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 2-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 2-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. ◁

1. INTRODUCTION

The generalized closed world assumption GCWA for positive disjunctive programs (introduced in [Min82]), the perfect semantics PERFECT for stratified programs (introduced in [Prz88]) and the stable semantics STABLE for the class of all disjunctive programs (introduced in [GL91, Prz91]) are the most prominent semantics

Address correspondence to Jürgen Dix, University of Koblenz, Dept. of Computer Science, Rheinau 1, D-56075 Koblenz, Germany, e-mail: dix@mailhost.uni-koblenz.de and Stefan Brass, University of Hannover, Institut für Informatik, Lange Laube 22, D-30159 Hannover, Germany, e-mail: sb@informatik.uni-hannover.de

This is a revised and extended version of the papers [BD94a] presented at the 10th Workshop of Logic Programming in Zürich, Switzerland, Oct. 6–8, 1994, and of [BD95b] presented at LPNMR '95, Lexington, Kentucky, June 26–28, 1995.

THE JOURNAL OF LOGIC PROGRAMMING

655 Avenue of the Americas, New York, NY 10010 0743-1066/94/$7.00

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. This has been shown in [BD94b] and independently in [SS95]. 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 2-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 $P \mapsto P'$ 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: $\mathrm{SEM}(P) = \mathrm{SEM}(P')$.

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 non-disjunctive 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 *G*eneralized *P*rinciple of *P*artial *E*valuation (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 [ABW88]. 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 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 [Dix92, Dix95a, Dix95b] for normal (i.e. non-disjunctive), and in [DM94a] for disjunctive programs. Here we build on and use some results of the recent work reported in [BD95c]. In the companion paper [BD97] 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" behaviour of a semantics (Subsection 2.3).

2.1. Semantics

We consider disjunctive programs over some fixed signature Σ. We assume that Σ 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

$$A_1 \vee \cdots \vee A_k \leftarrow B_1 \wedge \cdots \wedge B_n \wedge \neg C_1 \wedge \cdots \wedge \neg C_n,$$

where the $A_i/B_i/C_i$ are Σ-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 $\mathcal{A} \leftarrow \mathcal{B} \wedge \neg\mathcal{C}$.

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 ($\mathbf{f}$), 0 with undefined ($\mathbf{u}$), and 1 with true ($\mathbf{t}$).

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 iff for every rule

$$A_1 \vee \cdots \vee A_k \leftarrow B_1 \wedge \cdots \wedge B_n \wedge \neg C_1 \wedge \cdots \wedge \neg C_n,$$

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 can not be false — $I[\![\mathit{head}]\!]$ must be either true or undefined.

Definition 2.3. (Semantics SEM)
A semantics SEM is a mapping from a class of logic programs P over Σ into the set of three-valued Herbrand models

$$\mathrm{SEM}(P) \subseteq \mathrm{MOD}^{Herbrand}_{3-val}(P),$$

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

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 $P \cup \{A \leftarrow \mathit{Body}\}$ where A is a new atom. In particular, they hold for the classes of *positive disjunctive*, *general disjunctive*, *positive non-disjunctive*, *stratified non-disjunctive* and *general non-disjunctive* 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 $\mathcal{A} \leftarrow \mathcal{B} \wedge \neg\mathcal{C}$ in P with $A \in \mathcal{A}$ and $I \models \mathcal{B} \wedge \neg\mathcal{C}$,

b) *supported* iff for every ground atom A with $I \models A$ there is a rule $\mathcal{A} \leftarrow \mathcal{B} \wedge \neg\mathcal{C}$ in P with $A \in \mathcal{A}$, $I \models \mathcal{B} \wedge \neg\mathcal{C}$, and $I \not\models \mathcal{A} - \{A\}$.

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

   Here P/I is the GL-Transform of P 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 SUPP(P), Weak-SUPP(P), and 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:

$$\mathrm{GCWA}(P) := \text{Min-MOD}_{2-val}^{Herbrand}(P).$$

While STABLE extends GCWA (in the sense that both coincide for positive disjunctive programs) this is neither true for Weak-SUPP nor for SUPP. It is well-known that PERFECT (introduced by Przymusinski in [Prz88]) coincides for stratified programs with STABLE (the stable semantics for disjunctive programs has been introduced independently by Gelfond and Lifschitz in [GL91] and by Przymusinski in [Prz91]) 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 $\mapsto$ is an arbitrary binary relation on the class of all programs.

We call it a SEM-equivalence transformation iff $\mathrm{SEM}(P_1) = \mathrm{SEM}(P_2)$ for all programs P_1, P_2 with $P_1 \mapsto P_2$.

The following two transformations allow us to *eliminate* certain rules. They are special cases of the D-reduction introduced in [DM94a].

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 $\mathcal{A} \leftarrow \mathcal{B} \wedge \neg\mathcal{C}$ 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 $\mathcal{A} \leftarrow \mathcal{B} \wedge \neg\mathcal{C}$ with $\mathcal{B} \cap \mathcal{C} \neq \emptyset$.

