A Logical Foundation for Logic Programming II: Semantics of General Logic Programs

Alexander Bochman

email: bochman@macs.biu.ac.il

Abstract

We suggested in [12] a general logical formalism for Logic Programming based on a four-valued inference. In this paper we give a uniform representation of various semantics for logic programs based on this formalism. The main conclusion from this representation is that the distinction between these semantics can be largely attributed to the difference in their underlying (monotonic) logical systems. Moreover, in most cases the difference can even be reduced to that of the language, that is, to the difference in the logical connectives allowed for representing derivable information.

Keywords. Foundations of logic programming, negation as failure, semantics for logic programs, nonmonotonic reasoning.

1 Introduction

This is a second part of the study of a general logical formalism for logic programming (see [12]). Briefly, the formalism, called *biconsequence relations*, involves rules (bisequents) of the form $a : b \Vdash c : d$ that serve as direct representations of program rules of a most general kind, including disjunctive heads with possibly negative literals. We define first the notion of *circumscription* of a biconsequence relation, and then construct its *nonmonotonic* *completion* as a closure of the circumscription with respect to what we call *coherence rules*. The strength of these rules depend on the language $\mathcal{L}$ in which they are formulated. Finally, the *nonmonotonic semantics* of a biconsequence relation, $SEM^{\mathcal{L}}$, is defined as a pair (SEM^{+}, SEM^{-}) of sets of propositions in the language $\mathcal{L}$ that are, respectively, provable ($\Vdash A :$) and refutable ($A : \Vdash$) in the nonmonotonic completion.

In this part we will give representation of major semantics for normal disjunctive (and even more general) programs suggested in the literature. All of these representations will be based on the same notion of a nonmonotonic completion and semantics of a logic program, the only distinction being the language $\mathcal{L}$ in which the programs are represented.

2 WFS, D-WFS and Static Semantics

If our language $\mathcal{L}$ contains no connectives, the coherence rules are reduced to the case of singular atomic premises or conclusions, that is, to rules of the form

$$\frac{\Vdash A :}{: A \Vdash} \qquad\qquad \frac{A : \Vdash}{\Vdash : A}$$

where A is a propositional atom. Even in this case, however, the resulting semantics gives us the well-founded semantics for normal logic programs.

The following theorem holds for the language without connectives, that is for the case when the coherence rules (and the nonmonotonic semantics!) are restricted to atomic propositions. We will not give a direct proof of this theorem in the paper, because it is actually a consequence of more general representation results given later.

Theorem 2.1. *If P is a normal program, then $SEM_P^{\emptyset}$ coincides with the well-founded semantics of P.*

Though appropriate for normal logic programs, such a semantics is surely too weak for disjunctive programs, since it ignores, in effect, disjunctive information. Hence, a natural step is to extend the language at least by adding disjunction. If the language contains disjunction, Positive Coherence is already equivalent to the following structural rule (see [11]):

$$\frac{\Vdash a :}{: a \Vdash}$$

(where a is a set of propositions), though Negative Coherence is still reducible to its singular variant. It turns out that for disjunctive programs, the resulting semantics will coincide with the *Disjunctive Well-Founded Semantics (D-WFS)*, suggested recently by Brass and Dix (see [14]).

D-WFS is definable as the least semantics that is invariant under the following "Reduction & Elimination" transformations of a disjunctive program:

- *Elimination of Tautologies*: elimination of program clauses containing common atoms in heads and bodies. This transformation can be seen as a 'reductive' version of the Reflexivity axiom for biconsequence relations.

- *Elimination of Non-Minimal Rules*: if a rule $a \leftarrow b, \mathbf{not}\, c$ belongs to a program, then all rules of the form $a, a' \leftarrow b, b', \mathbf{not}\, c, \mathbf{not}\, c'$ can be eliminated. This is a reductive version of the Monotonicity rule for biconsequence relation.

- The *Generalized Principle of Partial Evaluation* (GPPE) – see [12].

- *Positive Reduction*: if an atom A does not belong to heads of clauses of a program, then all occurrences of A in negative premises can be eliminated. It is easy to see that this principle is a consequence of our Negative Coherence.

- *Negative Reduction*: If $a_0 \leftarrow$ belongs to a program, then all rules of the form $a \leftarrow b, \mathbf{not}\, c$ such that $a_0 \subseteq c$ can be eliminated. Clearly, this principle is a consequence of the above 'disjunctive' Positive Coherence rule.

Brass and Dix have shown that the construction of the D-WFS can be given in the form of a *confluent calculus* in which the above reduction rules can be applied in any order. It turns out that the construction of the nonmonotonic $\vee$-completion satisfies a similar confluence property, that is, the coherence rules can be applied both before and 'in the course of' circumscription. This is due to a special character of the coherence rules in this case that generate a negative extension of the source biconsequence relation (see Definition 3.2 in [12]).

The following theorem shows that D-WFS coincides with the $\vee$-semantics of a program. Notice that in this case SEM^+ contains disjunctions of atoms, while SEM^- – atomic propositions only. Though the definition of D-WFS admits also pure negative disjunctions (negations of conjunctions) of atoms, it is clear that what can be actually inferred in it is only negative atoms.

Theorem 2.2. *If P is a disjunctive program, then $SEM^{\vee}_{P}$ coincides with the disjunctive well-founded semantics (D-WFS) of P.*

As a result of the correspondence between D-WFS and the $\vee$-completion of a program, the relevant reduction calculus of Brass and Dix can be used as a method of computing this semantics. In other words, this reduction calculus provides a fairly general way of transforming our purely declarative description into a computation procedure.

The above semantics is also closely connected with Przymusinski's *static semantics* of general logic programs [27] (see [15]). The latter is defined in the framework of the Autoepistemic Logic of minimal Beliefs (AEB) which is formulated in the classical language augmented with a modal belief operator $\mathcal{B}$. General logic programs are translatable into this formalism by treating disjunction, conjunction and $\leftarrow$ as ordinary classical connectives, while **not** is translated as $\mathcal{B}\neg$. As can be seen, our four-valued translation is quite similar, the only distinction being the treatment of **not**. Moreover, the construction of the static semantics involves minimization with respect to objective propositions that has the same effect as circumscription in our construction (see [10] for details).

