CHARACTERIZATIONS OF THE DISJUNCTIVE STABLE SEMANTICS BY PARTIAL EVALUATION

STEFAN BRASS AND JÜRGEN DIX

There are three most prominent semantics defined for certain subclasses of disjunctive logic programs: GCWA (for positive programs), PERFECT (for stratified programs), and STABLE (defined for the whole class of all disjunctive programs). While there are various competitors based on 3-valued models, notably WFS and its disjunctive counterparts, there are no other semantics consisting of two-valued models. We argue that the reason for this is the Partial Evaluation property (also called Unfolding or Partial Deduction) well known from logic programming. In fact, we prove characterizations of these semantics and show that if a semantics SEM satisfies Partial Evaluation and Elimination of Tautologies, then (1) SEM is based on two-valued minimal models for positive programs, and (2) if SEM satisfies in addition Elimination of Contradictions, it is based on stable models. We also show that if we require Isomorphy and Relevance, then STABLE is completely determined on the class of all stratified disjunctive logic programs. © Elsevier Science Inc., 1997

1. INTRODUCTION

The generalized closed world assumption GCWA for positive disjunctive programs (introduced in [24]), the perfect semantics PERFECT for stratified programs (introduced in [25]), and the stable semantics STABLE for the class of all disjunctive programs (introduced in [22, 26]) are the most prominent semantics based on two-valued models. Why are there no other such semantics? We answer...
this question by introducing a framework that enables us to prove characterizations of these semantics and thus to detect the real principles behind them. The starting point is the observation that all these semantics satisfy certain abstract conditions—the most important one being the Partial Evaluation property known from logic programming. This has been shown in [4] and independently in [30]. Our aim in this paper is to prove the converse, namely, that only these semantics satisfy Partial Evaluation and some additional properties.

The underlying notion of a semantics is very general, namely, a mapping from logic programs to sets of models. We do not even require that a semantics is based on two-valued models, although for some characterizations we have to add a property (Elimination of Contradictions) that implies this. Our approach is based on purely abstract properties of a semantics SEM. These conditions come in the form of syntactical transformations $P \rightarrow P'$ on instantiated logic programs. For a particular syntactical transformation, the corresponding condition states that the transformation is equivalence-preserving, i.e., it does not change the underlying semantics: $\text{SEM}(P) = \text{SEM}(P')$.

We distinguish between partial and complete characterizations of a semantics SEM. While the first notion states that any semantics (satisfying certain properties) is contained in SEM (i.e., it may select a subset of the models), the latter notion states that any such semantics in fact coincides with SEM.

Our abstract properties can be illustrated by introducing some additional semantics weaker than GCWA and STABLE. This means that our framework allows us to distinguish very carefully between various possible semantics. We generalize the notion of supported model from the nondisjunctive to the disjunctive context and get two different notions: the weakly supported (Weak-SUPP) and the supported (SUPP) models. Weakly supported models are obtained from our underlying notion of a semantics by requiring Elimination of Contradictions and GPPE, the Generalized Principle of Partial Evaluation (see Lemma 4.4). Note that weakly supported and supported models collapse for normal programs to the well-known notion of supported model defined in [1]. Minimal models for positive disjunctive programs are obtained from GPPE and Elimination of Tautologies (see Theorem 4.1). Stable models for all disjunctive programs are obtained by still adding Elimination of Contradictions (see Theorem 4.4).

To get the whole set of all minimal models for positive programs (i.e., characterizing GCWA completely) or to get the whole set of all stable models for stratified disjunctive programs (i.e., characterizing PERFECT completely), we only have to assume two additional properties: Relevance and Isomorphy (see Theorem 4.2 and Theorem 4.5).

Abstract properties of logic programming semantics have been already investigated in [14–16] for normal (i.e., nondisjunctive) and in [18] for disjunctive programs. Here we build on and use some results of our recent work [7]. In the companion paper [9], we investigate the normal form of a program and show how it can be used to compute various semantics.

The paper is organized as follows. Section 2 introduces the semantics Weak-SUPP, SUPP, and reviews GCWA, PERFECT, and STABLE. We also introduce our abstract properties as certain equivalence-preserving transformations. In Section 3, we investigate which of our properties are satisfied for these semantics. Finally, in Section 4, we give characterizations of these semantics: any semantics satisfying certain abstract properties is characterized by these. Our main results are Theorems 4.1, 4.2, 4.4, and 4.5. We conclude with Section 5.
2. SEMANTICS AND TRANSFORMATIONS

In this section, we first introduce our setting of what we call a semantics (Subsection 2.1). We then present our syntactical transformations (Subsection 2.2), and finally we define four properties to ensure a "good" behavior of a semantics (Subsection 2.3).

2.1. Semantics

We consider disjunctive programs over some fixed signature \( \Sigma \). We assume that \( \Sigma \) is infinite, so that there is always one more unused ground atom.

**Definition 2.1** (Program \( P \)). A program \( P \) is a finite set of ground rules of the form

\[
A_1 \lor \cdots \lor A_k \leftarrow B_1 \land \cdots \land B_m \land \neg C_1 \land \cdots \land \neg C_n,
\]

where the \( A_i, B_i, C_i \) are \( \Sigma \)-atoms, \( k \geq 1, m \geq 0, \) and \( n \geq 0 \). We identify such a rule with the triple consisting of the three sets of atoms \( \mathcal{A} := \{A_1, \ldots, A_k\}, \mathcal{B} := \{B_1, \ldots, B_m\}, \mathcal{C} := \{C_1, \ldots, C_n\} \), and denote it in the form \( \mathcal{A} \leftarrow \mathcal{B} \land \neg \mathcal{C} \).

**Definition 2.2** (Three-valued model \( I \)). A three-valued Herbrand interpretation \( I \) (or short: an interpretation) is a mapping which assigns to every ground atom \( A \) a number \( I[A] \in \{-1, 0, 1\} \). We identify \(-1\) with false (f), \(0\) with undefined (u), and \(1\) with true (t).

We also use the notation True(\( I \)), False(\( I \)), and Undef(\( I \)) for the sets of true, false, and undefined ground atoms of \( I \), restricted to atoms actually occurring in the underlying program (so that all these sets are finite.)

An interpretation \( I \) is a model of a logic program \( P \) if and only if (iff) for every rule

\[
A_1 \lor \cdots \lor A_k \leftarrow B_1 \land \cdots \land B_m \land \neg C_1 \land \cdots \land \neg C_n
\]

in \( P \), the following holds:

\[
\max\{I[A_1], \ldots, I[A_k]\} \geq \min\{I[B_1], \ldots, I[B_m], -I[C_1], \ldots, -I[C_n]\}.
\]

In particular, if an interpretation \( I \) makes the body of a rule true, then the head must also be true in \( I \). If the body is undefined, then the head cannot be false—\( I[\text{head}] \) must be either true or undefined.

**Definition 2.3** (Semantics \( \text{SEM} \)). A semantics \( \text{SEM} \) is a mapping from a class of logic programs \( P \) and \( \Sigma \) into the set of three-valued Herbrand models

\[
\text{SEM}(P) \subseteq \text{MOD}^{\text{Herbrand}}_{3-val}(P),
\]

with the following additional restriction: If a ground atom \( A \in \Sigma \) does not occur in \( P \), then \( \text{SEM}(P) \models \neg A \) (i.e., for every \( I \in \text{SEM}(P) \): \( I[A] = -1 \)).

Note that not all semantics are defined on the whole class of all disjunctive programs. Our results hold for all classes of programs that are closed under the transformations to be introduced below and that contain with any program \( P \) also \( P \cup \{A \leftarrow \text{Body}\} \), where \( A \) is a new atom. In particular, they hold for the classes of positive disjunctive, general disjunctive, positive nondisjunctive, stratified nondisjunctive, and general nondisjunctive programs.
We could also have defined a semantics as a mapping into a set of two-valued models. In fact, our Elimination of Contradictions condition just implies this. But this assumption is not needed for our characterization theorems for GCWA (Theorems 4.1 and 4.2). And, obviously, the weaker the underlying notion of a semantics, the stronger the characterization theorems.

The semantics we are interested in are indeed based on two-valued models.

