0$, we represent the rule by its head. If $r = s = 0$, we write $\bot$ for the head of the rule. A *program* is a set of rules.

The stable-model semantics was defined first for *normal programs* (rule heads have exactly one atom and no negated atoms). It was later extended to *disjunctive programs* (rule heads have no negated atoms) [12], programs as understood here (that is, collections of rules as defined above) [13], and to *programs with nested expressions* [14]. Finally, the case of arbitrary theories was addressed by Pearce [5] and, later and in a different way, by Ferraris and Lifschitz [8, 9]. These two approaches are equivalent. We will now discuss each of them, starting with the latter one.

For a formula F and a set of atoms Y, we define the *GL-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 definitions 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 GL-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).

This notion of a stable model generalizes all earlier ones. It also coincides with the one proposed by Pearce [5]. The approach by Pearce is based on the logic HT [6], 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. $\langle X, Y\rangle \models_{ht} F \wedge G$ if $\langle X, Y\rangle \models_{ht} F$ and $\langle X, Y\rangle \models_{ht} G$
4. $\langle X, Y\rangle \models_{ht} F \vee G$ if $\langle X, Y\rangle \models_{ht} F$ or $\langle X, Y\rangle \models_{ht} G$
5. $\langle X, Y\rangle \models_{ht} F \rightarrow G$ if $Y \models F \rightarrow G$; and $\langle X, Y\rangle \not\models_{ht} F$, or $\langle X, Y\rangle \models_{ht} G$.

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 $\mathcal{F}$.

Pearce [5] 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*). Lifschitz and Ferraris [9] proved that the two approaches are equivalent by showing the following two key results.

Theorem 1. *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 $\langle X, Y\rangle \models_{ht} \mathcal{F}$.*

3 FLP Semantics

Faber et al. [10] 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

$$A_1 \wedge \ldots \wedge A_m \wedge \neg B_1 \wedge \ldots \wedge \neg B_n \rightarrow C_1 \vee \ldots \vee C_r,$$

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 GL-reduct) is either R, if $Y \models A_1 \wedge \ldots \wedge A_m \wedge \neg B_1 \wedge \ldots \wedge \neg B_n$, 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. [10] 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

$$F \rightarrow C_1 \vee \ldots \vee C_r, \tag{2}$$

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 $\mathcal{P} = \{\neg\neg A \rightarrow A\}$ has only one "Faber et al." stable model, $\emptyset$, but two standard stable models, $\emptyset$ and $\{A\}$).

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

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 GL-reduct. There is no change for $F = \bot$, $F = A$, where $A \in At$, and $F = G \circ H$, 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 $F = G \rightarrow H$. 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 $F = G \rightarrow H$:

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

However, under that choice, all occurrences of $\rightarrow$ (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 $\mathcal{F} = \{\neg\neg p\}$ and $\mathcal{G} = \{\neg q \rightarrow \neg\neg p\}$. As q 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 $Y = \{p\}$. Since $\neg\neg p = (p \rightarrow \bot) \rightarrow \bot$ and $Y \not\models p \rightarrow \bot$, 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 $Y \models \neg q$, $\mathcal{G}^{\underline{Y}} = \{\neg q \rightarrow \neg\neg p\}$. 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 brings up yet another one. In $\mathcal{G}$, p has no head occurrence (informally, there is no "defining clause" for p in $\mathcal{G}$), yet $\mathcal{G}$ would have $\{p\}$ as a "stable" model.

Thus, we need to handle the case of $\rightarrow$ 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 $\rightarrow$ in the consequent of another occurrence of $\rightarrow$ are treated consistently in the same non-classical way. In the remainder of this section we will argue that it can be accomplished by the following definition:

FLP4. $$(G \rightarrow H)^{\underline{Y}} = \begin{cases} G \rightarrow H^{\underline{Y}} & \text{if } Y \models G \text{ and } Y \models H \\ \top & \text{if } Y \not\models G \\ \bot & \text{otherwise (that is, when } Y \not\models G \rightarrow H). \end{cases}$$