As is shown in [15], the static semantic is stronger than D-WFS. In an accompanying paper [10] we present a detailed description of this semantics. It is shown there that the static semantics of general logic programs can also be represented as an instance of our general construction, namely as a closure of a circumscribed biconsequence relation with respect to the following coherence rule:

$$\frac{a_1 :\Vdash c :}{: c \Vdash : a_1}$$

where a_1 is either empty or contains one proposition only. As can be easily seen, the above rule is stronger than $\vee$-coherence, but is weaker than $\{\vee, \neg\}$-coherence (see below).

A couple of examples, taken from [27], will help to clarify the correspondence between the static semantics and our construction. Let us consider the following (stratified) disjunctive program:

$$Write\_Paper\_1 \vee Write\_Paper\_2$$
$$Get\_Crazy \leftarrow Write\_Paper\_1 \wedge Write\_Paper\_2$$
$$Get\_Fired \leftarrow \textbf{not}\, Write\_Paper\_1 \wedge \textbf{not}\, Write\_Paper\_2$$

It is easy to show that the following bisequents belong to the corresponding circumscribed biconsequence relation:

$$(1) \qquad \Vdash^c Write\_Paper\_1, Write\_Paper\_2 :$$
$$(2) \qquad Write\_Paper\_1, Write\_Paper\_2 : \Vdash^c$$
$$(3) \qquad Get\_Crazy : \Vdash^c$$
$$(4) \qquad Get\_Fired : \Vdash^c : Write\_Paper\_1$$
$$(5) \qquad Get\_Fired : \Vdash^c : Write\_Paper\_2$$

The first three bisequents give us that the following formulas are provable in ${}^c\Vdash$: $Write\_Paper\_1 \vee Write\_Paper\_2$, $\neg Write\_Paper\_1 \vee \neg Write\_Paper\_2$, $\neg Get\_Crazy$. Now, since the above 'static' coherence rule is applicable to (1) in our case, we obtain

$$(6) \qquad : Write\_Paper\_1, Write\_Paper\_2 \ {}^c\Vdash$$

and hence, using (4) and (5) and a purely logical inference, we conclude

$$(7) \qquad Get\_Fired : \ {}^c\Vdash$$

Finally, applying Negative Coherence to (3) and (7), we obtain

$${}^c\Vdash \ : Get\_Crazy \quad \text{and} \quad {}^c\Vdash \ : Get\_Fired$$

in full accordance with the D-WFS and static semantics.

The following example, also from [27], is especially interesting in comparing our approaches.

$$Goto\_Australia \vee Goto\_Europe$$
$$Goto\_Both \leftarrow Goto\_Australia \wedge Goto\_Europe$$
$$Save\_Money \leftarrow \textbf{not}\, Goto\_Both$$
$$Cancel\_Reservation \leftarrow \textbf{not}\, Goto\_Australia$$
$$Cancel\_Reservation \leftarrow \textbf{not}\, Goto\_Europe$$

Now the following bisequents belong to the circumscribed biconsequence relation:

$$(1) \qquad \Vdash Goto\_Australia \vee Goto\_Europe :$$

$$(2) \qquad Goto\_Australia \wedge Goto\_Europe : \Vdash$$

$$(3) \qquad Goto\_Both : \Vdash$$

In addition, the following bisequent is a consequence of the last two clauses of the program:

$$: Goto\_Australia \wedge Goto\_Europe \Vdash Cancel\_Reservation :$$

However, the 'static' coherence rule is inapplicable to (2), because the relevant formula contains a conjunction. Consequently, the corresponding nonmonotonic semantics says that we are not going to both Europe and Australia (and hence will save money), but still does not imply that we should cancel reservation, as is predicted by the perfect semantics for this (stratified) program. As can be seen, this is because the semantics *does not use* negative conjunctive information.

Remark. As Przymusinski himself demonstrates, his modal formalism can accommodate this inference, but to this end it should sacrifice, in fact, the intended interpretation of the modal operator $\mathcal{B}$ used to translate **not**. To be more exact, in order to allow the inference, we should accept an axiom saying that $\mathcal{B}$ is distributive with respect to disjunctions. Moreover, in this case we will obtain a stationary semantics described in the next section. However, such an axiom makes it practically impossible to see the corresponding modal operator as an operator of *belief*, since it requires, for example, that our beliefs should be complete: for any proposition A we should believe either in A or in $\neg A$. In this respect our representation based on a four-valued logic has an advantage that these semantics are obtainable as natural and justifiable variations of the source construction.

3 Local Completions: Stationary Semantics and Stable Classes

In order to account for Przymusinski's stationary semantics in its different versions, we only need to extend the class of formulas admissible for coherence rules.

3.1 $\{\vee, \wedge\}$-Completion and 'Pure' Stationary Semantics.

We begin with an earlier version of the stationary semantics, suggested by Przymusinski in [26]. In fact, the formalism used by Przymusinski in that paper is almost identical with the four-valued logic in the language $\{\vee, \neg, \sim\}$, the only addition being that our switching negation $\sim$ (that corresponds to **not** in Przymusinski's formalism) satisfies the rule of double negation. Actually, the language of [26] does not allow for iterations of **not** at all and treats formulas of the form **not** A (where A is a classical proposition in the language $\{\vee, \neg\}$) as new propositional atoms.

As is shown in the first part [12] (Proposition 2.1), any bisequent can be represented by a formula in the language $\{\vee, \neg, \sim\}$. Consequently, any theory P in Przymusinski's formalism can be directly identified with a biconsequence relation $\Vdash_P$ in the language $\{\vee, \neg\}$. Moreover, Przymusinski's notion of a minimal model is actually equivalent to our notion of a positively minimal bimodel. Consequently, a formula A is minimally entailed by a theory P in the sense of Przymusinski ($P \models_{min} A$) if and only if it is provable in the circumscribed biconsequence relation corresponding to P, that is $\Vdash^c_P A :$.

The 'official' definition describes a *stationary expansion* of a disjunctive program P as any theory P^* satisfying the following condition:

$$P^* = P \cup \{\mathbf{not}\, F \mid P^* \models_{min} \neg F\},$$

where F is a *pure* conjunction of positive (respectively, negative) literals. *Stationary completion* is defined then as a least stationary expansion.

We will call the above notions of stationary expansion and completion *pure* ones. Though Przymusinski remarks in a footnote that the restriction to pure positive and negative conjunctions in the definition is unnecessary, we will see that it actually reduces stationary completions to the case of the language $\{\vee, \wedge\}$ in our construction.