**Definition 2.4** (Weakly supported, supported, and stable models). A two-valued model $I$ of a (disjunctive) logic program $P$ is

(a) **weakly supported** iff for every ground atom $A$ with $I \models A$, there is a rule $\mathcal{A} \leftarrow \mathcal{B} \land \neg \mathcal{C}$ in $P$ with $A \in \mathcal{A}$ and $I \models \mathcal{B} \land \neg \mathcal{C}$,

(b) **supported** iff for every ground atom $A$ with $I \models A$, there is a rule $\mathcal{A} \leftarrow \mathcal{B} \land \neg \mathcal{C}$ in $P$ with $A \in \mathcal{A}$, $I \models \mathcal{B} \land \neg \mathcal{C}$, and $\mathcal{A} \neq \{A\},$

(c) **stable** iff $I$ is a minimal model of the positive disjunctive program $P/I$.

Here $P/I$ is the GL-Transform of $P$ with respect to $(\text{wrt}) I$. It is obtained from $P$ by evaluating all negative literals according to $I$: if $I \models \neg A$, then drop $\neg A$ from the clause; if $I \models A$, then drop the whole clause containing $\neg A$.

We use $\text{SUPP}(P)$, $\text{Weak-SUPP}(P)$, and $\text{STABLE}(P)$ for the respective sets of intended models. Another famous semantics is GCWA. It is only defined for positive disjunctive programs and given by the set of all minimal (two-valued) models:

$$\text{GCWA}(P) := \text{Min-MOD}^\text{Herbrand}_{\text{z-val}}(P).$$

While STABLE extends GCWA (in the sense that both coincide for positive disjunctive programs), this is true neither for Weak-SUPP nor for SUPP. It is well known that PERFECT (introduced by Przymusinski in [25]) coincides for stratified programs with STABLE (the stable semantics for disjunctive programs has been introduced independently for Gelfond and Lifschitz in [22] and by Przymusinski in [26]), and we will think, therefore, of PERFECT as the restriction of STABLE to this class of programs.

### 2.2. Transformations

To illustrate very clearly the differences of logic programming semantics, we base our discussion on abstract properties. All of them require that certain elementary transformations do not change the semantics of a given logic program, i.e., are SEM-equivalence transformations.

**Definition 2.5** (Equivalence transformation). A transformation $\rightarrow$ is an arbitrary binary relation on the class of all programs. We call it a SEM-equivalence transformation iff $\text{SEM}(P_1) = \text{SEM}(P_2)$ for all programs $P_1, P_2$ with $P_1 \rightarrow P_2$.

The following two transformations allow us to eliminate certain rules. They are special cases of the D-reduction introduced in [18].
Definition 2.6 (Elimination of Tautologies). A semantics SEM allows the elimination of tautologies iff the following transformation on instantiated logic programs is a SEM-equivalence transformation:

- Delete a rule \( \mathcal{A} \leftarrow \mathcal{B} \land \neg \mathcal{C} \) with \( \mathcal{A} \cap \mathcal{B} \neq \emptyset \).

Definition 2.7 (Elimination of Contradictions). A semantics SEM allows the elimination of contradictions iff the following transformation on instantiated logic programs is a SEM-equivalence transformation:

- Delete a rule \( \mathcal{A} \leftarrow \mathcal{B} \land \neg \mathcal{C} \) with \( \mathcal{B} \cap \mathcal{C} \neq \emptyset \).

The most important transformation, however, is partial evaluation in the sense of the “unfolding” operation (see [19, 4, 30]).

Definition 2.8 (GPPE). A semantics SEM satisfies GPPE iff the following transformation on instantiated logic programs is a SEM-equivalence transformation:

- Suppose that the atom \( B \) occurs in the heads of the \( n \geq 0 \) rules
  \[ \mathcal{A}_i \leftarrow \mathcal{B}_i \land \neg \mathcal{C}_i, \quad (i = 1, \ldots, n). \]

  Then replace a rule of the form
  \[ \mathcal{A} \leftarrow (\mathcal{B} \cup \{B\}) \land \neg \mathcal{C} \]

  by the \( n \) rules

  \[ (\mathcal{A} \cup (\mathcal{A}_i - \{B\})) \leftarrow (\mathcal{B} \cup \mathcal{B}_i) \land \neg (\mathcal{C} \cup \mathcal{C}_i), \quad (i = 1, \ldots, n). \]

  So this operation does nothing else than replacing the subgoal \( B \) by the bodies of all rules about \( B \) (known as “unfolding”). If, however, \( B \) has a disjunctive context in the head, this disjunctive context is added to the head of the resulting rule (this is a resolution step).\(^1\)

Let us illustrate these transformations with an example.

Example 2.1 (Illustrating example).

\[ P: A \lor B \leftarrow C, \quad \neg C \]
\[ A \lor C \leftarrow B, E \]
\[ E \lor B \leftarrow \neg B \]

By Elimination of Contradiction, we can eliminate the first rule. We then apply GPPE to replace the occurrence of \( B \) in the second rule. We get

\[ P': A \lor C \lor E \leftarrow E \lor C \]
\[ E \lor B \leftarrow \neg B \]

and can eliminate the first rule by Elimination of Tautologies and end up with the single rule \( E \lor B \leftarrow \neg B \).

\(^1\) We do not exclude the case \( B \in \mathcal{B} \) corresponding to a “double occurrence” of \( B \) in the body, of which only one is unfolded. This is needed in the proof of Lemma 4.4. On the other hand, as the example \( P := \{q \leftarrow p, p \lor p\} \) shows, it is important for Lemma 3.1 that we remove all occurrences of \( B \) from the heads \( \mathcal{A}_i \).
2.3. Consistency, Independence, Relevance, and Isomorphy

Besides allowing the above transformations, we would require from a good semantics that it has also the following four natural properties. Three of them point to a weakness of the stable semantics discussed elsewhere ([19]).

**Definition 2.9 (Consistency).** A semantics SEM satisfies **Consistency** iff SEM(P) ≠ ∅ for all programs P. We call a semantics **trivial** iff SEM(P) = ∅ for all P.

**Definition 2.10 (Independence).** A semantics SEM satisfies **Independence** iff

\[
\text{SEM}(P_1) \models Q \iff \text{SEM}(P_1 \cup P_2) \models Q,
\]

provided that the predicates occurring in P_1 and P_2 are disjoint, and the query Q contains only the predicates from P_1.

The requirement that a semantics should be **consistent** is immediate, although one could object that not even classical logic satisfies it. But this is no convincing argument because in the case of logic programs, we have a very restricted language and it should not be possible to explicitly express inconsistency (note that no negative literals can appear in the head of program clauses).

Independence goes a small step further and formalizes the idea that if a program P can be split into two disjoint parts that have nothing to do with each other (i.e., P = P_1 \cup P_2 and no predicate occurs in both parts), then the meaning of a predicate p with respect to the whole program P coincides with p’s meaning with respect to the part it belongs to. Independence expresses a kind of **consistency-persistence:** it holds in classical logic, provided that P_1 and P_2 are consistent. There, it corresponds to the well-known notion of **conservative extension.**

Independence is implied by the next condition, which has been introduced in [14] (see also [19]). This condition is very natural because it is the underlying principle of all top-down query evaluation methods: clauses that contain only predicates that have nothing to do with a given literal A should not affect A’s truth-value. More precisely: the truth-value of a literal should only depend on the call-graph below it. It is well known that any program P induces a notion of **dependency** between its atoms. We say that A **depends immediately** on B iff

- B appears in the body of a clause in P, such that A appears in its head, or
- A and B appear together in the head of a clause.

The binary relation **depends on** is the reflective and transitive closure of **depends immediately on.** The **dependence of** and the **rules relevant for** a set M of atoms are now defined by:

- \( \text{dependencies\_of}(M) \) is the set of atoms A such that there is \( B \in M \) which depends on A,
- \( \text{rel\_rul}(P, M) \) is the set of relevant rules of P with respect to M, i.e., the set of rules that contain an \( A \in \text{dependencies\_of}(M) \) in their head.