While it looks different than the original definition by Faber et al. [10], it preserves its basic idea of keeping intact the bodies of rules that contribute to the reduct. Indeed, when the implication is "strongly" satisfied (both its antecedent and consequent are satisfied by Y), we keep the antecedent unchanged. However, to make sure the implications occurring in the antecedent are treated in a consistent way, we replace the consequent recursively with its reduct. The case when Y "weakly" satisfies the implication, that is, does not satisfy its antecedent, is dealt with as in the previous attempt

(and as in the definition by Faber et al.), reflecting the principle that if the implication "does not fire," it is immaterial and can be replaced by $\top$ ("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 GL-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 GL-reduct (adjusted to the notation $F^{\underline{Y}}$), as well as the clause (FLP4) for the implication $\rightarrow$. 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 $F^{\underline{Y}}$.*

3.2 Basic Properties

We start with a generalization of the well-known property of the standard GL-reduct of disjunctive programs (cf. Theorem 1).

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 $F^{\underline{Y}} = G^{\underline{Y}} \wedge H^{\underline{Y}}$. By the definition:

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 $Y \models H^{\underline{Y}}$.

By the induction hypothesis, the equivalence of $Y \models F$ and $Y \models F^{\underline{Y}}$ follows. 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 that 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. [10] coincide. Thus, our approach is a generalization of the one by Faber et al.

Theorem 2. *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. [10] 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 $R = F \rightarrow C_1 \vee \ldots \vee C_r$ 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. We also extend this notation to sets of such formulas.

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., are models [10]; for FLP-stable models, it follows from Corollary 1). Thus, $\mathcal{P}'$ consists of those rules $R = F \rightarrow C_1 \vee \ldots \vee C_r$, for which $Y \models F$. In addition, it might possibly contain $\top$. The reduct $\mathcal{P}''$ differs only in that each formula $R = F \rightarrow C_1 \vee \ldots \vee C_r$ from $\mathcal{P}$ that is retained in $\mathcal{P}'$, contributes to $\mathcal{P}''$ its reduct $R'' = F \rightarrow C'_1 \vee \ldots \vee C'_t$, where $C'_1, \ldots C'_t$ are precisely those elements in $\{C_1, \ldots, C_r\}$ 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 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. The next result shows that our generalization behaves properly.

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 $\mathcal{F}$.*

Finally, we note 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 $\vee$.*

3.3 Minimal-Model Property

The main objective of Faber et al. [10] 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. Let $F = \neg A \vee A$ and $Y = \emptyset$. Since $Y \models A \rightarrow \bot$ and $Y \not\models A$, $(\neg A)^{\underline{Y}} = (A \rightarrow \bot)^{\underline{Y}} = \top$. 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 $(\neg A)^{\underline{Z}} = (A \rightarrow \bot)^{\underline{Z}} = \bot$ 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 *disjunctive-monotone* formulas. 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.

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, $X \models F^{\underline{Y}}$. For the inductive step, there are three cases to consider.
Case 1. $F = G \wedge H$. Clearly, both formulas G and H are disjunctive-monotone, $X \models G, X \models H, Y \models G$ 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 $X \models H$. Wlog 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 $F^{\underline{Y}} \equiv G^{\underline{Y}} \vee H^{\underline{Y}}$, $X \models F^{\underline{Y}}$.
Case 3. $F = G \rightarrow H$. Since $Y \models F$, $F^{\underline{Y}} \neq \bot$. If $F^{\underline{Y}} = \top$ then $X \models F^{\underline{Y}} = \top$. Thus, we may assume that $Y \models G$, $Y \models H$ and $F^{\underline{Y}} = G \rightarrow H^{\underline{Y}}$. If $X \not\models G$, then $X \models F^{\underline{Y}}$. If $X \models G$, then $X \models H$. Since H is disjunctive-monotone, 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 $G \rightarrow H$, 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 monotone-disjunctive, then the result follows from Proposition 4. If $F = G \rightarrow H$, where H is monotone-disjunctive, we reason as follows. Since $Y \models F$, we have $F^{\underline{Y}} = \top$; or $Y \models G$, $Y \models H$ and $F^{\underline{Y}} = G \rightarrow H^{\underline{Y}}$. 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. □