The most important transformation, however, is partial evaluation in the sense of the "unfolding" operation (see [DM94b, BD94b, SS95]):

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

- Suppose that the atom B occurs in the heads of the $n \geq 0$ rules

$$\mathcal{A}_i \leftarrow \mathcal{B}_i \wedge \neg\mathcal{C}_i, \qquad (i = 1, \ldots, n).$$

  Then replace a rule of the form

$$\mathcal{A} \leftarrow \big(\mathcal{B} \cup \{B\}\big) \wedge \neg\mathcal{C},$$

  by the n rules

$$\big(\mathcal{A} \cup (\mathcal{A}_i - \{B\})\big) \;\leftarrow\; (\mathcal{B} \cup \mathcal{B}_i) \;\wedge\; \neg(\mathcal{C} \cup \mathcal{C}_i), \qquad (i = 1, \ldots, n).$$

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]

$$\begin{array}{lll} P: & A \vee B \leftarrow C, & \neg C \\ & A \vee C \leftarrow B, E & \\ & E \vee B \leftarrow & \neg B \end{array}$$

[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 $P := \{q \leftarrow p,\ p \vee p\}$ shows, it is important for Lemma 3.1 that we remove all duplicate occurrences of B from the heads $\mathcal{A}_i$.

By *Elimination of Contradictions* we can eliminate the first rule. We then apply *GPPE* to replace the occurrence of B in the second rule. We get

$$\begin{array}{lcl} P' : A \vee C \vee E & \leftarrow & E, \neg B \\ \quad E \vee B & \leftarrow & \neg B \end{array}$$

and can eliminate the first rule by *Elimination of Tautologies* and end up with the single rule $E \vee B \leftarrow \neg B$.

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 ([DM94b]):

Definition 2.9. (Consistency)
A semantics SEM satisfies *Consistency* iff $\mathrm{SEM}(P) \neq \emptyset$ for all programs P.
We call a semantics *trivial* iff $\mathrm{SEM}(P) = \emptyset$ for all P.

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

$$\mathrm{SEM}(P_1) \models Q \iff \mathrm{SEM}(P_1 \cup P_2) \models Q,$$

provided that the predicates occurring in P_1 and P_2 are disjoint, and the query Q contains only 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. $P = P_1 \dot{\cup} P_2$ 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. *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 [Dix92] (see also [DM94b]). 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 reflexive and transitive closure of *depends immediately on*. The *dependencies of* and the *rules relevant for* a set M of atoms are now defined by:

- $\mathit{dependencies\_of}(M)$ is the set of atoms A such that there is $B \in M$ which depends on A.

- $\mathit{rel\_rul}(P, M)$ is the set of *relevant rules* of P with respect to M, i.e. the set of rules that contain an $A \in \mathit{dependencies\_of}(M)$ 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 "$\mathcal{RED}_{P'}(I)$" of a "*reduct* of an interpretation I in SEM(P) to a model of P'":

$$\mathcal{RED}_{P'}(I)[\![A]\!] := \begin{cases} I[\![A]\!] & \text{if } A \text{ occurs in } P', \\ \mathbf{f} & \text{otherwise.} \end{cases}$$

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. $\mathrm{SEM}\big(\mathit{rel\_rul}(P, M)\big) = \mathcal{RED}_{\mathit{rel\_rul}(P,M)}\big(\mathrm{SEM}(P)\big)$, i.e. $\mathrm{SEM}\big(\mathit{rel\_rul}(P, M)\big)$ consists exactly of the reducts of SEM(P) to $\mathit{rel\_rul}(P, M)$, and

2. If $I \in \mathrm{SEM}\big(P \cup \{X_{new} \vee A\}\big)$ (where X_{new} is a new atom not occurring in P) and $I[\![\neg A]\!] = \mathbf{t}$, then $\mathcal{RED}_P(I) \in \mathrm{SEM}(P)$.

The second condition simply states that the reduct of an intended model I for $P \cup \{X_{new} \vee A\}$ should also be an intended model for P. Adding a rule $X_{new} \vee A$ to P makes it possible for A to become true. But our special assumption that $I[\![\neg A]\!] = \mathbf{t}$ 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 need later is the following. Let A occur in P and X_{new} be a new atom not occurring in P. Then

$$\mathcal{RED}_P\big(\mathrm{SEM}(P \cup \{X_{new} \leftarrow \neg A\})\big) = \mathrm{SEM}(P).$$

This is a technical property needed to prove "(1)$\Longrightarrow$ (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 "$A \vee B$". This is strange because the program is completely symmetric in A and B and therefore if $\{A\} \in \mathrm{SEM}\big(\{A \vee B\}\big)$ then also $\{B\} \in \mathrm{SEM}\big(\{A \vee B\}\big)$ should hold. It turns out later (Theorems 4.2 and 4.5) that the following property is indeed sufficient to exclude such anomalous behaviour:

Definition 2.12. (Isomorphy)
A semantics SEM satisfies *Isomorphy*, iff

$$\mathrm{SEM}\big(\mathcal{I}(P)\big) = \mathcal{I}\big(\mathrm{SEM}(P)\big)$$

for all programs P and isomorphisms $\mathcal{I}$ on the set of all Σ-ground atoms.

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

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

Lemma 2.1. (Relevance $\Longrightarrow$ Independence $\Longrightarrow$ Consistency)

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

b) Let SEM be a non-trivial 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 non-trivial semantics satisfying *Isomorphy* and *Independence*. Let P be given. We have to prove $\mathrm{SEM}(P) \neq \emptyset$. The non-triviality implies that there is a P' with $\mathrm{SEM}(P') \neq \emptyset$. Using *Isomorphy* we can assume w. l. o. g. that P and P' have no common symbols. We can now apply *Independence* and get

$$\mathrm{SEM}(P) \models \mathbf{f} \iff \mathrm{SEM}(P \cup P') \models \mathbf{f} \iff \mathrm{SEM}(P') \models \mathbf{f}$$

which contradicts $\mathrm{SEM}(P') \neq \emptyset$. $\square$

3. PROPERTIES OF THE SEMANTICS

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

We note that M_P^{supp}, the perfect Herbrand model introduced in [ABW88], shares all our properties. While Clark's completion comp ([Cla78]) 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 [DM94b, DM94c]). 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 non-stratified 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 [SS95].