Adding conjunction to the language of formulas susceptible to coherence retains the previous Positive Coherence rule, but gives a stronger, 'multiple' variant of Negative Coherence:

$$\frac{a : \Vdash}{\Vdash : a}$$

Our first result identifies the $\{\vee, \wedge\}$-completion of a program with the circumscription of its pure stationary completion (without the disjunctive inference rule, DIR – see below).

Theorem 3.1. *If $S(P)$ is a pure stationary completion of a disjunctive program P, then* ${}^{c}\Vdash_{P}^{\{\vee,\wedge\}} = \Vdash^{c}_{S(P)}$.

The above theorem says, in effect, that if we will identify the $\{\vee, \wedge\}$-completion with a set of formulas in the language $\{\vee, \neg, \sim\}$, then it will coincide with the set of formulas that are *minimally entailed* by the stationary completion of a program. As an immediate consequence, we have

Corollary 3.2. *If $S(P)$ is a pure stationary completion of a disjunctive program P and $SEM_P = (SEM^+, SEM^-)$ is a $\{\vee, \wedge\}$-semantics of P, then*

- $A \in SEM^+$ *iff* **not** $\neg A \in S(P)$*;*
- $A \in SEM^-$ *iff* **not** $A \in S(P)$*.*

The construction of the stationary semantics for disjunctive programs in [26] involved also the disjunctive inference rule, DIR, introduced in order to cover the perfect semantics for stratified disjunctive programs. The resulting 'extended' stationary semantics has many interesting features that we are intending to discuss elsewhere. We mention here only that two special cases of DIR, used in [22] for defining the stationary semantics, have turned out to be structural equivalents of the coherence rules in the language $\{\vee, \wedge, \mathbf{L}\}$. However, DIR is used in Przymusinski's construction as a general constraint on disjunctive logic programs and hence is applied, in a sense, both before and in the course of circumscription. Moreover, confluence no longer holds in this case, and applications of DIR cannot be restricted to the stage after performing circumscription.

3.2 $\{\vee, \neg\}$-Completion and 'Full' Stationary Semantics.

In a later paper, [27], Przymusinski gives another description of what he sees as an intended notion of a stationary semantics. It is obtained from the static semantics by way of adding an axiom of distributivity of the belief operator with respect to disjunctions. As can be easily shown, however, this characterization is equivalent to a definition of a stationary expansion obtained by dropping the restriction to pure conjunctions and taking instead arbitrary classical formulas. We are going to show now that this version of stationary semantics amounts to the case of the language $\{\vee, \neg\}$ in our construction.

Local negation and disjunction generate an important class of connectives we call *local* ones. These are the connectives that are determined 'locally' with respect to truth and falsity and behave classically in each of these contexts. As is shown in [11], the Positive and Negative Coherence rules in the language $\{\vee, \neg\}$ are already equivalent, and each of them amounts to the following structural rule:

$$\frac{a : \Vdash c :}{: c \Vdash : a}$$

The rule can be informally described as saying that any data consistent with respect to non-falsity should be also consistent with respect to truth (or that an entailment with respect to truth implies an entailment with respect to non-falsity). The semantic condition corresponding to this rule is that a negative part of any bimodel should be also a positive part of some bimodel, that is, if (u, v) is a bimodel, then there exists w such that (v, w) is also a bimodel.

The following definition is a straightforward reformulation of the definition of stationary expansions in the framework of biconsequence relations.

Definition 3.1. A biconsequence relation $\Vdash_*$ will be called a *stationary expansion* of a biconsequence relation $\Vdash$ if $\Vdash_*$ coincides with the least biconsequence relation containing $\Vdash$ and bisequents of the form $\Vdash: A$, for all $\{\vee, \neg\}$-propositions A such that $\Vdash_*^c \neg A :$.

As follows from the definition, any stationary expansion $\Vdash_*$ is uniquely determined by its circumscription, $\Vdash_*^c$. The following theorem gives a characterization of such circumscribed stationary expansions in terms of certain $\{\vee, \neg\}$-coherent extensions of the source biconsequence relation.

Theorem 3.3. *Let $\Vdash$ be an affirmative biconsequence relation. A biconsequence relation $\Vdash_0$ is a negative, $\{\vee, \neg\}$-coherent and regular extension of $\Vdash^c$ if and only if there is a stationary expansion $\Vdash_*$ of $\Vdash$ such that $\Vdash_0 = \Vdash_*^c$.*

As an immediate consequence of the above theorem, we obtain the following

Corollary 3.4. *$\Vdash_0$ is a nonmonotonic $\{\vee, \neg\}$-completion of an affirmative biconsequence relation $\Vdash$ iff $\Vdash_0 = \Vdash_*^c$, where $\Vdash_*$ is a stationary completion of $\Vdash$.*

Thus, the nonmonotonic $\{\vee, \neg\}$-completion of an affirmative biconsequence relation coincides with the circumscription of its stationary completion. The following example shows that the resulting semantics is stronger than the $\{\vee, \wedge\}$-semantics even for disjunctive programs.

$$\begin{array}{ll} \Vdash A, C : & : A, B \Vdash D : \\ : C \Vdash A : & : A, D \Vdash A, D : \\ : A \Vdash C : & : A, D \Vdash B, D : \end{array}$$

In the $\{\vee, \wedge\}$-completion of this biconsequence relation the following sets of propositional atoms form positive parts of bimodels: $\{C\}$, $\{C, D\}$, $\{A\}$, $\{A, B, C\}$. On the other hand, the $\{\vee, \neg\}$-completion has only two bimodels, $(\{A\}, \{A\})$ and $(\{C, D\}, \{C, D\})$ (and hence coincides with the full classical completion – see below). Consequently the latter completion contains $\Vdash A, D :$ and $B : \Vdash$, and hence the corresponding semantics is more informative than the $\{\vee, \wedge\}$-semantics, both positively and negatively.

A somewhat simpler example showing the difference between the two semantics can be given for (affirmative) biconsequence relations containing negative conclusions:

$$\begin{array}{ll} : A \Vdash A : & : C \Vdash B : A \\ : B \Vdash A : & : B, C \Vdash B : \\ \Vdash A : C & : B, C \Vdash C : \end{array}$$

The $\{\vee, \wedge\}$-semantics for this biconsequence relation is $\langle \{A \vee B\}, \emptyset \rangle$, while the $\{\vee, \neg\}$-semantics is

$$\langle \{A\}, \{\neg A \wedge C, \neg B \wedge C\} \rangle.$$

In many respects, the $\{\vee, \neg\}$-semantics is the largest semantics in our 'classification' having all the good properties a reasonable semantics could have. For example, the $\{\vee, \neg\}$-completion still enjoys the confluence property (see above), and hence the coherence rules can be applied before and in the course of circumscription. In addition, such a semantics always exists for affirmative logic programs (see Proposition 5.2 in [12]). However, there are some seemingly plausible desiderata suggested for logic programs that this semantics still does not satisfy. For example, in the case of stratified disjunctive programs, it is weaker than the perfect semantics.