Corollary 3. *Let $\mathcal{F}$ be a theory such that every formula in $\mathcal{F}$ is of the form H or $G \rightarrow H$, 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 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 $H_k \rightarrow (H_{k-1} \rightarrow (\ldots \rightarrow (H_1 \rightarrow H_0)\ldots))$, where $k \geq 0$ 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. It follows also that the problem to decide whether a model of a formula is a minimal model is in the class coNP. Thus, the existence of the FLP-stable model is in the class Σ_2^P. Σ_2^P-hardness of the problem follows from the fact that on disjunctive programs FLP-stable models coincide with stable models [10], 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 3. *The problem of the existence of the FLP-stable model is Σ_2^P-complete. The skeptical reasoning with FLP-stable models (is a given atom a member of every FLP-stable model) is Π_2^P-complete. The 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:

$5'$. $\langle X, Y\rangle \models_{flp} G \rightarrow H$ if $\quad Y \models G \rightarrow H$; 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 $\models_{ht}$ [9].

Proposition 5. *Let $\mathcal{F}$ be a theory. For every sets $X \subseteq Y \subseteq At$, if $\langle X, Y\rangle \models_{flp} \mathcal{F}$, then $Y \models \mathcal{F}$.*

Proof. It suffices to prove that for every formula F, if $\langle X, Y\rangle \models_{flp} F$, then $Y \models F$ 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 $F = G \rightarrow H$ and $\langle X, Y\rangle \models_{flp} F$ then, in particular, $Y \models F$ (by the definition of $\models_{flp}$ for the case of implication). Thus, the result follows. □

The $\models_{flp}$ relation and the FLP-reduct are closely connected (cf. Theorem 1).

Proposition 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, $\langle X, Y\rangle \models_{flp} A$. Conversely, if $\langle X, Y\rangle \models_{flp} A$, 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, $\langle X, Y\rangle \models_{\mathit{flp}} G$ and $\langle X, Y\rangle \models_{\mathit{flp}} H$. Thus, $\langle X, Y\rangle \models_{\mathit{flp}} G \wedge H$, as needed. Conversely, let $\langle X, Y\rangle \models_{\mathit{flp}} G \wedge H$. Then $\langle X, Y\rangle \models_{\mathit{flp}} G$ and $\langle X, Y\rangle \models_{\mathit{flp}} H$ 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, $G^{\underline{Y}} \wedge H^{\underline{Y}} \equiv (F \wedge G)^{\underline{Y}}$. Thus, $X \models (F \wedge G)^{\underline{Y}}$.

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

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

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

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 $\langle Y, Y\rangle \models_{\mathit{flp}} \mathcal{F}$ and for every $X \subset Y$, $\langle X, Y\rangle \not\models_{\mathit{flp}} \mathcal{F}$.*

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 Proposition 6. The former condition is equivalent to $\langle Y, Y\rangle \models_{\mathit{flp}} \mathcal{F}$. The latter one is equivalent to $\langle X, Y\rangle \not\models_{\mathit{flp}} \mathcal{F}$. Thus, the assertion follows. □

We are now ready to discuss the notion of strong FLP-equivalence. Theories $\mathcal{F}$ and $\mathcal{G}$ and *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 [7] to the case of FLP-stable models.

Theorem 4. *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 $\mathcal{H}$, $\langle X, Y\rangle \models_{\mathit{flp}} \mathcal{F} \cup \mathcal{H}$ if and only if $\langle X, Y\rangle \models_{\mathit{flp}} \mathcal{G} \cup \mathcal{H}$. 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. Wlog, we may assume that $\langle X, Y\rangle \models_{\mathit{flp}} \mathcal{F}$ and $\langle X, Y\rangle \not\models_{\mathit{flp}} \mathcal{G}$. By Proposition 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 $\langle Y, Y\rangle \not\models_{\mathit{flp}} \mathcal{G}$. Thus, for every $\mathcal{H}$, $Y \not\models \mathcal{G} \cup \mathcal{H}$ and so, 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-stable model of $\mathcal{F} \cup \mathcal{H}$. It follows that Y is not an FLP-stable model of $\mathcal{G} \cup \mathcal{H}$ and so, there is $Z \subset Y$ such that $Z \models (\mathcal{G} \cup \mathcal{H})^{\underline{Y}}$. Since $(\mathcal{G} \cup \mathcal{H})^{\underline{Y}} \equiv \mathcal{G}^{\underline{Y}} \cup \mathcal{H}$, $Z \models \mathcal{H}$ and so, $Z = X$. However, $X \not\models \mathcal{G}^{\underline{Y}}$, a contradiction. □