Our condition formalizes that if we are given a program P but are only interested in (determining the truth-values of) atoms belonging to a certain set M, then it is completely sufficient to look at the subset of P consisting of the rules relevant for M. Since this set P' usually is a proper subset of P formulated in a smaller language, the elements of SEM(P') and SEM(P) are in general incomparable. Therefore, we need the notion "\( \text{RSP}_{p}(I) \)" of a "reduct of an interpretation I"
in SEM(P) to a model of P"':
\[ \text{RE}_{P}(I)[A] := \begin{cases} I[A] & \text{if } A \text{ occurs in } P', \\
I & \text{otherwise.} \end{cases} \]

**Definition 2.11** (Relevance). Let P be a program and M be a set of atoms occurring in P. A semantics SEM satisfies Relevance iff
1. SEM(rel_rul(P, M)) = rel_rul(P, M)(SEM(P)), i.e., SEM(rel_rul(P, M)) consists exactly of the reducts of SEM(P) to rel_rul(P, M), and
2. if I ∈ SEM(P ∪ {X new ∨ A}) (where X new is a new atom not occurring in P) and I[¬A] = t, then \( \text{RE}_{P}(I) \in \text{SEM}(P) \).

The second condition simply states that the reduct of an intended model I for P ∪ {X new ∨ A} should also be an intended model for P. Adding a rule X new ∨ A to P makes it possible for A to become true. But our special assumption that I[¬A] = t forces X new to be true in I and, therefore, all remaining atoms should also form an intended model for the program P alone.

A very special case of Relevance that we needed later is the following. Let A occur in P and X new be a new atom not occurring in P. Then
\[ \text{RE}_{P}(\text{SEM}(P \cup \{X_{\text{new}} \leftarrow \neg A\})) = \text{SEM}(P). \]

This is a technical property needed to prove "(1) ⇒ (2)" in Theorem 4.3. Every sensible semantics should satisfy it, even semantics which do not satisfy Relevance in general.

Let us note that all our transformations are still very weak. As an example, it is possible to construct a semantics SEM (satisfying all our properties) which selects only one minimal model from the program consisting of "A ∨ B." This is strange because the program is completely symmetric in A and B and, therefore, if \{A\} ∈ SEM((A ∨ B)), then also \{B\} ∈ SEM((A ∨ B)) should hold. It turns out later (Theorem 4.2 and 4.5) that the following property is indeed sufficient to exclude such anomalous behavior.

**Definition 2.12** (Isomorphy). A semantics SEM satisfies Isomorphy iff
\[ \text{SEM}(J(P)) = J(\text{SEM}(P)) \]
for all programs P and isomorphisms J on the set of all Σ-ground atoms.

This condition ensures that a semantics is invariant under a renaming (namely, an isomorphism J) of the underlying signature. The programs J(P) and P are syntactically different, but considered to represent equivalent programs, in the sense that their semantics coincide via J.

It is easy to see that the first three properties are of increasing strength.

**Lemma 2.1** (Relevance ⇒ Independence ⇒ Consistency).

(a) If SEM satisfies Relevance, then it also satisfies Independence.

(b) Let SEM be a nontrivial semantics satisfying Isomorphy. If SEM satisfies Independence, then it also satisfies Consistency.

**Proof.** (a) is immediate by definition. To prove (b), let SEM be a nontrivial semantics satisfying Isomorphy and Independence. Let P be given. We have to prove SEM(P) ≠ ∅. The nontriviality implies that there is a P' with SEM(P') ≠ ∅. Using Isomorphy, we can assume without loss of generality (w.l.o.g.) that P and P'
have no common symbols. We can now apply Independence and get
\[ \text{SEM}(P) \models f \iff \text{SEM}(P \cup P') \models f \iff \text{SEM}(P') \models f, \]
which contradicts \( \text{SEM}(P') \neq \emptyset \). □

3. PROPERTIES OF THE SEMANTICS

The next theorem illustrates how most semantics behave according to our conditions. Before we prove this theorem by a series of lemmas, we first give some additional explanations and comments.

We note that \( M_{\text{upp}}^{\sharp} \), the perfect Herbrand model introduced in [1], shares all our properties. While Clark's completion comp ([13]) fails to satisfy Elimination of Tautologies, Consistency, Independence as well as Relevance, the well-founded semantics only fails Elimination of Contradictions.

While GCWA satisfies all the properties introduced in the last section, STABLE satisfies all except for Consistency, Independence, and Relevance (see also [19, 20]). Let us note that from all our transformations, only GPPE is not closed for a particular class of programs: GPPE might transform a stratified disjunctive program into a nonstratified one. Nevertheless, we can say that PERFECT satisfies GPPE, because STABLE does (for general disjunctive programs) and PERFECT coincides with the restriction of STABLE to stratified disjunctive programs. As already noted in the introduction, GPPE for STABLE and GCWA has also been established independently by Sakama and Seki in [30].

While both Weak-SUPP and SUPP fail to satisfy elimination of tautologies, at least SUPP satisfies GPPE (Weak-SUPP does not). Neither Weak-SUPP, SUPP, nor STABLE satisfy Independence.

**Theorem 3.1** (Properties of Various Semantics). The following table summarizes the properties of various semantics:

<table>
<thead>
<tr>
<th>Properties of Logic-Programming Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>( M_{\text{upp}}^{\sharp} )</td>
</tr>
<tr>
<td>comp</td>
</tr>
<tr>
<td>WFS</td>
</tr>
<tr>
<td>GCWA</td>
</tr>
<tr>
<td>PERFECT</td>
</tr>
<tr>
<td>Weak-SUPP</td>
</tr>
<tr>
<td>SUPP</td>
</tr>
<tr>
<td>STABLE</td>
</tr>
<tr>
<td>STATIC</td>
</tr>
<tr>
<td>D-WFS</td>
</tr>
</tbody>
</table>

Note that the first three semantics are only defined for nondisjunctive programs (\( M_{\text{upp}}^{\sharp} \) only for stratified ones). GCWA ([24]) is defined for positive disjunctive and PERFECT ([25]) for stratified disjunctive programs. The remaining semantics are defined for the whole class of all disjunctive programs. The semantics STATIC ([27, 11]) and D-WFS ([4, 7]) are listed only for completeness. Their properties are considered in [5, 9].
That WFS and $M_p^{supp}$ satisfy GPPE has been proved in [28, 29] and [2]. WFS does not eliminate contradictions as shown in the following.

**Example 3.1.** We consider the program

$$
A \leftarrow \neg A
$$

$$
B \leftarrow A, \neg A
$$

$$
B \leftarrow B
$$

The well-founded model is obviously empty: $A, B$ are undefined. But if we apply Elimination of Contradictions, we have to remove the second clause. The well-founded model of the resulting program now derived $\neg B$.

### 3.1. Properties of GCWA, STABLE, and PERFECT

**Lemma 3.1** (Minimal models are invariant under applying GPPE). Let $P$ be a positive instantiated logic program and $P'$ be the result of applying GPPE to $P$, i.e., for some rule $\mathcal{A} \leftarrow \mathcal{B} \cup \{B\}$ in $P$:

$$
P' := P - \{\mathcal{A} \leftarrow \mathcal{B} \cup \{B\}\}
$$

$\cup \{\mathcal{A} \cup (\mathcal{A}' - \{B\}) \leftarrow (\mathcal{B} \cup \mathcal{B}'), \mathcal{A} \leftarrow \mathcal{B}' \in P with B \in \mathcal{A}'\}.$

**Proof.** We first show that a minimal model of $P$ (resp., $P'$) also is a model of the other logic program:

1. Let $I$ be a minimal model of $P$ (in fact, in this direction, we do not need the minimality of $I$). If $I$ would not be a model of $P'$, it would have to violate one of the new rules, e.g.,

$$
\mathcal{A} \cup (\mathcal{A}' - \{B\}) \leftarrow (\mathcal{B} \cup \mathcal{B}').
$$

Violating it means that $I \not\models B$, $I \models \mathcal{B}'$, $I \not\models \mathcal{A}$, $I \not\models \mathcal{A}' - \{B\}$. But since we know that $I \models \mathcal{A}' \leftarrow \mathcal{B}'$ (it is contained in $P$), we can conclude that $I \models B$. But now $I \not\models \mathcal{A} \leftarrow \mathcal{B} \cup \{B\}$ (for the same reason) yields $I \not\models \mathcal{A}$, which is a contradiction.