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.

Properties of Logic-Programming Semantics							
Semantics	*Taut.*	*Contr.*	*GPPE*	*Cons.*	*Indep.*	*Relev.*	*Isom.*
M_P^{supp}	•	•	•	•	•	•	•
comp	—	•	•	—	—	—	•
WFS	•	—	•	•	•	•	•
GCWA	•	•	•	•	•	•	•
PERFECT	•	•	(•)	•	•	•	•
Weak-SUPP	—	•	—	—	—	—	•
SUPP	—	•	•	—	—	—	•
STABLE	•	•	•	—	—	—	•
STATIC	•	—	•	•	•	•	•
D-WFS	•	—	•	•	•	•	•

Note that the first three semantics are only defined for non-disjunctive programs (M_P^{supp} only for stratified ones). GCWA ([Min82]) is defined for positive disjunctive and PERFECT ([Prz88]) for stratified disjunctive programs. The remaining semantics are defined for the whole class of all disjunctive programs. The semantics STATIC ([Prz95, BDP96]) and D-WFS ([BD94b, BD95c]) are listed only for completeness. Their properties are considered in [BD95a, BD97].

That WFS and M_P^{supp} satisfy GPPE has been proved in [Sek91, Sek93] and [AD94, AD95]. WFS does not eliminate contradictions as shown in the next example.

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 derives $\neg B$.

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:

$$\begin{aligned} P' := P &- \big\{\mathcal{A} \leftarrow \mathcal{B} \cup \{B\}\big\} \\ &\cup \big\{\mathcal{A} \cup (\mathcal{A}' - \{B\}) \leftarrow (\mathcal{B} \cup \mathcal{B}') \bigm| \mathcal{A}' \leftarrow \mathcal{B}' \ \in \ P \text{ with } B \in \mathcal{A}'\big\}. \end{aligned}$$

Then P and P' have the same minimal models.

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 $I \models \mathcal{B}$, $I \models \mathcal{B}'$, $I \not\models \mathcal{A}$, $I \not\models \mathcal{A}' - \{B\}$. 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 $I \models \mathcal{A} \leftarrow \mathcal{B} \cup \{B\}$ (for the same reason) yields $I \models \mathcal{A}$, 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 $I \not\models \mathcal{A}' - \{B\}$. 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 were also a minimal model I_0 with $I_0 \preceq I_1$, 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

$$\mathcal{A} \leftarrow (\mathcal{B} \cup \{B\}) \wedge \neg\mathcal{C}$$

and let P' be the result of applying GPPE to the body literal B. We have to show for any Herbrand interpretation I:

I is a stable model of P iff I 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

   $$\mathcal{A} \vee (\mathcal{A}' - \{B\}) \leftarrow \mathcal{B} \wedge \mathcal{B}' \wedge \neg(\mathcal{C} \wedge \mathcal{C}')$$

   be a rule resulting from the application of GPPE to P, namely by inserting

   $$\mathcal{A}' \leftarrow \mathcal{B}' \wedge \neg\mathcal{C}'$$

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 $\mathcal{A} \leftarrow \big(\mathcal{B} \cup \{B\}\big)$ is still contained in P'/I if it results not only from $\mathcal{A} \leftarrow \big(\mathcal{B} \cup \{B\}\big) \wedge \neg\mathcal{C}$, but also from another rule in P (and P'). However, this case is even simpler: Formally, let us call this rule ψ. Lemma 3.1 shows that the minimal models of P/I and $P'/I - \{\psi\}$ are identical. But since $\psi \in P/I$, every minimal model of $P'/I - \{\psi\}$ already satisfies ψ, so we do not change the set of minimal models by adding ψ. □

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 $P' = P - \{\mathcal{A} \leftarrow \mathcal{B} \wedge \neg\mathcal{C}\}$ 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.

- Suppose now that I is a minimal model of P'/I. Since $I \not\models B$, it is also a model of P/I. It is also minimal, since any smaller model would be also a model of P'/I. □

3.2. Properties of Weak-Supp, SUPP and 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 $\mathcal{A} \leftarrow \big(\mathcal{B} \cup \{B\}\big) \wedge \neg\mathcal{C}$ in P:

$$\begin{aligned} P' := P \;-\; & \big\{\mathcal{A} \leftarrow \big(\mathcal{B} \cup \{B\}\big) \wedge \neg\mathcal{C}\big\} \\ \cup\; & \big\{\mathcal{A} \cup \big(\mathcal{A}' - \{B\}\big) \leftarrow (\mathcal{B} \cup \mathcal{B}') \wedge \neg(\mathcal{C} \cup \mathcal{C}' \;\big|\; \\ & \qquad\qquad \mathcal{A}' \leftarrow \mathcal{B}' \wedge \neg\mathcal{C}' \;\in\; P,\; B \in \mathcal{A}'\big\}. \end{aligned}$$

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

$$\mathcal{A} \cup \big(\mathcal{A}' - \{B\}\big) \leftarrow \big(\mathcal{B} \cup \mathcal{B}'\big) \wedge \neg\big(\mathcal{C} \cup \mathcal{C}'\big).$$