3.3 Stationary Expansions vs. Stable Classes.

Surprisingly enough, the language $\{\vee, \neg\}$ turns out to be also appropriate for describing a semantics of stable classes for normal programs suggested by Baral and Subrahmanian [3, 4].

Stable classes are defined in these papers as sets of sets of propositions S satisfying the following condition:

$$S = \{\mathbf{F}(s) \mid s \in S\},$$

where $\mathbf{F}$ is a *Gelfond-Lifschitz operator* giving the least model of the corresponding Gelfond-Lifschitz transformation of a normal program.

The correspondence between stable classes and our construction is based on an easily established fact (cf. [8]) that a pair (u, v) is a positively minimal bimodel of a normal program if and only if $u = \mathbf{F}(v)$. Consequently, stable classes turn out to correspond to certain sets of positively minimal bimodels.

As the following theorem shows, in the finite case there is a natural correspondence between stable classes of a normal program and its stationary expansions.

Theorem 3.5. *If P is a finite normal program, then negative parts of bimodels of any stationary expansion of $\Vdash_P$ form a stable class of P and any stable class of P is determined in this way by some stationary expansion of $\Vdash_P$.*

Note that any stationary expansion of $\Vdash$ is a negative extension of $\Vdash$ obtained by adding some negative propositions of the form $\sim A$ to it. If $\Vdash$ is a normal biconsequence relation, the added negative propositions constitute the only constraint on the set of possible negative parts of bimodels. Consequently, we have the following

Corollary 3.6. *Let P be a normal program, s a set of $\{\vee, \neg\}$-propositions and $\Vdash_*$ the least biconsequence relation containing P and $\sim\neg s$. If $\Vdash_*$ is a stationary expansion of $\Vdash_P$, then the stable class generated by $\Vdash_*$ coincides with the set of ordinary classical models of s.*

As an example, let us consider a (normal) biconsequence relation generated by the following bisequents:

$$: D \Vdash A : \qquad A : B, C \Vdash D :$$
$$: C \Vdash B : \qquad B : A, D \Vdash C :$$

It is easy to show that a stationary expansion of this biconsequence relation can be obtained by adding the following set of bisequents:

$$\Vdash : \neg A \wedge \neg B$$
$$\Vdash : (A \wedge B) \leftrightarrow (\neg C \wedge \neg D)$$
$$\Vdash : C \wedge D$$

This stationary expansion generates a stable class containing four interpretations: $\{A\}$, $\{B\}$, $\{A, B, C\}$ and $\{A, B, D\}$. As can be easily seen, these are precisely classical models of the following set of formulas:

$$\neg(\neg A \wedge \neg B)$$
$$\neg((A \wedge B) \leftrightarrow (\neg C \wedge \neg D))$$
$$\neg(C \wedge D)$$

in full accordance with the above result.

Note that the finiteness restriction in the formulation of the above theorem could be weakened in many ways. For example, the theorem still holds for *head-finite* programs, in which only the set of all atoms occurring in heads of their clauses is required to be finite. This is because in this case also any stable class should be finite[1]. In addition, the correspondence between stable classes and bimodels of the associated coherent extensions is preserved also in case when such classes contain only finite sets. And in all cases the validity with respect to a stable class is preserved: An atom A belongs to all elements of a stable class S if and only if it is provable in the associated biconsequence relation, and does not belong to any set from S iff it is refutable (provably false) in it.

Still, in the infinite case some bimodels of the associated biconsequence relation do not correspond to elements of a stable class (and even to positively minimal bimodels of the program). For example, consider the following program:

$$A_j \leftarrow \textbf{not}\, A_0, \text{ for all } j \neq 0;$$
$$A_j \leftarrow \textbf{not}\, A_i, \text{ for all } i, j \text{ such that } i \neq 0 \text{ and } i \neq j+1.$$

It can be shown that the set of all sets of the form $\overline{\{A_i\}}$ constitutes a stable class of the program. However, the corresponding $\{\vee, \neg\}$-coherent

[1] Note that finiteness has been made a part of the definition of stable classes in [3].

biconsequence relation admits also (At, At) as its bimodel, where At is a set of all atoms.

Summing up the above considerations, we can claim that stationary expansions can serve as a natural generalization of the stable class semantics to more general programs. Note that, since the nonmonotonic $\{\vee, \neg\}$-completion of a normal program is also the least coherent and regular extension of $\Vdash^c_P$, we obtain that SEM_P contains exactly those propositions that hold in all stable classes. Speaking more generally, a reasoning based on stable classes (and stationary expansions) can be characterized as reasoning with respect to the circumscribed consequence relation subject to the following constraint (corresponding to a combination of regularity and coherence):

$$a : \Vdash b : \qquad \text{iff} \qquad : b \Vdash : a$$

Remark. Baral and Subrahmanian considered some 'preferred' stable classes as determining the meaning of a logic program. To begin with, they restricted their attention to strict (inclusion minimal) stable classes. (The latter correspond to maximal stationary expansions.) In [3] the authors studied two possible preference orders on such stable classes. In fact, it is possible to describe these orders in terms of the corresponding orderings on maximal stationary expansions as follows:

- $\Vdash_1 \leq_s \Vdash_2$ iff, for any set of atomic propositions a, $a : \Vdash_2$ implies $a : \Vdash_1$;
- $\Vdash_1 \leq_h \Vdash_2$ iff, for any set of atomic propositions a, $\Vdash_1 a :$ implies $\Vdash_2 a :$;

This approach clearly indicates a possible direction in which our basic construction of nonmonotonic completion could be extended. As it seems, both this and similar 'preference-based' approaches can be accounted for in terms of some further *maximization principles* imposed on the nonmonotonic completion.