2. Let $I$ be a minimal model of $P'$. If $I$ would not be a model of $P$, it would have to violate $\mathcal{A} \leftarrow \mathcal{B} \cup \{B\}$ (the only element of $P' - P$). So we would have $I \models \mathcal{B}$, $I \models B$, and $I \not\models \mathcal{A}$. Since $I$ is a minimal model of $P'$, $I_0 := I - \{B\}$ cannot be a model of $P'$. So it must violate a rule $\mathcal{A}' \leftarrow \mathcal{B}'$ in $P'$, and, by the construction, $B \in \mathcal{A}'$ and $B \notin \mathcal{B}'$. It follows that $I \models \mathcal{B}'$ and $I \notin \mathcal{A}' - \{B\}$. But this means that $I$ does not satisfy the rule

$$
\mathcal{A} \cup (\mathcal{A}' - \{B\}) \leftarrow (\mathcal{B} \cup \mathcal{B}'),
$$

which is contained in $P'$, again a contradiction.

Now it immediately follows that a minimal model $I$ of $P$ is not only a model of $P'$ but also a minimal model: Suppose it would not, i.e., there were a model $I_1$ with $I_1 \prec I$. Then there would be also a minimal model $I_0$ with $I_0 \leq I_1$, i.e., $I_0 \prec I$. But this minimal model $I_0$ of $P'$ is also a model of $P$, contradicting the assumed minimality of $I$.

The same argument holds with $P$ and $P'$ interchanged. □
Lemma 3.2. STABLE satisfies GPPE.

PROOF. So let \( P \) be an instantiated logic program containing a rule
\[
\mathcal{A} \leftarrow (\mathcal{B} \cup \{B\}) \land \neg \mathcal{C},
\]
and let \( P' \) be the result of applying GPPE to the body literal \( B \). We have to show for any Herbrand interpretation \( I \):
\[
I \text{ is a stable model of } P \text{ iff } I \text{ is a stable model of } P'.
\]
We distinguish two cases:

1. \( I \not\models \neg \mathcal{C} \): Then neither the above rule nor the rules resulting from GPPE are contained in the transformed program, so we have that \( P/I = P'/I \). This trivially implies that \( I \) is a stable model of \( P \) iff it is a stable model of \( P' \).
2. \( I \models \neg \mathcal{C} \): We again distinguish two cases.
   (a) \( P'/I \) simply results from \( P/I \) by the corresponding application of GPPE. For instance, let
   \[
   \mathcal{A} \vee (\mathcal{A}' - \{B\}) \leftarrow B' \land B' \land \neg(\mathcal{C} \land \mathcal{C}')
   \]
   be a rule resulting from the application of GPPE to \( P \), namely, by inserting
   \[
   \mathcal{A}' \leftarrow B' \land \neg \mathcal{C}'
   \]
   into the above rule. These rules are only relevant for the transformed programs if \( I \not\models \mathcal{C}' \). But then the corresponding positive parts are again related by GPPE. Now we can apply Lemma 3.1 to conclude that \( P/I \) and \( P'/I \) have the same minimal models. So \( I \) is either a minimal model of both or of none.
   (b) It is also possible that the rule \( \mathcal{A} \leftarrow (\mathcal{B} \cup \{B\}) \land \neg \mathcal{C} \) is still contained in \( P'/I \) if it results not only from \( \mathcal{A} \leftarrow (\mathcal{B} \cup \{B\}) \land \neg \mathcal{C} \), but also from another rule in \( P \) (and \( P' \)). However, this case is even simpler: Formally, let us call this rule \( \psi \). Lemma 3.1 shows that the minimal models of \( P/I \) and \( P'/I \setminus \{\psi\} \) are identical. But since \( \psi \in P/I \), every minimal model of \( P'/I \setminus \{\psi\} \) already satisfies \( \psi \), so we do not change the set of minimal models by adding \( \psi \).

Lemma 3.3. STABLE allows the elimination of tautologies.

PROOF. A tautology remains a tautology after the GL-Transformation. But the minimal models depend only on the logical contents of the transformed program, so a tautology makes no difference. □

Lemma 3.4. STABLE allows the Elimination of Contradictions.

PROOF. Let \( P' = P - \{\mathcal{A} \leftarrow \mathcal{B} \land \neg \mathcal{C}\} \) with \( B \in \mathcal{B} \cap \mathcal{C} \), and let \( I \) be any interpretation. If \( I \not\models \neg \mathcal{C} \), we have \( P/I = P'/I \). So let \( I \models \neg \mathcal{C} \). Then we have
\[
P/I = P'/I \cup \{\mathcal{A} \leftarrow \mathcal{B}\}.
\]
Furthermore, we know that \( I \not\models B \):

- Suppose that \( I \) is a minimal model of \( P/I \). Then it is, of course, also a model of \( P'/I \). It is also minimal, since if there were a smaller model \( I' \), this model would satisfy \( I' \not\models B \), and thus would also be a model of \( P/I \).
3.2. Properties of Weak-Supp, SUPP, and \textit{comp}

Lemma 3.5 (GPPE). Let $P$ be an instantiated logic program and $P'$ be the result of applying GPPE to $P$, i.e., for some rule $\mathcal{A} \leftarrow (\mathcal{B} \cup \{B\}) \land \neg \mathcal{C}$ in $P$:

$$
P' := P - \{\mathcal{A} \leftarrow (\mathcal{B} \cup \{B\}) \land \neg \mathcal{C}\}
$$

$$
\cup \{\mathcal{A} \cup (\mathcal{A}' - \{B\}) \leftarrow (\mathcal{B} \cup \mathcal{B}') \land (\neg \mathcal{C} \lor \mathcal{C}') | \mathcal{A}' \leftarrow \mathcal{B}' \land \neg \mathcal{C}' \in P, B \in \mathcal{A}'\}.
$$

Then $I$ is a supported model of $P$ iff $I$ is a supported model of $P'$.

PROOF.

1. First we prove that if $I$ is a model of $P$, then it is a model of $P'$. Since $I \models P$, only a rule in $P' - P$ could possibly be violated, i.e., one of the rules

$$
\mathcal{A} \cup (\mathcal{A}' - \{B\}) \leftarrow (\mathcal{B} \cup \mathcal{B}') \land (\neg \mathcal{C} \lor \mathcal{C}')
$$

If this rule is violated, we especially have $I \models \mathcal{B} \land \neg \mathcal{C}'$ and $I \not\models \mathcal{A}' - \{B\}$. But since $I$ is a model of $\mathcal{A}' \leftarrow \mathcal{B}' \land \neg \mathcal{C}'$, it follows that $I \models \mathcal{B}$. But then we have $I \models (\mathcal{B} \cup \{B\}) \land \neg \mathcal{C}'$ and $I \not\models \mathcal{A}$, which means that $I$ violates the rule $\mathcal{A} \leftarrow (\mathcal{B} \cup \{B\}) \land \neg \mathcal{C}$. But this contradicts $I \models P$.

2. Next, we prove that a supported model of $P$ is also a supported model of $P'$. The only ground atoms $A$ which may have lost their support are those previously supported by the rule $\mathcal{A} \leftarrow (\mathcal{B} \cup \{B\}) \land \neg \mathcal{C}$. So we have $I \models (\mathcal{B} \cup \{B\}) \land \neg \mathcal{C}$ and $I \not\models \mathcal{A} - \{A\}$. Since $I \models \mathcal{B}$, there must be a rule $\mathcal{A}' \leftarrow \mathcal{B}' \land \neg \mathcal{C}'$ supporting $B$. This means that $I \models \mathcal{B} \land \neg \mathcal{C}'$ and $I \not\models \mathcal{A}' - \{B\}$. Therefore, the following rule also supports $A$:

$$
\mathcal{A} \cup (\mathcal{A}' - \{B\}) \leftarrow (\mathcal{B} \cup \mathcal{B}') \land (\neg \mathcal{C} \lor \mathcal{C}')
$$

3. Next, we prove that a supported model of $P'$ is also a model of $P$. If this were not the case, then $I$ must violate the rule $\mathcal{A} \leftarrow (\mathcal{B} \cup \{B\}) \land \neg \mathcal{C}$, i.e., $I \models (\mathcal{B} \cup \{B\}) \land \neg \mathcal{C}$ and $I \not\models \mathcal{A}$.