If this rule is violated, we especially have $I \models \mathcal{B}' \wedge \neg\mathcal{C}'$ and $I \not\models \mathcal{A}' - \{B\}$. But since I is a model of $\mathcal{A}' \leftarrow \mathcal{B}' \wedge \neg\mathcal{C}'$, it follows that $I \models B$. But then we have $I \models \big(\mathcal{B} \cup \{B\}\big) \wedge \neg\mathcal{C}$ and $I \not\models \mathcal{A}$, which means that I violates the rule $\mathcal{A} \leftarrow \big(\mathcal{B} \cup \{B\}\big) \wedge \neg\mathcal{C}$. 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 $\mathcal{A} \leftarrow \big(\mathcal{B} \cup \{B\}\big) \wedge \neg\mathcal{C}$. So we have $I \models \big(\mathcal{B} \cup \{B\}\big) \wedge \neg\mathcal{C}$ and $I \not\models \mathcal{A} - \{A\}$. Since $I \models B$, there must be a rule $\mathcal{A}' \leftarrow \mathcal{B}' \wedge \neg\mathcal{C}'$ supporting B. This means that $I \models \mathcal{B}' \wedge \neg\mathcal{C}'$ and $I \not\models \mathcal{A}' - \{B\}$. Therefore, the following rule also supports A:

$$\mathcal{A} \cup \big(\mathcal{A}' - \{B\}\big) \leftarrow \big(\mathcal{B} \cup \mathcal{B}'\big) \wedge \neg\big(\mathcal{C} \cup \mathcal{C}'\big).$$

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 $\mathcal{A} \leftarrow \big(\mathcal{B} \cup \{B\}\big) \wedge \neg\mathcal{C}$, i.e. $I \models \big(\mathcal{B} \cup \{B\}\big) \wedge \neg\mathcal{C}$ and $I \not\models \mathcal{A}$.

   Let $\mathcal{A}' \leftarrow \mathcal{B}' \wedge \neg\mathcal{C}'$ 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 $\mathcal{A} \leftarrow \big(\mathcal{B} \cup \{B\}\big) \wedge \neg\mathcal{C}$ is a tautology and cannot be violated.

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

$$\mathcal{A} \cup \big(\mathcal{A}' - \{B\}\big) \leftarrow \big(\mathcal{B} \cup \mathcal{B}'\big) \wedge \neg\big(\mathcal{C} \cup \mathcal{C}'\big).$$

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

$$\mathcal{A} \cup \big(\mathcal{A}' - \{B\}\big) \leftarrow \big(\mathcal{B} \cup \mathcal{B}'\big) \wedge \neg\big(\mathcal{C} \cup \mathcal{C}'\big).$$

   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 $\mathcal{A} \leftarrow \big(\mathcal{B} \cup \{B\}\big) \wedge \neg\mathcal{C}$). Since A is the only true atom in $\mathcal{A} \cup \big(\mathcal{A}' - \{B\}\big)$, it follows that $A \in \mathcal{A}$, and therefore A is supported by $\mathcal{A} \leftarrow \big(\mathcal{B} \cup \{B\}\big) \wedge \neg\mathcal{C}$ in P.
   - Now let $I \not\models B$. Since $I \models \mathcal{B}' \wedge \neg\mathcal{C}'$, and the rule $\mathcal{A}' \leftarrow \mathcal{B}' \wedge \neg\mathcal{C}'$ 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 $\mathcal{A} \cup \big(\mathcal{A}' - \{B\}\big)$). But since $I \not\models B$, the rule $\mathcal{A}' \leftarrow \mathcal{B}' \wedge \neg\mathcal{C}'$ supports A. □

Example 3.2. Of course, the supported model semantics does not allow *Elimination of Tautologies*. For instance, $I := \{p\}$ is a supported model of $P := \{p \leftarrow p\}$, 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. □

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-SUPP does not satisfy GPPE. Consider the following logic program and its partial evaluation:

$$\begin{array}{ll} A \leftarrow B. & A \vee C. \\ B \vee C. & B \vee C. \end{array}$$

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/KANCHANSUT in [DK89], BRY in [Bry90], and HU/YUAN in [HY91] also considered rules with only negative literals.

Lemma 4.1. (Normal Form)
Let SEM be a semantics satisfying GPPE *and* Elimination of Tautologies.
Then any program P is SEM-equivalent to a program P' where all clauses have the form $\mathcal{A} \leftarrow \neg\mathcal{C}$, 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 $\mathcal{A} \leftarrow \mathcal{B} \wedge \neg\mathcal{C}$ containing A both in $\mathcal{A}$ and $\mathcal{B}$. 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. □

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

$$p(X) \leftarrow p\big(f(X)\big)$$

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 $\mathcal{A} \leftarrow \mathit{true}$. 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 model 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 $\mathcal{A} \leftarrow \mathcal{B}$ in P/I. This means especially $I' \models \mathcal{B}$, and therefore $I \models \mathcal{B}$. Furthermore, I satisfies also the negative body literals (because the rule is contained in P/I). Finally we have $I' \not\models \mathcal{A}$, i.e. $I \not\models \mathcal{A} - \{A\}$, so A is contained in $\mathcal{A}$, and this rule supports it. □

Theorem 4.1. (Partial Characterization of GCWA)
Let SEM be a semantics satisfying GPPE *and* Elimination of Tautologies.
Then: SEM$(P) \subseteq$ Min-MOD$^{Herbrand}_{2-val}(P)$ 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 $I \in \mathrm{SEM}(P)$ with $I \notin \text{Min-MOD}^{Herbrand}_{2-val}(P)$. 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 $\perp$ be a new atom. We consider the program

$$P' = P \;\cup\; \{\perp \leftarrow \mathrm{True}(I),\ \mathrm{Undef}(I)\}.$$

Now we only have to show

$$(*)\quad P' \text{ is SEM-equivalent to } P \text{ (i.e. } \mathrm{SEM}(P') = \mathrm{SEM}(P)),$$

and we arrive at a contradiction because $I[\![\perp]\!] = \mathbf{f}$ ($I \in \mathrm{SEM}(P)$ and P does not contain $\perp$) and therefore I can not be a 3-valued model of $\perp \leftarrow \mathrm{True}(I),\ \mathrm{Undef}(I)$.