4 Invariant Completion and Partial Stable Semantics

An interesting additional requirement that can be imposed on a four-valued reasoning is that reasoning with respect to truth should be 'isomorphic' to reasoning with respect to non-falsity (after all, both have the same meaning for us in the classical case). This requirement can be met by restricting our logical connectives to those having identical characterizations with respect to truth and non-falsity. Making precise this latter restriction would lead us to the class of connectives we call *invariant* (see [11]). It turns out that invariant connectives are exactly the classical four-valued connectives that are expressible via $\{\vee, \neg, \sim\}$. As we will show now, the corresponding nonmonotonic completion will give us a generalization of Przymusinski's partial stable semantics from [24].

It is shown in [11] that the coherence rules in the the language $\{\vee, \neg, \sim\}$ amount to the following structural rule:

(Invariance) $$\frac{a : b \Vdash c : d}{d : c \Vdash b : a}$$

Any $\{\vee, \neg, \sim\}$-coherent biconsequence relation will be also called *invariant*. Invariant biconsequence relations are characterized by the property that if (u, v) is a bimodel, then (v, u) is also a bimodel. Accordingly, a bimodel (u, v) of an arbitrary biconsequence relation $\Vdash$ will be called *invariant* if (v, u) is also a bimodel of $\Vdash$. Clearly, an invariant biconsequence relation can now be characterized by the condition that all its bimodels are invariant.

The following definition describes the main semantic objects of this section.

Definition 4.1. Consistent bimodels of an invariant completion of a biconsequence relation $\Vdash$ will be called *p-stable bimodels* of $\Vdash$.

The following theorem gives a direct description of such bimodels:

Theorem 4.1. *(u, v) is a p-stable bimodel of a biconsequence relation $\Vdash$ if and only if $u \subseteq v$ and both (u, v) and (v, u) are positively minimal bimodels of $\Vdash$.*

The next theorem shows that p-stable bimodels of any normal program coincide with its partial stable models.

Theorem 4.2. *If P is a normal program, then p-stable bimodels of $\Vdash_P$ are exactly partial stable models of P.*

The above representation of the partial stable semantics in terms of p-stable bimodels makes also apparent its connection with the stable class semantics described in the preceding section. Namely, the partial stable semantics is obtained by restricting the set of stable classes to ordered classes containing at most two elements.

Przymusinski has also suggested a generalization of partial stable semantics for disjunctive programs (see [25]). The resulting *disjunctive partial stable semantics* is somewhat weaker, however, than ours. To begin with, the following result shows that any p-stable bimodel is a disjunctive partial stable model.

Theorem 4.3. *If P is a disjunctive program, then any p-stable bimodel of $\Vdash_P$ is a disjunctive partial stable model of P.*

The following simple example demonstrates that the two semantics do not coincide:

$$\Vdash A, B : \qquad : A \Vdash A :$$

The invariant completion of this program contains a single consistent bimodel $(\{A\}, \{A\})$ (that corresponds to a stable model $\{A, \sim B\}$), while Przymusinski's semantics accepts also $\{B\}$ (without negative literals) as another disjunctive partial stable model.

Remark. The difference between p-stable bimodels and disjunctive partial stable models is due to the fact that the latter are defined in terms of *consistent* Herbrand models. In fact, the only thing we must do in order to obtain a characterization of p-stable bimodels is to drop the restriction to consistent models in the definition of a minimal model, given in [25]. Still, the two definitions can be shown to coincide for normal programs. They also give the same results for *total* stable models (see below).

It can be shown that any disjunctive partial stable model in the sense of Przymusinski is also a positively minimal bimodel of a program. However, Przymusinski's original definition has a disturbing feature that the resulting models are not determined solely by positively minimal bimodels. In other words, they violate the Positive Minimization principle. As a result, GPPE also fails. This can be demonstrated by adding the following clause

$$B : A \Vdash C :$$

to the previous example. An interpretation $\{B, \sim C\}$ is not even a model of this program in the sense of Przymusinski, though it is a partial stable model of the residual program obtained by applying GPPE (the latter amounts to eliminating the last clause).

Now we consider an alternative description of the p-stable semantics. Note first that a bimodel (u, v) is consistent iff (v, u) is complete. Consequently, in computing the p-stable semantics we can restrict our attention to bimodels that are either consistent or complete. This restriction can be expressed with the help of the following structural rule (see [11] for details):

$$\text{(CC)} \qquad A : A \Vdash B : B$$

Biconsequence relations satisfying the rule CC will be called *ordered*.

Recall that any bisequent is expressible by a formula in an invariant language (see Proposition 2.1 in [12]). Hence, the set of invariant propositions provable in a nonmonotonic completion provides, in a sense, an equivalent description of the whole completion. Note also that a proposition $\sim A$ is true in a bimodel if and only A is false in it. Consequently, if our language contains a switching negation $\sim$, then any bimodel is uniquely determined by the set of propositions that are true with respect to it. Consequently, any bimodel (u, v) can be identified in this case with its positive part, u.

The following result gives an alternative characterization of the set of $\{\vee, \sim\}$-propositions that belong to the p-stable semantics.

Theorem 4.4. *Let $\Vdash^{cc}$ be the least ordered invariant biconsequence relation containing $\Vdash^{c}$. Then a $\{\vee, \sim\}$-proposition belongs to all p-stable bimodels of $\Vdash$ iff it is provable in $\Vdash^{cc}$.*

As follows from the above result, the p-stable semantics is determined by the set of all propositions provable in the circumscribed biconsequence relation with added rules Invariance and CC. Furthermore the next result shows that a *brave reasoning* with respect to p-stable semantics (that is, validity with respect to at least one p-stable bimodel) amounts to a 'classical' consistency check in $\Vdash^{cc}$.

Theorem 4.5.