Let $\mathcal{A}' \leftarrow \mathcal{B}' \land \neg \mathcal{C}'$ be a rule supporting $B$. This rule is contained in $P$, because the rules in $P' - P$ contain $B$ only in their heads if $B \in \mathcal{A}$, in which case $\mathcal{A} \leftarrow (\mathcal{B} \cup \{B\}) \land \neg \mathcal{C}$ is a tautology and cannot be violated.

Since $\mathcal{A}' \leftarrow \mathcal{B}' \land \neg \mathcal{C}'$ supports $B$, we have $I \models \mathcal{B}' \land \neg \mathcal{C}'$ and $I \not\models \mathcal{A}' - \{B\}$. But this means that the following rule in $P'$ is violated, which is a contradiction:

$$
\mathcal{A} \cup (\mathcal{A}' - \{B\}) \leftarrow (\mathcal{B} \cup \mathcal{B}') \land (\neg \mathcal{C} \lor \mathcal{C}')
$$

4. Finally, we have to show that a supported model $I$ of $P'$ is also supported with respect to (wrt) $P$. So suppose that $A$ is supported by one of the rules in $P' - P$:

$$
\mathcal{A} \cup (\mathcal{A}' - \{B\}) \leftarrow (\mathcal{B} \cup \mathcal{B}') \land (\neg \mathcal{C} \lor \mathcal{C}')
$$
We consider two cases:

- Suppose first that \( I \models B \). Then \( I \models \mathcal{A} \) (we have already proven that \( I \models P \), so especially \( I \) satisfies \( \mathcal{A} \leftarrow (\mathcal{B} \cup \{B\}) \land \neg C' \). Since \( A \) is the only true atom in \( \mathcal{A} \cup (\mathcal{A}' - \{B\}) \), it follows that \( A \in \mathcal{A} \), and therefore \( A \) is supported by \( \mathcal{A} \leftarrow (\mathcal{B} \cup \{B\}) \land \neg C' \) in \( P \).

- Now let \( I \not\models B \). Since \( I \models \mathcal{A}' \land \neg C' \), and the rule \( \mathcal{A}' \leftarrow \mathcal{B}' \land \neg C' \) is contained in \( P' \), it follows that \( A \in \mathcal{A}' \) (otherwise \( I \not\models \mathcal{A}' \), because \( A \) is the only true atom in \( \mathcal{A}' \cup (\mathcal{A}' - \{B\}) \)). But since \( I \not\models B \), the rule \( \mathcal{A}' \leftarrow \mathcal{B}' \land \neg C' \) supports \( A \). \( \square \)

Example 3.2. Of course, the supported model semantics does not allow Elimination of Tautologies. For instance, \( I := \{p\} \) is a supported model of \( P' := \{p \rightarrow p\} \), but it is not a supported model of \( P := \emptyset \). However, a supported model of \( P' \) is always a supported model of \( P \). This example also applies, mutatis mutandis, to Weak-SUPP.

Lemma 3.6. SUPP and Weak-SUPP allow Elimination of Contradictions.

Proof. This is trivial, because the supporting condition requires a rule with a body which is satisfied in the supported model. Since an inconsistent body can never be satisfied, the rule can simply be deleted. \( \square \)

This last lemma also implies (with the observation that the supported models are exactly the models of comp) that Clark's completion comp satisfies GPPE.

Example 3.3. Weak-SUP does not satisfy GPPE. Consider the following logic program and its partial evaluation:

\[
A \leftarrow B, \quad A \lor C.
\]

\[
B \lor C, \quad B \lor C.
\]

Now consider \( I := \{B,C\} \). This is a weakly supported model of the resulting program, but it does not satisfy the first rule in the original program.

4. CHARACTERIZATIONS OF THE SEMANTICS

In this section, we give characterizations of the semantics Weak-SUPP, SUPP, GCWA, PERFECT, and STABLE in terms of our abstract properties. We begin with a useful lemma and note that Dung and Kanchansut in [17], Bry in [12], and Hu and Yuan in [23] also considered rules with only negative literals.

Lemma 4.1 (Normal form). Let \( \text{SEM} \) be a semantics satisfying GPPE and Elimination of Tautologies. Then any program \( P \) is \( \text{SEM} \)-equivalent to a program \( P' \) where all clauses have the form \( \mathcal{A} \leftarrow \neg C' \), i.e., there do not occur any positive atoms in the bodies. Moreover, if \( P \) is a positive program, then \( P' \) is a set of positive disjunctions (containing no body literals at all).

Proof. The proof is by induction on the number \( n \) of ground atoms occurring as positive body literals in \( P \). Nothing has to be proven in the case \( n = 0 \). So suppose that at least one ground atom \( A \) occurs as a positive body literal in \( P \). Then we
first eliminate all tautologies \( A \leftarrow B \land \neg C \) containing \( A \) both in \( A \) and \( B \). Second, we apply GPPE to all remaining occurrences of \( A \) as a positive body literal. Since the rules containing \( A \) in the head do not contain it in the positive part of the body, none of the resulting rules contains \( A \) as a positive body literal. Furthermore, the resulting rules contain only positive body literals which were already contained in \( P \) as positive body literals. Therefore, the number of atoms occurring as positive body literals decreased by at least one, and the inductive hypothesis is applicable. \( \Box \)

This lemma does not hold if we allow infinite instantiated programs or finite programs containing variables and function symbols. As an example, let us consider

\[
p(X) \leftarrow p(f(X))
\]

or, equivalently, the infinite propositional program

\[
P_0 \leftarrow P_1, P_1 \leftarrow P_2, \ldots, P_i \leftarrow P_{i+1}, \ldots
\]

GPPE and Elimination of Tautologies can only unfold finite loops, but this is an infinite one.

**Lemma 4.2.** An interpretation \( I \) is a supported model of a program \( P \) without positive body literals iff it is a stable model of \( P \).

In fact, for all programs, a stable model is supported, and the converse is true for all programs without positive body literals.

**Proof.**

1. Let \( I \) be a supported model of \( P \). Clearly, \( I \) is also a supported model of \( P/I \). But \( P/I \) is a set of disjunctive facts of the form \( A \leftarrow \text{true} \). Now the supportedness condition means that every true atom occurs in a disjunction where all other atoms are false. So no smaller model than \( I \) can still be a model of \( P/I \).

2. Let \( I \) be a stable model of \( P \), i.e., a minimal mode of \( P/I \). Let \( A \) be an atom true in \( I \) and \( I' := I - \{A\} \). So \( I' \) is smaller than \( I \) and must violate a rule \( A \leftarrow B \) in \( P/I \). This means especially \( I' \models B \), and therefore \( I \models B \). Furthermore, \( I \) satisfies also the negative body literals (because the rule is contained in \( P/I \)). Finally we have \( I' \neq A \), i.e., \( I \neq A \) \( \neq \{A\} \), so \( A \) is contained in \( A \), and this rule supports it. \( \Box \)

**Theorem 4.1** (Partial characterization of GCWA). Let \( SEM \) be a semantics satisfying GPPE and Elimination of Tautologies. Then \( SEM(P) \subseteq \text{Min-MOD}_{2\text{-val}}^\text{Herbrand}(P) \) for any positive disjunctive program \( P \), i.e., any such semantics is based on 2-valued minimal models for positive programs. In particular, GCWA is the weakest semantics with these properties.

**Proof.** We give an indirect but nevertheless constructive proof. Let \( I \in SEM(P) \) with \( I \not\in \text{Min-MOD}_{2\text{-val}}^\text{Herbrand}(P) \). We have to derive a contradiction. Due to Lemma 4.1, we can assume w.l.o.g. that \( P \) only consists of positive disjunctions. Let \( \bot \) be a new atom. We consider the program

\[
P' = P \cup \{ \bot \leftarrow \text{True}(I), \text{Undef}(I) \}.
\]
Now we only have to show

\[(\ast) \quad P' \text{ is SEM-equivalent to } P \ (i.e., \ SEM(P') = SEM(P)) ,\]

and we arrive at a contradiction because \(I[\bot] = f (I \in SEM(P) \text{ and } P \text{ does not contain } \bot) \) and therefore \(I\) cannot be a 3-valued model of \(\bot \leftarrow \text{True}(I), \text{Undef}(I)\).