4 Normal Forms and a Comparison with Stable-Model Semantics

The following result was obtained by Cabalar and Ferraris [16]. It concerns representing theories by programs — theories consisting of rules (formulas of the form (1)).

Theorem 5. *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. In what follows we write $\neg Y$ for $\{\neg y \mid y \in Y\}$. We first state three auxiliary results (the proofs are simple and we omit them).

Proposition 7. *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 $V = Y$.*

Proposition 8. *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 $V = Y$.*

Proposition 9. *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.*

Theorem 6. *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 $X \subset Y$, we take the formula defined in Proposition 7 (with $Z = At(F)$). For each countermodel $\langle X, Y\rangle$ such that $X = Y$, we take the formula from Proposition 8. We take for $\mathcal{G}$ the set of all rules constructed in that way from countermodels of formulas in $\mathcal{F}$. By Proposition 9, $\mathcal{F}$ 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 that not every stable model of a theory is an FLP-stable model of a theory. We also note that not every FLP-stable model is a stable model. For instance, let $F = (A \vee \neg A) \rightarrow A$, and $Y = \{A\}$. It is easy to check that $F^Y = A \rightarrow A$. Since $\emptyset \models F^Y$, Y is not a stable model of F^Y. However, $F^{\underline{Y}} = (A \vee \neg A) \rightarrow A \equiv A$. Thus, Y is an FLP-stable model of F. It follows that the two semantics 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 10. *Let R be a rule. Then, R has the same HT- and FLP-models.*

Theorems 5 and 6 yield now the following two corollaries relating the two semantics.

Corollary 5. *For every theory $\mathcal{F}$ there are programs $\mathcal{F}'$ and $\mathcal{F}''$ 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 $\mathcal{F}'$*
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 $\mathcal{F}''$*

Corollary 6. *For every theory $\mathcal{F}$ there are programs $\mathcal{F}'$ and $\mathcal{F}''$ such that*

1. *Y is a stable model of $\mathcal{F}$ if and only if Y is an FLP-stable model of $\mathcal{F}'$*
2. *Y is an FLP-stable model of $\mathcal{F}$ if and only if Y is a stable model of $\mathcal{F}''$*

5 Supported Models

The approach that yielded generalizations of stable- and FLP-model semantics for arbitrary propositional theories can also be applied to 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. $$(G \rightarrow H)^{\underline{Y}}_{=} = \begin{cases} H^{\underline{Y}}_{=} & \text{if } Y \models G \text{ and } Y \models H \\ \top & \text{if } Y \not\models G \\ \bot & \text{otherwise.} \end{cases}$$

This notion of reduct is motivated by the definition of supported models in the case of programs with disjunctive rules (no negation in the head) [17]. The reduct that is relevant there consists of the heads of rules with bodies satisfied by Y. In the first case, we define the reduct $(G \rightarrow H)^{\underline{Y}}_{=}$ to be $H^{\underline{Y}}_{=}$ rather than just H due to the same reasons we followed when generalizing the FLP-reduct.

Definition 2. *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 $\mathcal{F}^{\underline{Y}}_{=}$.*

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

Proposition 11. *For every theory $\mathcal{F}$ and every set of atoms Y:*

1. *$Y \models \mathcal{F}$ if and only if $Y \models F^{\underline{Y}}_{=}$*
2. *if Y is a supported model of $\mathcal{F}$, then Y is a model of $\mathcal{F}$*
3. *if Y is a supported model of $\mathcal{F}$, then every atom in Y has a head occurrence in $\mathcal{F}$.*

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.