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

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

Case 2: $\mathrm{Undef}(I) = \emptyset$.
This means that $I \in \mathrm{MOD}^{Herbrand}_{2-val}(P)$ but I is not a minimal model (our general assumption). There is therefore a $I' \models P$ with $I' \prec I$. Thus there exists an atom Y with $I'[\![Y]\!] = \mathbf{f} \neq \mathbf{t} = I[\![Y]\!]$. We apply GPPE to P' and replace Y. For any $Y \vee \mathcal{A} \in P$: $I'[\![\mathcal{A}]\!] = \mathbf{t}$ (since $I' \models P$) and therefore also $I[\![\mathcal{A}]\!] = \mathbf{t}$. So any of the disjunctions $Y \vee \mathcal{A}$ 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 non-trivial 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 $\mathrm{SEM}(P) \subseteq \text{Min-MOD}_{2-val}(P)$. Now we have to show the converse. So let $I \in \text{Min-MOD}_{2-val}(P)$.

In order to show that $I \in \mathrm{SEM}(P)$, 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

$$P'_0 := \{A \vee X_A \mid A \text{ is false in } I\}.$$

Now let P'_1 contain for every disjunction $\mathcal{A} \leftarrow\, \in P$ the rule

$$\{A \in \mathcal{A} \mid I \models A\} \;\leftarrow\; \{X_A \mid A \in \mathcal{A},\, I \not\models A\}.$$

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 $I'_0 \in \mathrm{SEM}(P'_0)$. By Lemma 2.1, SEM is consistent, so there must be an $\hat{I}_0 \in \mathrm{SEM}(P'_0)$. By the preceding theorem, $\hat{I}_0$ is a minimal model. But because of the simple structure of P'_0, there is an isomorphism which transforms $\hat{I}_0$ into I'_0. So our isomorphy condition yields $I'_0 \in \mathrm{SEM}(P'_0)$.

By Relevance, there must be an $I' \in \mathrm{SEM}(P')$ such that $\mathcal{RED}_{P'_0}(I') = I'_0$. 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 withe $I \models A$. 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

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

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

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

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 non-disjunctive programs, where WFS is a counterexample as can be seen from Theorem 3.1).

For the next theorems we need the *Elimination of Contradictions*. This condition implies that a semantics is based on 2-valued models:

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

Proof. Let $I \in \mathrm{SEM}(P)$ and suppose that there were an atom A with $I[\![A]\!] = \mathbf{u}$. Then we consider $P' = P \cup \{\bot \leftarrow A, \neg A\}$. Like 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: SEM(P) $\subseteq$ Weak-SUPP(P).*

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

$$P' = P \cup \{\bot \leftarrow X, \mathrm{True}(I),\ \neg\mathrm{False}(I)\}.$$

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 like 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 non-disjunctive programs, Weak-SUPP and SUPP collapse and SUPP satisfies GPPE:

Corollary 4.1. (SUPP for non-disjunctive programs)
Let SEM be a semantics for non-disjunctive programs satisfying GPPE *and* Elimination of Contradictions. *Then: SEM(P) $\subseteq$ SUPP(P).*

In particular, SUPP is the weakest semantics for non-disjunctive 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*[2]*:*

(1) SEM(P) $\subseteq$ Weak-SUPP(P) for all P.

(2) SEM(P) $\subseteq$ SUPP(P) for all P.

(3) SEM(P) $\subseteq$ 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 $I \in$ Weak-SUPP(P) but $I \notin$ SUPP(P). Then there is an atom X with $I[\![X]\!] = \mathbf{t}$ and for all clauses $X \vee \mathcal{A} \leftarrow \mathcal{B} \wedge \neg\mathcal{C} \in P$ either $I[\![\mathcal{B} \wedge \neg\mathcal{C}]\!] = \mathbf{f}$ or there is $Y_X \in \mathcal{A}$ with $I[\![Y_X]\!] = \mathbf{t}$. Let us assume that there are m clauses with X in their head, and that n of these satisfy $I[\![\mathcal{B} \wedge \neg\mathcal{C}]\!] = \mathbf{t}$, i.e. they have the form

$$X \vee Y_i \vee \mathcal{A}_i \leftarrow \mathcal{B}_i \wedge \neg\mathcal{C}_i$$

W. l. o. g. we select an atom B_j ($j = 1, \ldots, m-n$) with $I[\![B_j]\!] = \mathbf{f}$ 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

[2]In *(1) implies (2)* we need the instance of *Relevance* introduced after Definition 2.11.

apply GPPE). Here we need to make the assumption that SEM satisfies the special instance of *Relevance* introduced in Section 2.3. Let

$$P' = P \ \cup \ \{\bot \vee B_1 \vee \ldots \vee B_{m-n} \leftarrow X, \mathrm{True}(I), \neg\mathrm{False}(I)\}.$$

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

We first replace the X in $\bot \vee B_1 \vee \ldots \vee B_{m-n} \leftarrow X, \mathrm{True}(I), \neg\mathrm{False}(I)$ 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

$$\bot \vee B_1 \vee \ldots \vee B_{m-n} \vee Y_i \vee \mathcal{A}_i \leftarrow Y_1, \ldots, Y_n, \mathcal{B}_i \wedge \neg\mathcal{C}_i, \mathrm{True}(I), \neg\mathrm{False}(I)$$

that can also be cancelled since Y_i is contained both in the head and in 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 $\mathrm{SEM}(P) = \mathrm{SEM}(P')$. By Lemma 4.2, we know that $\mathrm{SUPP}(P') = \mathrm{STABLE}(P')$. So we finally have:

$$\mathrm{SEM}(P) = \mathrm{SEM}(P') \subseteq \mathrm{SUPP}(P') = \mathrm{STABLE}(P') = \mathrm{STABLE}(P).$$

The implication (3) $\Longrightarrow$ (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.