Indeed, we will show \((\ast)\) by applying GPPE and Elimination of Tautologies to \(P'\); this allows us to eliminate the whole new rule in \(P'\).

**Case 1: Undef(I) \neq \emptyset.** Let \(A \in \text{Undef}(I)\). We apply GPPE in \(P'\) to replace \(A\), i.e., we get for any \(A \vee \alpha \in P\) a new rule \(\alpha\). But since \(I[A] = u\), and \(I\) is a model of \(A \vee \alpha\), \(\alpha\) contains another atom which evaluates to \(t\) in \(I\), and therefore \(\alpha\) can be eliminated using Elimination of Tautologies.

**Case 2: Undef(I) = \emptyset.** This means that \(I \in \text{MOD}_{\text{Herbrand}}^{2,\text{val}}(P)\) but \(I\) is not a minimal model (our general assumption). There is therefore an \(I' \models P\) with \(I' \prec I\). Thus there exists a atom \(Y\) with \(I'[Y] = f \neq t = I[Y]\). We apply GPPE to \(P'\) and replace \(Y\). For any \(Y \vee \alpha \in P\), \(I'[\alpha] = t\) (since \(I' \models P\)), and therefore also \(I[\alpha] = t\). So any of the disjunctions \(Y \vee \alpha\) contains an atom true in \(I\), and therefore all rules can be eliminated by Elimination of Tautologies.

The last theorem only tells us that any semantics satisfying our two conditions selects minimal models. It still leaves open the possibility to select a proper subset of them. To get the whole set of all minimal models, we have to add *Isomorphy* and *Relevance*.

**Theorem 4.2** (Complete characterization of GCWA). *Any nontrivial semantics satisfying GPPE, Elimination of Tautologies, Isomorphy, and Relevance coincides with GCWA on positive disjunctive programs.*

**Proof.** Let \(SEM\) be a semantics satisfying these conditions. By Lemma 4.1, it suffices to consider only programs \(P\) without body literals. The preceding theorem has shown that \(SEM(P) \subseteq \text{Min-MOD}_{2,\text{val}}(P)\). Now we have to show the converse. So let \(I \in \text{Min-MOD}_{2,\text{val}}(P)\).

In order to show that \(I \in SEM(P)\), we first transform \(P\) into a program \(P'\) by replacing false head literals by true body literals. More formally, we introduce a new atom \(X_A\) for every atom \(A\) which is false in \(I\). Then let

\[
P_0' := \{ A \vee X_A | A \text{ is false in } I \}.
\]

Now let \(P'_1\) contain for every disjunction \(\alpha \in P\) the rule

\[
\{ A \in \alpha | I \models A \} \leftarrow \{ X_A | A \in \alpha, I \not\models A \}.
\]

For instance, if \(p\) is true and \(q\) is false, we transform \(p \vee q \leftarrow\) into \(p \leftarrow X_q\). Let finally \(P' := P_0' \cup P'_1\).

Now consider the model \(I_0'\) of \(P_0'\) which makes all \(X_A\) true and all those \(A\) false that were already false in \(I\). It obviously is a minimal model of \(P_0'\). We will show that \(I_0' \in SEM(P_0')\). By Lemma 2.1, \(SEM\) is consistent, so there must be an \(I_0' \in SEM(P_0')\). By the preceding theorem, \(I_0'\) is a minimal model. But because of the simple structure of \(P_0'\), there is an isomorphism which transforms \(I_0'\) into \(I_0\). So our isomorphy condition yields \(I_0' \in SEM(P_0')\).

By Relevance, there must be an \(I' \in SEM(P')\) such that \(SEM(P')(I') = I_0'\). We will show that \(I'\) must make all atoms true which are true in \(I\) (and since it extends \(I_0\), it must also make all atoms false which are false in \(I\)). So let \(A\) be an atom with
$I \models A$. Because of the minimality of $I$, $P$ must contain a disjunction in which $A$ is the only true atom (otherwise we could make $A$ false and still have a model). But this means that $P'$ contains a rule

$$A \leftarrow X_{A_1} \land \cdots \land X_{A_n},$$

where all the body atoms are true. Therefore, $I' \models A$.

So $I' \in \text{SEM}(P')$ is an extension of $I$. We can now finally apply the second part of the Relevance condition to get $I = \text{SEM}'(I') \subseteq \text{SEM}(P)$. □

It is worth noting that we do not assume in the last theorem that SEM is based on two-valued models. We get this automatically from our conditions, although this does not follow for arbitrary programs (not even for nondisjunctive programs, where WFS is a counterexample, as can be seen from Theorem 3.1).

For the next theorems, we need the Elimination of Contradictions.

Lemma 4.3. Let SEM satisfy Elimination of Contradictions. Then $\text{SEM}(P) \subseteq \text{MOD}^\text{Herbrand}_{\text{2-val}}(P)$.

PROOF. Let $I \in \text{SEM}(P)$ and suppose that there were an atom $A$ with $I[A] = u$. Then we consider $P' = P \cup \{ \bot \leftarrow A, \neg A \}$. As in the proof of Theorem 4.1, it suffices to eliminate the new rule, because then $\bot$ is false in $P'$, which is a contradiction (every model of $P'$ makes $\bot$ at least undefined). But the rule can be eliminated just by applying Elimination of Contradictions. □

Lemma 4.4 (Partial characterization of Weak-SUPP). Let SEM be a semantics satisfying GPPE and Elimination of Contradictions. Then $\text{SEM}(P) \subseteq \text{Weak-SUPP}(P)$.

PROOF. Assume $I \notin \text{Weak-SUPP}(P)$, i.e., there is an atom $X$ with $I[X] = t$ and for all rules $X \lor \mathcal{A} \leftarrow \mathcal{B} \land \neg \mathcal{C} \in P$, $I[\mathcal{B} \land \neg \mathcal{C}] = f$. Let

$$P' = P \cup \{ \bot \leftarrow X, \text{True}(I), \neg \text{False}(I) \}.$$ We replace the first occurrence of $X$ by GPPE. Our assumption guarantees that all clauses generated by GPPE can be eliminated with Elimination of Contradictions (note that in every new clause, the whole set True($I$) still occurs in the body —here we need duplicate occurrence of $X$ in the body). Therefore, we are done (using the same reasoning as in the proofs of the preceding lemma and theorem). □

Obviously, Weak-SUPP is not the weakest semantics satisfying these principles, because GPPE does not hold. For nondisjunctive programs, Weak-SUPP and SUPP collapse and SUPP satisfies GPPE.

Corollary 4.1 (SUPP for nondisjunctive programs). Let SEM be a semantics for nondisjunctive programs satisfying GPPE and Elimination of Contradictions. Then $\text{SEM}(P) \subseteq \text{SUPP}(P)$. In particular, SUPP is the weakest semantics for nondisjunctive programs with these properties.
Theorem 4.3 (First partial characterization of STABLE). Let SEM be a semantics satisfying GPPE and Elimination of Tautologies. The following three conditions are equivalent:

1. \( \text{SEM}(P) \subseteq \text{Weak-SUPP}(P) \) for all \( P \).
2. \( \text{SEM}(P) \subseteq \text{SUPP}(P) \) for all \( P \).
3. \( \text{SEM}(P) \subseteq \text{STABLE}(P) \) for all \( P \).