1. *A $\{\vee, \sim\}$-proposition A belongs to some p-stable bimodel of a biconsequence relation $\Vdash$ iff it can be classically true with respect to $\Vdash^{cc}$, that is, if $A : \nVdash^{cc} : A$.*

2. *A $\{\vee, \sim\}$-proposition A is false in some p-stable bimodel of a biconsequence relation $\Vdash$ iff it can be classically false with respect to $\Vdash^{cc}$, that is, if $: A \nVdash^{cc} A :$.*

As follows from the results of Przymusinski in [25], p-stable semantics subsumes both the well-founded semantics of normal logic programs and the perfect semantics for stratified disjunctive programs. Unfortunately, the following example given in [25] shows that it is not always consistent even for disjunctive programs:

$$\begin{aligned} Work \lor Tired \lor Sleep \\ Work \leftarrow \mathbf{not}\, Tired \\ Sleep \leftarrow \mathbf{not}\, Work \\ Tired \leftarrow \mathbf{not}\, Sleep \end{aligned}$$

The static, and even stable class semantics for this program contains $Work \lor Tired \lor Sleep$, and $\neg Work \lor \neg Tired \lor \neg Sleep$. However, the corresponding circumscribed biconsequence relation contains no invariant bimodels and hence it does not have a consistent p-stable semantics. Moreover, an example we already used earlier

$$\begin{aligned} \Vdash A, B : \\ : A \Vdash A : \\ : B \Vdash B : \end{aligned}$$

can also be used to demonstrate that the p-stable semantics may not exist even for those disjunctive programs that have a consistent invariant completion. Thus, the invariant completion of the above biconsequence relation has only one bimodel $(\{A\}, \{B\})$ that is neither consistent nor complete. Consequently, it does not have a p-stable semantics.

Finally, it can be shown that, as far as $\{\lor, \sim\}$-propositions are concerned, invariant semantics coincides with $\{\lor, \sim\}$-semantics.

Theorem 4.6. *A $\{\lor, \sim\}$-semantics for any biconsequence relation coincides with a restriction of the $\{\lor, \neg, \sim\}$-semantics to $\{\lor, \sim\}$-propositions.*

The above theorem shows, in effect, that in computing the set of $\{\lor, \sim\}$-propositions provable or refutable in the $\{\lor, \sim\}$-completion, we can consider provability in the corresponding invariant completion.

Remark. $\{\lor, \sim\}$-propositions can be considered as describing *generalized states* in the sense of Baral [2]. It remains to be seen how our construction is related to that suggested by Baral.

To end this section, we will consider a relatively simple logical procedure for computing the p-stable semantics. The procedure is based on an observation that in order to secure invariance of a biconsequence relation generated by a set of bisequents S, it is sufficient to add to S all 'mirror images' of bisequents from S.

If $K(\Vdash)$ is a kernel of a (locally finite) biconsequence relation $\Vdash$, we will denote by $K^i(\Vdash)$ the union of $K(\Vdash)$ and all bisequents $d : c \Vdash b : a$ such that $a : b \Vdash c : d$ belongs to $K(\Vdash)$. The following result shows that the resulting set determines the invariant completion of $\Vdash$.

Theorem 4.7. *If $\Vdash$ is a locally finite biconsequence relation, then $\Vdash_{K^i(\Vdash)}$ coincides with the invariant completion of $\Vdash$.*

Applying now Theorem 4.4, we obtain

Corollary 4.8. *A $\{\vee, \sim\}$-proposition belongs to all p-stable bimodels of $\Vdash$ iff it is provable from $K^i(\Vdash)$ using (CC) as the only additional rule.*

Further details on provability in ordered biconsequence relations can be found in [11].

5 Classical Completion and Stable Semantics

The following proposition, proved in [11], can be used to show that coherent biconsequence relations in the language with all the classical connectives coincide with ordinary 'classical' consequence relations.

Proposition 5.1. *A biconsequence relation is $\{\vee, \neg, \mathbf{L}\}$-coherent iff it satisfies the following two rules:*

(Consistency)	$A : A \Vdash$
(Completeness)	$\Vdash A : A$

As can be easily seen, biconsequence relations satisfying the above two rules are determined by classical bimodels, that is, by bimodels of the form (i, i). This means, in particular, that any bisequent $a : b \Vdash c : d$ is equivalent to a bisequent $a, d :\Vdash b, c :$. As a result, such a biconsequence relation can be considered as an ordinary consequence relation, or a sequent calculus.

Moreover, in this case the switching negation becomes equivalent to the local negation, while **L** reduces to a trivial operator. Consequently, the resulting system becomes identical to an ordinary classical sequent calculus.

The following result shows that the classical completion corresponds to stable, or answer set, semantics of general logic programs (see, e.g., [19, 20, 21]).

Theorem 5.2. *Stable models of any program P coincide with bimodels of the nonmonotonic completion of P in the full classical language.*

Thus, SEM_P in the full classical language is precisely the stable semantics of logic programs, since it contains all the propositions that are, respectively, true and false in all stable models.

The above results imply, in particular, that both brave and skeptical reasoning with respect to stable semantics amount, in fact, to a classical reasoning with respect to a circumscribed biconsequence relation. In other words, after performing circumscription of the source biconsequence relation, we can treat the resulting biconsequence relation as an ordinary classical sequent calculus. Propositions provable in this calculus will be exactly those that belong to the stable semantics, while propositions that are consistent with respect to it will coincide with those that hold in at least one stable model.

These considerations suggest an especially simple procedure for computing the stable semantics in the finite case. Let $K(\Vdash)$ be a kernel of $\Vdash$ considered as a set of formulas in the language $\{\vee, \neg, \sim\}$ (see above). We will denote by $K^s(\Vdash)$ the result of replacing all occurrences of $\sim$ by $\neg$ in all formulas from $K(\Vdash)$. Notice that $K^s(\Vdash)$ can be considered as a classical theory in the language $\{\vee, \neg\}$. The following result shows that ordinary classical models of this theory coincide with stable models of $\Vdash$. It follows from the fact that the classical completion of a biconsequence relation is the least classical (bi)consequence relation containing the kernel $K(\Vdash)$ (see Theorem 3.4 in the first part [12]).

Corollary 5.3. *If $\Vdash$ is a locally finite biconsequence relation, then stable models of $\Vdash$ are exactly classical models of $K^s(\Vdash)$.*

As a result, the computation of the stable semantics is reducible to checking provability and consistency with respect to a certain classical theory.

Remark. In the case of disjunctive programs, the above procedure coincides with the procedure of computing the stable semantics suggested by Brass and Dix in [13].

The stable semantics for logic programs constitutes, in a sense, a limit case – no nontrivial extension of it is possible in the framework of our general construction. It is also the strongest semantics: if a program has a classical completion, then all other semantics described in this paper will be consistent for this program.

6 Conclusions and Further Issues

This study has arisen from an 'abductive' problem of finding a most plausible logical explanation for existing semantics of logic programs involving negation as failure. However, as it is the case with any really plausible theoretical explanation, our logical framework has allowed us to discover some new facts and connections between existing semantics, and to suggest also new variants thereof. It provides also a clear theoretical basis for evaluating future proposals in this field.