Theorem 4.4. (Second Partial Characterization of STABLE)
Let SEM be a semantics satisfying GPPE, Elimination of Tautologies*, and* Elimination of Contradictions*. Then: SEM*$(P) \subseteq$ *STABLE*(P).

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

Now let $I \in \mathrm{SEM}(P')$, 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 $I \in \mathrm{STABLE}(P')$. 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

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

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

In order to derive P' from P'', we first apply GPPE to the atom A 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 $\mathcal{A}' \leftarrow \neg\mathcal{C}'$ in P'', such that $A \in \mathcal{A}'$.

- If $I \not\models \neg\mathcal{C}'$, i.e. $I \models B$ for some $B \in \mathcal{C}'$, then the body of the resulting rule is inconsistent because of $B \in \mathrm{True}(I)$.

- Else ($I \models \neg\mathcal{C}'$), 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. □

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 non-disjunctive programs we immediately get

Corollary 4.2. (M_P^{supp} for non-disjunctive programs)
Let SEM be a semantics for non-disjunctive programs satisfying GPPE*,* Elimination of Tautologies *and* Elimination of Contradictions.
Then: SEM(P) $\subseteq \{M_P^{supp}\}$ for all stratified programs P.
In particular, if SEM(P) $\neq \emptyset$ then SEM(P) $= \{M_P^{supp}\}$.

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) $= \emptyset$. But even Weak-SUPP is already inconsistent for programs of the form $p \leftarrow \neg p$. Since we have proven SEM(P) $\subseteq$ Weak-SUPP(P), this also applies to any semantics SEM with the above properties. Thus we have

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) $= \emptyset$ for all programs P.

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

$$\text{SEM}(P \cup P') \subseteq \text{Weak-SUPP}(P \cup P') = \emptyset.$$

So any formula Q would follow from SEM($P \cup P'$), and therefore also from 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: *Relevance* (see also [DM94b, Dix95b]). Using our framework it is possible to show

Theorem 4.5. (Complete Characterization of PERFECT)
Any non-trivial semantics SEM satisfying SEM $\subseteq$ Weak-SUPP, GPPE*,* Elimination of Tautologies*,* Isomorphy *and* Relevance *already coincides with PERFECT on stratified disjunctive programs.*

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 $P = P_n \cup \ldots \cup P_0$ be a stratification of P. We assume that P_0 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:

C_k: There is a non-deterministic transformation $P_{\leq k} \rightsquigarrow P^*_{\leq k}$ with:

(1) For any $I_k \in$ PERFECT($P_{\leq k}$) there is a $P^*_{\leq k}$ such that there is exactly one $\mathcal{M}^*_{I_k} \in$ PERFECT($P^*_{\leq k}$) with $\mathcal{RED}_{P_{\leq k}}(\mathcal{M}^*_{I_k}) = I_k$. In addition,

for all X_A introduced during the transformation $P_{\leq k} \rightsquigarrow P^*_{\leq k}$, we have $\mathcal{M}^*_{I_k} \models X_A$.

(2) If $\mathcal{M}^* \in \mathrm{SEM}(P^*_{\leq k})$ and for all X_A introduced via $P_{\leq k} \rightsquigarrow P^*_{\leq k}$ we have $\mathcal{M}^* \models X_A$, then $\mathcal{RED}_{P_{\leq k}}(\mathcal{M}^*) \in \mathrm{SEM}(P_{\leq k})$.

(3) $\mathrm{SEM}(P_{\leq k}) = \mathrm{PERFECT}(P_{\leq k})$.

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

To prove the induction step $k \rightsquigarrow k+1$, let $I_{k+1} \in \mathrm{PERFECT}(P_{\leq k+1})$. By *Relevance* and Induction hypothesis there is a $I_k \in \mathrm{PERFECT}(P_{\leq k}) = \mathrm{SEM}(P_{\leq k})$ with $\mathcal{RED}_{P_{\leq k}}(I_{k+1}) = I_k$. By Induction hypothesis there is $P^*_{\leq k}$ with the above properties, i.e. there is exactly one $\mathcal{M}^*_k \in \mathrm{PERFECT}(P^*_{\leq k})$ with $\mathcal{RED}_{P_{\leq k}}(\mathcal{M}^*_k) = I_k$. 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 $A \vee \mathcal{A} \leftarrow \mathcal{B} \wedge \neg\mathcal{C} \in P_{k+1}$ and $I_{k+1} \models \neg A$, then we replace all rules $A \vee \mathcal{A}_i \leftarrow \mathcal{B}_i \wedge \neg\mathcal{C}_i$ from P_{k+1} with $\mathcal{A}_i \neq \emptyset$ by $\mathcal{A}_i \leftarrow \mathcal{B}_i \wedge \neg\mathcal{C}_i, X_A$ and $X_A \vee A$. The important property of this transformation is that all atoms occurring in non-trivial heads of P^*_{k+1} are already true in $\mathcal{M}^*_k$. 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 $\mathcal{M}^*_k$ and the new X_A. Therefore there is only one extension $\mathcal{M}^*_{k+1}$ of $\mathcal{M}^*_k$ to a model of P^*_{k+1} with $\mathcal{M}^*_{k+1} \models X_A$ for all X_A introduced during $P_{\leq k+1} \rightsquigarrow P^*_{\leq k+1}$. Obviously, by construction, $\mathcal{RED}_{P_{\leq k+1}}(\mathcal{M}^*_{k+1}) = I_{k+1}$ and we are done with (1).