**Proof.** Since "(2) implies (1)" is trivial, it suffices (in order to show equivalence between (1) and (2)) to derive a contradiction from the assumption that \( I \not\in \text{Weak-SUPP}(P) \) but \( I \not\in \text{SUPP}(P) \). Then there is an atom \( X \) with \( I[X] = t \), and for all clauses \( \varphi \leftarrow \delta \land 
eg \varepsilon \in P \), either \( I[\delta \land 
eg \varepsilon] = f \) or there is \( Y_x \in \varphi \) with \( I[Y_x] = t \). Let us assume that there are \( m \) clauses with \( X \) in their head, and that \( n \) of these satisfy \( I[\delta \land 
eg \varepsilon] = t \), i.e., they have the form

\[
X \lor Y_1 \lor \varphi_i \leftarrow \delta_i \land 
eg \varepsilon_i.
\]

W.l.o.g. we select an atom \( B_j \) (\( j = 1, \ldots, m - n \)) with \( I[B_j] = f \) from the body of any of the remaining rules, if there were only negative literals, we could replace them by introducing new positive atoms together with appropriate new rules (i.e., we apply GPPE). Here we need to make the assumption that SEM satisfies the special instance of Relevance introduced in Section 2.3. Let

\[
P' = P \cup \{ \bot \lor B_1 \lor \cdots \lor B_{m-n} \leftarrow X, \text{True}(I), \neg \text{False}(I) \}.
\]

We have to show that \( P' \) can be reduced to \( P \) using our abstract properties.

We first replace the \( X \) in \( \bot \lor B_1 \lor \cdots \lor B_{m-n} \leftarrow X, \text{True}(I), \neg \text{False}(I) \) using GPPE. We get \( m \) clauses and can immediately cancel those \( m - n \) of them (by Elimination of Tautologies) that contain a \( B_j \) both in the head and in the body. There remain for \( i = 1, \ldots, n \) the clauses

\[
\bot \lor B_1 \lor B \cdots \lor B_{m-n} \lor Y_1 \lor \varphi_i \leftarrow Y_1, \ldots, Y_n, \delta_i \land \neg \varepsilon_i, \text{True}(I), \neg \text{False}(I)
\]

that can also be cancelled since \( Y_i \) is contained both in the head and the body.

We now show that (2) implies (3). Let \( P' \) be a logic program without positive body literals according to Lemma 4.1. We then have \( \text{SEM}(P) = \text{SEM}(P') \). By Lemma 4.2, we know that \( \text{SUPP}(P') = \text{STABLE}(P') \). So we finally have

\[
\text{SEM}(P) = \text{SEM}(P') \subseteq \text{SUPP}(P') = \text{STABLE}(P') = \text{STABLE}(P).
\]

The implication (3) \( \Rightarrow \) (2) is trivial. \( \Box \)

(2) of Theorem 4.3 shows that STABLE is the weakest semantics selecting only supported models and satisfying GPPE and Elimination of Tautologies. Since SUPP itself satisfies GPPE, this shows that the Elimination of Tautologies is, in a strict sense, the only difference between STABLE and SUPP.

(1) of Theorem 4.3 shows that STABLE is the weakest semantics selecting only weakly supported models and satisfying GPPE, Elimination of Tautologies (and the special case of Relevance). Note: the assumption that an atom has only a weak support is indeed a very weak assumption.

\[2\] In (1) implies (2), we need the instance of Relevance introduced after Definition 2.11.
Theorem 4.4 (Second partial characterization of STABLE). Let SEM be a semantics satisfying GPPE, Elimination of Tautologies, and Elimination of Contradictions. Then SEM(P) ⊆ STABLE(P).

Proof. Let P' be a logic program without positive body literals according to Lemma 4.1. We then have SEM(P) = SEM(P') and STABLE(P) = STABLE(P'), so it suffices to show SEM(P') ⊆ STABLE(P').

Now let \( I \in SEM(P') \), and let \( \bot \in \Sigma \) be a ground atom with \( I \not\models \bot \). Because of Definition 3, \( I \models \neg \bot \). We have to show that \( I \models STABLE(P') \). Suppose this would not be the case, i.e., \( I \) would not be a minimal model of \( P'/I \). Let \( I_0 \) be a smaller model of \( P'/I \), and \( A \) be a ground atom with \( I \models A \) and \( I_0 \not\models A \).

Let

\[
P'' := P' \cup \{ \bot \leftarrow A, True(I), \neg False(I) \}.
\]

Obviously, \( I \not\models P'' \), so \( I \not\in SEM(P'') \). However, we will show that our transformations allow to derive \( P' \) from \( P'' \), so SEM(P'') = SEM(P') must hold, which contradicts \( I \in SEM(P') \).

In order to derive \( P' \) from \( P'' \), we first apply GPPE to the atom \( \bot \) in the body of the new rule. We now show that the resulting rules are either tautologies or have an inconsistent body. Consider a rule \( \mathcal{A}' \leftarrow \neg \mathcal{A}' \) in \( P'' \), such that \( I \models \mathcal{A}' \):

- If \( I \not\models \neg \mathcal{A}' \), i.e., \( I \models B \) for some \( B \in \mathcal{A}' \), then the body of the resulting rule is inconsistent because of \( B \models True(I) \).
- Else (\( I \models \neg \mathcal{A}' \)), we can conclude that \( \mathcal{A}' \in P'/I \). Since \( I_0 \models \mathcal{A}' \) and \( I_0 \not\models A \), there is an atom \( B \in \mathcal{A}' B \neq A \) with \( I_0 \models B \), and therefore also \( I \models B \). But then the rule resulting from the application of GPPE contains \( B \) both in the head and in the positive part of the body, so it is a tautology. \( \square \)

By Theorem 4.4, STABLE is the weakest semantics satisfying GPPE, Elimination of Tautologies, and Elimination of Contradictions. In particular, if \( P \) is stratified, then SEM(P) consists of perfect models. For nondisjunctive programs, we immediately get the following.

Corollary 4.2 (\( M_{upp}^{\text{supp}} \) for nondisjunctive programs). Let SEM be a semantics for nondisjunctive programs satisfying GPPE, Elimination of Tautologies, and Elimination of Contradictions. Then SEM(P) ⊆ \( M_{upp}^{\text{supp}} \) for all stratified programs \( P \). In particular, if SEM(P) ∉ 0, then SEM(P) = \( M_{upp} \).

Our next result is an impossibility result. It is well known that STABLE is not always consistent, i.e., it is possible that STABLE(P) = 0. But even Weak-SUPP is already inconsistent for programs of the form \( p \leftarrow \neg p \). Since we have proven SEM(P) ⊆ Weak-SUPP(P), this also applies to any semantics SEM with the above properties. Thus we have the following.

Corollary 4.3 (Impossibility result). Let SEM be a semantics defined for all disjunctive logic programs satisfying GPPE, Elimination of Contradictions, and Independence. Then SEM is trivial, i.e., SEM(P) = 0 for all programs \( P \).

The argument is simple. Choose \( P' := \{ A \leftarrow \neg A \} \) with a new predicate \( A \). Then by Lemma 4.4,

\[
SEM(P \cup P') \subseteq Weak-SUPP(P \cup P') = 0.
\]
So any formula $Q$ would follow from $\text{SEM}(P \cup P')$, and therefore also from $\text{SEM}(P)$, for any logic program $P$.

The failure of Independence is a weakness of both Weak-SUPP and the stable semantics. We already introduced in Section 2.2 a principle strongly related to this: \textit{Relevance} (see also [19, 16]). Using our framework, it is possible to show the following.

\textbf{Theorem 4.5 (Complete characterization of PERFECT).} Any nontrivial semantics \textit{SEM} satisfying $\text{SEM} \subseteq \text{Weak-SUPP}$, \textit{GPPE}, \textit{Elimination of Tautologies}, \textit{Isomorphy}, and \textit{Relevance} already coincides with \textit{PERFECT} on stratified disjunctive programs.