$5''$. $\langle X, Y\rangle \models_{spp} F \rightarrow G$ if $Y \models F \rightarrow G$, and $Y \not\models F$ or $\langle X, Y\rangle \models_{spp} G$.

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

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

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

The main consequence of Proposition 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 $X \subset Y$, $\langle X, Y\rangle \not\models_{spp} \mathcal{F}$.*

We use these results to study strong SPP-equivalence of theories. 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 the same SPP-models then they are strongly SPP-equivalent. Unlike in the other two cases, though, a weaker condition suffices to provide 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 $A \in At$. We now have the following characterization of strong SPP-equivalence. Unlike the one developed for programs [18], where the general case is established through a certain reduction to normal programs, the present characterization is direct. We state first an auxiliary property, which can be demonstrated by simple induction.

Proposition 13. *For every formula F and for every interpretation Y, $F^{\underline{Y}}_{=}$ is monotone.*

Theorem 7. *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 SPP-model of $\mathcal{F}$. It follows that Y is a supported model of $\mathcal{F} \cup Y$. Thus, Y is a supported model of $\mathcal{G} \cup Y$. Consequently, Y is a model of $\mathcal{G}$ and so, $\langle Y, Y\rangle$ is an SPP-model of $\mathcal{G}$ (indeed, by Proposition 11, $Y \models G^{\underline{Y}}_{=}$, and the claim follows by Proposition 12). Next, let $\langle Y \setminus \{A\}, Y\rangle$ be an SPP-model of $\mathcal{F}$. It follows that $Y \setminus \{A\} \models \mathcal{F}^{\underline{Y}}_{=}$. Let us assume that $\langle Y \setminus \{A\}, Y\rangle$ is not an SPP-model of $\mathcal{G}$. Then, $Y \setminus \{A\} \not\models \mathcal{G}^{\underline{Y}}_{=}$. Let us consider $\mathcal{G} \cup (Y \setminus \{A\})$. Since $\langle Y, Y\rangle$ is an SPP-model of $\mathcal{F}$, $\langle Y, Y\rangle$ is an SPP-model of $\mathcal{G}$ (we proved that above). Since every model of $(\mathcal{G} \cup (Y \setminus \{A\}))^{\underline{Y}}_{=} \equiv \mathcal{G}^{\underline{Y}}_{=} \cup (Y \setminus \{A\})$ 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 we have $Y \setminus \{A\} \models (\mathcal{F} \cup (Y \setminus \{A\}))^{\underline{Y}}_{=}$, a contradiction. Thus, $\langle Y \setminus \{A\}, Y\rangle$ is an essential SPP-model of $\mathcal{G}$. By symmetry, 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, $\langle Y, Y\rangle \models_{spp} \mathcal{G} \cup \mathcal{H}$ and so, $Y \models (\mathcal{G} \cup \mathcal{H})^{\underline{Y}}_{=}$. 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 13, $Y \setminus \{A\} \models \mathcal{G}^{\underline{Y}}_{=}$ 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 $\mathcal{F}$. Moreover, $Y \setminus \{A\} \models \mathcal{F}^{\underline{Y}} \cup \mathcal{H}^{\underline{Y}} = (\mathcal{F} \cup \mathcal{H})^{\underline{Y}}$. This contradicts Y being a supported model of $\mathcal{F} \cup \mathcal{H}$. Thus, no such X exists and Y is a supported model of $\mathcal{G} \cup \mathcal{H}$. The claim follows now by the symmetry argument. □

We conclude with two properties of supported-model semantics. The first one relates (FLP-)stable and supported models. The second one is a normal form result.

Theorem 8. *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 $\mathcal{F}$.*

Theorem 9. *Let $\mathcal{F}$ be a theory. Then there is a normal program $\mathcal{G}$ 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).*

6 Discussion and Conclusions

Ferraris and Lifschitz [9] proved 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 GL-reduct. We showed that the approach by Ferraris and Lifschitz 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. 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 $\models_{ht}$, which captures the stable-models semantics defines a logic, namely the logic HT. To the contrary, the relation $\models_{flp}$, which captures the FLP semantics 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. 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.