(2) follows immediately from the induction Hypothesis and *Relevance* (the second condition in Definition 2.11).

(3) is the combination of (1) and (2): given a $I_{k+1} \in \mathrm{PERFECT}(P_{\leq k+1})$ we get using (1) a $\mathcal{M}^*_{k+1}$ with $\mathcal{RED}_{P_{\leq k+1}}(\mathcal{M}^*_{k+1}) = I_{k+1}$. Since the assumptions of (2) are satisfied, we have

$$I_{k+1} = \mathcal{RED}_{P_{\leq k+1}}(\mathcal{M}^*_{k+1}) \in \mathrm{SEM}(P_{\leq k+1}). \quad \Box$$

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 non-stratified, but yet equivalent (using some of our transformations) to a stratified one. For example although the program P_{ns}: "$A \leftarrow$, $A \leftarrow \neg A$" is non-stratified, it is certainly equivalent to "$A \leftarrow$" for any reasonable semantics and thus its intended model should be $\{A\}$.

We can eliminate such *non-genuine* cycles through negation by using our work in [BD95c] where we have associated to any program P a certain normal form $res(P)$, the *residual program*. $res(P)$ is obtained from P by using our transformations and some very weak reductions (if there is a clause $A \leftarrow$ then any occurrence of A (resp. $\neg A$) can be replaced by true (resp. false)). For the above program P_{ns} we get $res(P_{ns}) = A \leftarrow$.

We can also formally define the class of all programs that possess a stable model. This class extends the class of stratified programs but STABLE is not relevant on this class (see [DM94b]). It is easy to see that STABLE on this class satisfies *Independence* — thus *Relevance* is strictly stronger. Recently, however, Li-Yan

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 can not only be extended using our idea of deleting *non-genuine* cycles through negation, but it also represents the maximal such class. More formally, we have the following

Conjecture 4.1. (No Semantics for Genuine Non-Stratified Programs)
Let SEM be a non-trivial semantics satisfying GPPE*,* Elimination of Tautologies*,* Elimination of Contradictions *and* Relevance*.*
Then SEM is only defined on the class of all programs P such that $res(P)$ 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 ([Prz95]) or the D-WFS introduced by the authors ([BD94b, BD95c]).

We believe our conjecture to be true both for disjunctive and non-disjunctive programs. In the latter case already the wellfounded 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 [BD95a] 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 non-trivial 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 [BD96a, BD96b] a rigorous description of STABLE, WFS and their disjunctive counterparts based on certain confluent calculi of transformations is given. Finally [DS96] extends these transformations to programs with variables by using *constraint logic programming* techniques.

ACKNOWLEDGEMENTS

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

[ABW88.] K. R. Apt, H. A. Blair, and A. Walker. Towards a theory of declarative knowledge. In Jack Minker, editor, *Foundations of Deductive Databases*, chapter 2, pages 89–148. Morgan Kaufmann, 1988.

[AD94.] Chandrabose Aravindan and Phan Minh Dung. Partial Deduction of Logic Programs wrt Well-founded semantics. *New Generation Computing*, 13:45–74, 1994.

[AD95.] Chandrabose Aravindan and Phan Minh Dung. On The Correctness of Unfold/Fold Transformation of Normal and Extended Logic Programs. *Journal of Logic Programming*, 24(3):201–218, 1995.

[BD94a.] Stefan Brass and Jürgen Dix. A Characterization of the Stable Semantics by Partial Evaluation. In N. E. Fuchs and G. Gottlob, editors, *Proc. of the 10th Workshop on Logic Programming, Zürich, October 1994*, 1994.

[BD94b.] Stefan Brass and Jürgen Dix. A disjunctive semantics based on unfolding and bottom-up evaluation. In Bernd Wolfinger, editor, *Innovationen bei Rechen- und Kommunikationssystemen,* (IFIP '94-Congress, Workshop FG2: Disjunctive Logic Programming and Disjunctive Databases), pages 83–91, Berlin, 1994. Springer.

[BD95a.] Stefan Brass and Jürgen Dix. A General Approach to Bottom-Up Computation of Disjunctive Semantics. In J. Dix, L. Pereira, and T. Przymusinski, editors, *Nonmonotonic Extensions of Logic Programming*, LNAI 927, pages 127–155. Springer, Berlin, 1995.

[BD95b.] Stefan Brass and Jürgen Dix. Characterizations of the Stable Semantics by Partial Evaluation. In A. Nerode, W. Marek, and M. Truszczyński, editors, *Logic Programming and Non-Monotonic Reasoning, Proceedings of the Third International Conference*, LNCS 928, pages 85–98, Berlin, June 1995. Springer.

[BD95c.] Stefan Brass and Jürgen Dix. Disjunctive Semantics based upon Partial and Bottom-Up Evaluation. In Leon Sterling, editor, *Proceedings of the 12th Int. Conf. on Logic Programming, Tokyo*, pages 199–213. MIT Press, June 1995.

[BD96a.] Stefan Brass and Jürgen Dix. Characterizing D-WFS: Confluence and Iterated GCWA. In L.M. Pereira J.J. Alferes and E. Orlowska, editors, *JELIA '96*, LNCS 1126. Springer, 1996.

[BD96b.] Gerhard Brewka and Jürgen Dix. Knowledge representation with logic programs. Technical report, Tutorial Notes of the 12th European Conference on Artificial Intelligence (ECAI '96), 1996. Appeared as Technical Report 15/96, Dept. of CS of the University of Koblenz-Landau. Will also appear as Chapter 6 in *Handbook of Philosophical Logic*, 2nd edition, Volume 6, *Methodologies*.