\textbf{Proof.} The proof is very similar to the proof of Theorem 4.2 and proceeds by induction along the stratification. We do not need the normal form here (in fact, the normal form could destroy the stratification). Let $P = P_i \cup \cdots \cup P_0$ be a stratification of $P$. We assume that $P_i$ is positive (it could even be empty with respect to a language containing some atoms). Let $P_{\leq k-1} := \bigcup_{i=0}^{k-1} P_i$ and $P_{\leq k} = P_k \cup P_{\leq k-1}$. We prove by induction on $k$:

\begin{enumerate}
  \item There is a nondeterministic transformation $P_{\leq k} \leadsto P_{\leq k}^*$ with:
  \begin{enumerate}
    \item For any $I_k \in \text{PERFECT}(P_{\leq k})$, there is a $P_{\leq k}^*$ such that there is exactly one $\mathcal{M}_{\leq k}^* \in \text{PERFECT}(P_{\leq k})$ with $\mathcal{R} \mathcal{D}_{P_{\leq k}}(\mathcal{M}_{\leq k}^*) = I_k$. In addition, for all $X_A$ introduced during the transformation $P_{\leq k} \leadsto P_{\leq k}^*$, we have $\mathcal{M}_{\leq k}^* \models X_A$.
    \item If $\mathcal{M}^{*} \in \text{SEM}(P_{\leq k}^*)$ and for all $X_A$ introduced via $P_{\leq k} \leadsto P_{\leq k}^*$ we have $\mathcal{M}^{*} \models X_A$, then $\mathcal{M}^{*} \models X_A$.
    \item $\text{SEM}(P_{\leq k}) = \text{PERFECT}(P_{\leq k})$.
  \end{enumerate}

For $k = 0$, we only need to apply Theorem 4.2.

To prove the induction step $k \rightarrow k + 1$, let $I_{k+1} \in \text{PERFECT}(P_{\leq k+1})$. By Relevance and the Induction hypothesis, there is an $I_k \in \text{PERFECT}(P_{\leq k}) = \text{SEM}(P_{\leq k})$ with $\mathcal{R} \mathcal{D}_{P_{\leq k}}(I_{k+1}) = I_k$. By the Induction hypothesis, there is $P_{\leq k}^*$ with the above properties, i.e., there is exactly one $\mathcal{M}_{\leq k}^* \in \text{PERFECT}(P_{\leq k}^*)$ with $\mathcal{R} \mathcal{D}_{P_{\leq k}}(\mathcal{M}_{\leq k}^*) = I_k$. We define

$$P_{\leq k+1}^* := P_{k+1}^* \cup P_{\leq k}^*,$$

where $P_{k+1}^*$ is constructed as follows. For any atom $A$ occurring in $P_{k+1}$ but not in $P_k$, if $A \lor \mathcal{A} \leftarrow \mathcal{B} \land \neg \mathcal{C} \in P_{k+1}$ and $I_{k+1} \models \neg A$, then we replace all rules $A \lor \mathcal{A} \leftarrow \mathcal{B} \land \neg \mathcal{C}$ from $P_{k+1}$ with $\mathcal{A} \neq \emptyset$ by $\mathcal{A} \leftarrow \mathcal{B} \land \neg \mathcal{C}$, $X_A$ and $X_A \lor A$. The important property of this transformation is that all atoms occurring in nontrivial heads of $P_{k+1}^*$ are already true in $\mathcal{M}_{\leq k}^*$. Therefore, the truth of atoms occurring in trivial heads (heads that consist of only one atom) are uniquely determined by the truth of their bodies. But the body literals consist of those already decided by $\mathcal{M}_{\leq k}^*$ and the new $X_A$. Therefore, there is only one extension $\mathcal{M}_{k+1}^*$ of $\mathcal{M}_{\leq k}^*$ to a model of $P_{k+1}^*$ with $\mathcal{M}_{k+1}^* \models X_A$ for all $X_A$ introduced during $P_{\leq k+1} \leadsto P_{\leq k+1}^*$. Obviously, by construction, $\mathcal{R} \mathcal{D}_{P_{\leq k}}(\mathcal{M}_{k+1}^*) = I_{k+1}$ and we are done with (1).

(2) follows immediately from the Induction hypothesis and Relevance (the second condition in Definition 2.11).
characterizations of the stable semantics

(3) is the combination of (1) and (2): given an \( I_{k+1} \in \text{PERFECT}(P_{\leq k+1}) \), we get, using (1), an \( M^*_{k+1} \) with \( R \circ D_{P_{\leq k+1}}(M^*_{k+1}) = I_{k+1} \). Since the assumption of (2) are satisfied, we have

\[
I_{k+1} = R \circ D_{P_{\leq k+1}}(M^*_{k+1}) \in \text{SEM}(P_{\leq k+1}). \quad \square
\]

The question arises whether there exist semantics satisfying our properties on classes of programs that significantly extend the class of stratified (or locally stratified) programs. The general feeling is no—the proof of such a statement, however, is not trivial. This is because a given program might be nonstratified, but yet equivalent (using some of our transformations) to a stratified one. For example, although the program \( P_{ns}: A \leftarrow, A \leftarrow, A \rightarrow, \overline{A} \rightarrow, A \rightarrow \overline{A} \) is nonstratified, it is certainly equivalent to “\( A \leftarrow \)” for any reasonable semantics and thus its intended model should be \( \{A\} \).

We can eliminate such nongenuine cycles through negation by using our work in [7], where we have associated to any program \( P \) a certain normal form \( \text{res}(P) \), the residual program. \( \text{res}(P) \) is obtained from \( P \) by using our transformations and some weak reductions (if there is a clause \( A \leftarrow \), then any occurrence of \( A \) (resp., \( \overline{A} \)) can be replaced by true (resp., false)). For the above program \( P_{ns} \), we get

\[
\text{res}(P_{ns}) = A \leftarrow.
\]

We can also formally define the class of all programs that possess a stable model. This class extends the class of stratified programs, but \text{STABLE} is not relevant on this class (see [14]). It is easy to see that \text{STABLE} on this class satisfies Independence—thus Relevance is strictly stronger. Recently, however, Yuan noted that the stable semantics on the smaller class of all programs without an odd number of negative edges through negation satisfies Relevance.

We believe that this class cannot only be extended using our idea of deleting nongenuine cycles through negation, but it also represents the maximal such class. More formally, we have the following.

Conjecture 4.1 (No semantics for genuine nonstratifed programs). Let \( \text{SEM} \) be a nontrivial semantics satisfying GPPE, Elimination of Tautologies, Elimination of Contradictions, and Relevance. Then \( \text{SEM} \) is only defined on the class of all programs \( P \) such that \( \text{res}(P) \) contains no cycles with an odd number of negative edges. There is no semantics beyond this class.

Note that if we cancel the Elimination of Contradictions, then there are semantics defined on the whole class of programs, e.g., the static semantics of Przymusinski ([27]) or the D-WFS introduced by the authors ([4, 7]).

We believe our conjecture to be true both for disjunctive and nondisjunctive programs. In the latter case, already the well-founded semantics WFS satisfies all properties except Elimination of Contradictions.

We think that the last two results and our conjecture show us that if we leave the class of stratified disjunctive programs, then a semantics should be based on three-valued models, i.e., Elimination of Contradictions should be given up.

5. conclusions

In this paper, we have shown that partial evaluation is an interesting property. It not only holds for various semantics, but it also characterizes these semantics together with some other weak transformation conditions.
GPPE is a powerful principle. Together with Elimination of Tautologies, it enables us to define a normal form of a program (Lemma 4.1). Both properties are sufficient to ensure that a semantics only selects minimal two-valued models for positive disjunctive programs (Theorem 4.1). Together with Elimination of Contradictions (resp., the assumption that a semantics is based on weakly supported models), we can partially characterize the disjunctive stable semantics (Theorem 4.4, resp., Theorem 4.3).

We were also able to characterize GCWA (on positive disjunctive programs) and STABLE on stratified disjunctive programs completely, by simply adding Isomorphy and Relevance (Theorems 4.2 and 4.5).

It is interesting that for programs in normal form, supported models coincide with stable models. This fact can be used to compute these semantics (see [5] for more details).

Our impossibility result tells us that a reasonable semantics for the class of all disjunctive programs should not satisfy Elimination of Contradictions, i.e., should be based on three-valued models.

Finally, our conjecture formally states that there are no semantics (besides the trivial one) having our properties on nontrivial extensions of the class of all stratified programs.

Abstract properties like the ones considered in this article help to understand semantics and their underlying principles much better than the original fixpoint definitions. In [8, 10], a rigorous description of STABLE, WFS, and their disjunctive counterparts based on certain confluent calculi of transformations is given. Finally, [21] extends these transformations to programs with variables by using constraint logic programming techniques.

We are indebted to Li-Yan Yuan, F. Miguel Dionisio, and to two anonymous referees for helpful comments on a draft of this paper.

REFERENCES