It has turned out that common kinds of reasoning about logic programs are reducible, in effect, to different kinds of logical, monotonic reasoning in a certain nonmonotonic extension of the source theory. Recall that in this paper we have used only *one* general semantics for logic programs, more specific semantics being merely partial cases obtained by restricting the underlying language.

Among other things, the representation of various semantics as species of one general semantics suggests an entirely different, more tolerant, view of these semantics. Instead of the question what is the best semantics in all cases, we can consider the question what expressive means are most appropriate for a particular reasoning task. On this view, the main question is an acceptable trade-off between expressive means and computational cost for a particular context of reasoning. On any choice, we are dealing, in effect, with the same semantics.

As to the computational cost, our construction shows that the computation of nonmonotonic semantics involves only one nontrivial step, namely circumscription. All the rest is a rule-based monotonic reasoning. It should be clear, however, that this issue deserves a further study.

Among the vast number of issues and problems that still need to be studied about our formalism, we want to mention only two.

One major issue is what logical rules are admissible or appropriate to use *before* circumscription. If a logical rule is admissible in this sense, it may greatly facilitate the computation of the nonmonotonic completion. Thus, the correspondence between our construction and stationary semantics can be used to show that $\{\vee, \neg\}$-coherence rules can also be used before (as well as in the course of) circumscription. Taking another example, the Consistency rule is admissible in this sense with respect to the stable semantics (see [8, 6]), though the Completeness rule is not. In fact, some alternative semantics for logic programs, not described in this paper, can be obtained by imposing certain logical rules on the source biconsequence relation, e.g., Schlipf's stable-by-case semantics [29] (see [8] for details). In addition, the issue is intimately connected with some current approaches to *nonmonotonic revision* of logic programs (see, e.g., [18, 32]), according to which in cases when the resulting nonmonotonic semantics is inconsistent, we can extend the source program by new rules (e.g., contrapositions of some of the existing rules) that would restore consistency.

Another major issue is that the nonmonotonic semantics can be strengthened in many cases by considering only some 'preferred' bimodels of a nonmonotonic completion. This approach has been intensively studied, mainly in the context of normal programs, but see, e.g., [30], where it is applied to disjunctive programs. As we already mentioned in discussing the stable class semantics, many such constructions can be accounted for in our framework by applying some general *maximization principles* to the nonmonotonic completion.

Finally, as we mentioned in the first part of this study, there is a fairly general way of 'lifting' our formalism to a formalism that subsumes classical inference (see [7]). In this way we can also obtain a representation of extended logic programs containing a second, 'classical' negation[2]. In addition, such an extension provides a natural framework for capturing some recent attempts to 'transfer' semantics for logic program to broader nonmonotonic contexts. See [7, 6, 10] for some initial results in this direction.

[2]Actually, our coherence rules for such languages are strongly related to the coherence rule for the second negation from [1].

References

[1] J.J. Alferes and L.M. Pereira. (1992) On logic program semantics with two kinds of negation. In K. R. Apt, editor, *Logic Programming: Proc. 1992 Joint Int. Conf. and Symp.*, pages 574–589, Cambridge, Mass., MIT Press.

[2] C. Baral (1992) Generalized negation as failure and semantics of normal disjunctive logic programs. In A. Voronkov (ed) *Logic Programming and Automated Reasoning* (LNAI 624), pp. 309–319.

[3] C. Baral and V.S. Subrahmanian (1991) Dualities between alternative semantics for logic programming and nonmonotonic reasoning (Extended abstract). *Proc. First Int. Workshop on Logic Programming and Nonmonotonic Reasoning*, pp. 69–86.

[4] C. Baral and V.S. Subrahmanian (1992) Stable and extension class theory for logic programs and default logics. *J. of Automated Reasoning*, **8**: 345–366.

[5] A. Bochman (1995) Default consequence relations as a logical framework for logic programs. *Proc. Third International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR'95*, Lecture Notes in Artificial Intelligence, 928, pp. 245–258.

[6] A. Bochman (1995) On bimodal nonmonotonic logics and their unimodal and nonmodal equivalents. *Proc. IJCAI'95*, Morgan Kaufmann, pp. 1518–1524.

[7] A. Bochman (1996) Biconsequence relations for nonmonotonic reasoning. In L.C. Aiello, J. Doyle, and S.C. Shapiro (eds.) *Principles of Knowledge Representation and Reasoning: Proc. Fifth Int. Conference (KR'96)*, Morgan Kaufmann, San Francisco, CA.

[8] A. Bochman (1996) On a logical basis of normal logic programs. *Fundamenta Informaticae*, **28**:223–245.

[9] A. Bochman (1996) What is a four-valued reasoning and how it can be nonmonotonic. (Submitted to *J. of Logic and Computation*)

[10] A. Bochman (1996) A study of Przymusinski's static semantics. To appear in *Proc. LPNMR'97*.

[11] A. Bochman (1997) Biconsequence relations: A general formalism of reasoning with inconsistency and incompleteness. (Submitted)

[12] A. Bochman (1997) A logical foundation for logic programming I: Biconsequence relations and nonmonotonic completion. *J. of Logic Programming*

[13] S. Brass and J. Dix (1995) A general approach to bottom-up computation of disjunctive semantics. In J. Dix, L. Pereira, and T. Przymusinski (eds.) *Nonmonotonic Extensions of Logic Programming*, LNAI 927, Springer, pp. 127–155.

[14] S. Brass and J. Dix (1997) Characterizations of the Disjunctive Well-founded Semantics: Confluent Calculi and Iterated GCWA. *Journal of Automated Reasoning*, to appear.

[15] S. Brass, J. Dix I. Niemelä and T. Przymusinski (1997) Comparison and efficient computation of the static and the Disjunctive WFS. In G. Brewka, E. Weydert, and C. Witteveen, editors, *Proceedings of the third Dutch-German Workshop on Nonmonotonic Reasoning and its Applications*, pages 37–42. February 1997.

[16] S. Brass, J. Dix and T. Przymusinski (1996) Super logic programs. In L.C. Aiello, J. Doyle, and S.C. Shapiro (eds.) *Principles of Knowledge Representation and Reasoning: Proc. Fifth Int. Conference (KR'96)*, Morgan Kaufmann, San Francisco, CA., pages 529–541.

[17] J. Dix and M. Müller (1994) Partial evaluation and relevance for approximations of the stable semantics. Z.W. Raz and M. Zemankova (eds) *Proc. ISMIS'94*, LNAI 869, pp. 511–520.