[BDP96.] Stefan Brass, Jürgen Dix, and Teodor. C. Przymusinski. Characterizations and Implementation of Static Semantics of Disjunctive Programs. Technical Report TR 4/96, University of Koblenz, Department of Computer Science, Rheinau 1, February 1996.

[BD97.] Stefan Brass and Jürgen Dix. Semantics of Disjunctive Logic Programs Based on Partial Evaluation. *Journal of Logic Programming*, forthcoming, 1997.

[Bry90.] François Bry. Negation in logic programming: A formalization in constructive logic. In Dimitris Karagiannis, editor, *Information Systems and Artificial Intelligence: Integration Aspects*, pages 30–46, Berlin, 1990. Springer.

[Cla78.] K. L. Clark. Negation as Failure. In H. Gallaire and J. Minker, editors, *Logic and Data-Bases*, pages 293–322. Plenum, New York, 1978.

[Dix92.] Jürgen Dix. A Framework for Representing and Characterizing Semantics of Logic Programs. In B. Nebel, C. Rich, and W. Swartout, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR '92)*, pages 591–602. San Mateo, CA, Morgan Kaufmann, 1992.

[Dix95a.] Jürgen Dix. A Classification-Theory of Semantics of Normal Logic Programs: I. Strong Properties. *Fundamenta Informaticae*, XXII(3):227–255, 1995.

[Dix95b.] Jürgen Dix. A Classification-Theory of Semantics of Normal Logic Programs: II. Weak Properties. *Fundamenta Informaticae*, XXII(3):257–288, 1995.

[DK89.] P. M. Dung and K. Kanchansut. A fixpoint approach to declarative semantics of logic programs. In E.L. Lusk and R.A. Overbeek, editors, *Proceedings of North American Conference Cleveland,Ohio, USA*. MIT Press, October 1989.

[DM94a.] Jürgen Dix and Martin Müller. An Axiomatic Framework for Representing and Characterizing Semantics of Disjunctive Logic Programs. In Pascal Van Hentenryck, editor, *Proceedings of the 11th Int. Conf. on Logic Programming, S. Margherita Ligure*, pages 303–322. MIT, June 1994.

[DM94b.] Jürgen Dix and Martin Müller. Partial Evaluation and Relevance for Approximations of the Stable Semantics. In Z.W. Ras and M. Zemankova, editors, *Proceedings of the 8th Int. Symp. on Methodologies for Intelligent Systems, Charlotte, NC, 1994*, LNAI 869, pages 511–520, Berlin, 1994. Springer.

[DM94c.] Jürgen Dix and Martin Müller. The Stable Semantics and its Variants: A Comparison of Recent Approaches. In L. Dreschler-Fischer and B. Nebel, editors, *Proceedings of the 18th German Annual Conference on Artificial Intelligence (KI '94), Saarbrücken, Germany*, LNAI 861, pages 82–93, Berlin, 1994. Springer.

[DS96.] Jürgen Dix and Frieder Stolzenburg. Computation of Non-Ground Disjunctive Well-Founded Semantics with Constraint Logic Programming (preliminary report). In L. M. Pereira J. Dix and T. Przymusinski, editors, *Proceedings of the Postconference WS on* Nonmonotonic Extensions of LP *following JICSLP '96, Bad Honnef*, pages 290–302. University of Koblenz, TR 20/96, August 1996.

[GL91.] Michael Gelfond and Vladimir Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–387, 1991. (Extended abstract appeared in: Logic Programs with Classical Negation. *Proceedings of the 7-th International Logic Programming Conference, Jerusalem*, pages 579-597, 1990. MIT Press.).

[HY91.] Yong Hu and Li Yan Yuan. Extended Well-Founded Model Semantics for General Logic Programs. In Koichi Furukawa, editor, *Proceedings of the 8th Int. Conf. on Logic Programming, Paris*, pages 412–425. MIT, June 1991.

[Min82.] Jack Minker. On indefinite databases and the closed world assumption. In *Proceedings of the 6th Conference on Automated Deduction, New York*, pages 292–308, Berlin, 1982. Springer.

[Prz88.] Teodor Przymusinski. On the declarative semantics of deductive databases and logic programs. In Jack Minker, editor, *Foundations of Deductive Databases*, chapter 5, pages 193–216. Morgan Kaufmann, 1988.

[Prz91.] Teodor Przymusinski. Stable Semantics for Disjunctive Programs. *New Generation Computing Journal*, 9:401–424, 1991. (Extended abstract appeared in: Extended stable semantics for normal and disjunctive logic programs. *Proceedings of the 7-th International Logic Programming Conference, Jerusalem*, pages 459–477, 1990. MIT Press, Cambridge, Mass.).

[Prz95.] Teodor Przymusinski. Static Semantics For Normal and Disjunctive Logic Programs. *Annals of Mathematics and Artificial Intelligence*, 14:323–357, 1995.

[Sek91.] Hirohisa Seki. Unfold/fold transformation of stratified programs. *Theoretical Computer Science*, 86(1):107–139, 1991.

[Sek93.] Hirohisa Seki. Unfold/fold transformation of general logic programs for the well-founded semantics. *J. Logic Programming*, 16(1&2):5–23, 1993.

[SS95.] Chiaki Sakama and Hirohisa Seki. Partial Deduction of Disjunctive Logic Programs: A Declarative Approach. In *Logic Program Synthesis and Transformation – Meta Programming in Logic*, LNCS 883, pages 170–182, Berlin, 1995. Springer.