References

1. Gelfond, M., Lifschitz, V.: The Stable Semantics for Logic Programs. In: Kowalski, R.A., Bowen, K.A. (eds.) Proceedings of the 5th International Conference and Symposium on Logic Programming, pp. 1070–180. MIT Press, Cambridge, MA (1988)
2. Marek, V., Truszczyński, M.: Stable Models and an Alternative Logic Programming Paradigm. In: Apt, K., Marek, W., Truszczyński, M., Warren, D. (eds.) The Logic Programming Paradigm: a 25-Year Perspective, pp. 375–398. Springer, Berlin (1999)
3. Niemelä, I.: Logic Programming with Stable Model Semantics as a Constraint Programming Paradigm. Annals of Mathematics and Artificial Intelligence 25, pp. 241–273 (1999)
4. Gelfond, M., Leone, N.: Logic Programming and Knowledge Representation – the A-Prolog Perspective. Artificial Intelligence 138, pp. 3–38 (2002)
5. Pearce, D.: A New Logical Characterisation of Stable Models and Answer Sets. In: Dix, J., Pereira, L.M., Przymusinski, T.C. (eds.) Non-Monotonic Extensions of Logic Programming. LNCS, vol. 1216, pp. 57–70 Springer, Berlin (1997)
6. Heyting, A.: Die Formalen Regeln der Intuitionistischen Logik. Sitzungsberichte der Preussischen Akademie von Wissenschaften. Physikalisch-mathematische Klasse, pp. 42–56 (1930)
7. Lifschitz, V., Pearce, D., Valverde, A.: Strongly Equivalent Logic Programs. ACM Transactions on Computational Logic 2(4), pp. 526–541 (2001)
8. Ferraris, P.: Answer Sets for Propositional Theories. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) Logic Programming and Nonmonotonic Reasoning, 8th International Conference, LNAI, vol. 3662, pp. 119–131. Springer, Berlin (2005)
9. Ferraris, P., Lifschitz, V.: Mathematical Foundations of Answer Set Programming. In Artëmov, S., Barringer, H., d'Avila Garcez, A., Lamb, L.C., Woods, J. (eds.) We Will Show Them! Essays in Honour of Dov Gabbay, pp. 615–664. College Publications (2005)
10. Faber, W., Leone, N., Pfeifer, G.: Recursive Aggregates in Disjunctive Logic Programs: Semantics and Complexity. In: Alferes, J.J., Leite, J.A. (eds.) Proceedings of the 9th European Conference on Artificial Intelligence. LNAI, vol. 3229, pp. 200–212. Springer, Berlin (2004)
11. Niemelä, I., Simons, P.: Extending the Smodels System with Cardinality and Weight Constraints. In Minker, J. (ed.) Logic-Based Artificial Intelligence, pp. 491–521. Kluwer Academic Publishers, Boston (2000)
12. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing 9, pp. 365–385 (1991)
13. Lifschitz, V., Woo, T.: Answer Sets in General Nonmonotonic Reasoning. In: Nebel, B., Rich, C., Swartout, W.R. (eds.) Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning, pp. 603–614, Morgan Kaufmann (1992)
14. Lifschitz, V., Tang, L.R., Turner, H.: Nested Expressions in Logic Programs. Annals of Mathematics and Artificial Intelligence 25, pp. 369–389 (1999)
15. Eiter, T., Gottlob, G.: On the Computational Cost of Disjunctive Logic Programming: Propositional Case. Annals of Mathematics and Artificial Intelligence 15, pp. 289–323 (1995)
16. Cabalar, P., Ferraris, P.: Propositional Theories are Strongly Equivalent to Logic Programs. Theory and Practice of Logic Programming 7, pp. 745–759 (2007)
17. Brass, S., Dix, J.: Characterizations of the Disjunctive Stable Semantics by Partial Evaluation. Journal of Logic Programming 32, pp. 207–228 (1997)
18. Truszczyński, M., Woltran, S.: Hyperequivalence of Logic Programs with Respect to Supported Models. In: Fox, D., Gomes, C.P. (eds.) Proceedings of the 23rd National Conference on Artificial Intelligence, pp. 560–565. AAAI Press (2008)