[18] L. Giordano and A. Martelli (1995) A logical characterization for truth maintenance systems with dependency-directed backtracking. *Computational Intelligence* **11**: 11–46.

[19] K. Inoue and C. Sakama (1994) On positive occurrences of negation as failure. In *Proc. 4th Int. Conf. on Principles of Knowledge Representation and Reasoning, KR'94*, Morgan Kauffman, San Francisco, CA., 1994, pages 293–304.

[20] V. Lifschitz (1994) Minimal belief and negation as failure. *Artificial Intelligence*, 70:53–72, 1994.

[21] V. Lifschitz and T. Woo (1992) Answer sets in general nonmonotonic reasoning (preliminary report). In *Proc. Third Int. Conf. on Principles of Knowledge Representation and Reasoning, KR'92*, Morgan Kauffman, 1992, pages 603–614.

[22] M. Müller and J. Dix (1993) Implementing semantics of disjunctive logic programs using fringes and abstract properties (extended abstract). In *Proc. Second Int. Workshop on Logic Programming and Nonmonotonic Reasoning*, L.M. Pereira and A. Nerode (eds.), MIT Press, Cambridge, MA, 1993, pages 43–59.

[23] T.C. Przymusinski (1988) On the declarative semantics of stratified deductive databases and logic programs. In J. Minker (ed) *Foundations of Deductive Databases and Logic Programming*, pp. 193–216, Morgan Kauffman, Los Altos, CA.

[24] T.C. Przymusinski (1990) The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae,* **13**: 445–464.

[25] T.C. Przymusinski (1991) Stable semantics for disjunctive programs *New Generation Computing*, **9**: 401–424.

[26] T.C. Przymusinski (1991) Semantics of disjunctive logic programs and deductive databases. In *Proc. Second Int. Conf. on Deductive and Object-Oriented Databases, DOOD'91*, Springer Verlag, pp. 85–107.

[27] T.C. Przymusinski (1995) Static semantics for normal and disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, **14**:323–357.

[28] T.C. Przymusinski (1995) Semantics of normal and disjunctive logic programs: A unifying framework. In J. Dix, L. Pereira, and T. Przymusinski, eds., *Proc. Workshop on Non-Monotonic Extensions of Logic Programming*, Santa Margherita Ligure, Italy, June 1994, pages 43–67, Springer Verlag (Lect. Notes in AI).

[29] J. S. Schlipf (1992). Formalizing a logic for logic programming. *Annals of Mathematics and Artificial Intelligence* **5**:279–302.

[30] J.-H. You and L. Y. Yuan (1994) A three-valued semantics for deductive databases and logic programs. *J. of Computer and System Sciences*, **49**: 334–361.

[31] A. Van Gelder, K.A. Ross and J.S. Schlipf (1991) The well-founded semantics for general logic programs. *J. ACM,* **38**: 620–650.

[32] C. Witteveen and W. van der Hoek (1995) Revision by communication. *Proc. Third International Conference on Logic Programming and Nonmonotonic Reasoning, LPNMR'95*, Lecture Notes in Artificial Intelligence, 928, pp. 189–202.

Proofs of the Main Results

We will give here proofs for all the new theorems stated in this part.

Theorem 2.2. *If P is a disjunctive program, then $SEM_P^{\vee}$ coincides with the disjunctive well-founded semantics (D-WFS) of P.*

Proof. We will show that the set of all minimal bisequents of the form $: b \Vdash c :$ (where $c \neq \emptyset$) that belong to the $\vee$-completion of a program P coincides with what was termed in [14] a *residual program* for P, that is, with the set of clauses that form the 'final reduct' of P with respect to the above transformations. In order to prove this fact, we need a number of auxiliary results.

In what follows, the elimination and reduction rules characterizing D-WFS will be called *BD-transformations*.

Auxiliary Lemma 1. *If P is reducible to P' using the BD-transformations, then P and P' have the same nonmonotonic $\vee$-completion.*

Proof. It is sufficient to show that each of the BD-transformations preserves the $\vee$-completion.

If P' is obtained from P by an application of Elimination of Tautologies or Elimination of Non-Minimal Rules, then the result immediately follows from the fact that P and P' generate the same biconsequence relation (due to Reflexivity and Monotonicity). If P' is obtained by GPPE, then the result follows from Theorem 3.3 in [12], since the nonmonotonic completion is defined as an extension of a circumscribed biconsequence relation.

Assume that an atom A does not belong to heads of the clauses from P, and P' is obtained from P by Positive Reduction with respect to A. Then it is easy to see that $\Vdash_{P'}$ is a negative extension of $\Vdash_P$. To be more exact, a bisequent $a : b \Vdash c : d$ belongs to $\Vdash_{P'}$ if and only if $a : b, A \Vdash_P c : d$. This implies, in particular, that (u, v) is a bimodel of $\Vdash_{P'}$ if and only if $(u, v \setminus \{A\})$ is a bimodel of $\Vdash_P$, and hence both biconsequence relations have the same bimodels (u, v) for which $A \notin v$. Now since A does not belong to heads of the rules of P and P', the corresponding circumscriptions of both these programs contain $A : \Vdash$, and hence their completions contain $\Vdash : A$ (by Negative Coherence). This means that bimodels of the resulting completions satisfy the condition $A \notin v$. Since $\Vdash_P$ and $\Vdash_{P'}$ have the same bimodels of this kind, these completions coincide.

Finally, let P' be obtained from P by Negative Reduction with respect to a rule $a \leftarrow$ from P. Then, similarly to the previous case, it is easy to show that $\Vdash_P$ and $\Vdash_{P'}$ have the same bimodels (u, v) for which $a \cap v \neq \emptyset$. But $\Vdash a :$ belongs to both these biconsequence relations, and hence it belongs to their respective $\vee$-completions. Consequently, both these completions contain also $: a \Vdash$ (due to Positive Coherence), and therefore the relevant bimodels satisfy the condition $a \cap v \neq \emptyset$. Hence both these completions coincide, since they are determined by the same bimodels. $\square$

As an immediate consequence of the above lemma, we obtain that if $res(P)$ is a residual set of clauses for P, then it is included in the $\vee$-completion of P.

A set of disjunctive program clauses P will be called *BD-irreducible* if no BD-transformation is applicable to it. Thus, any residual program, $res(P)$, will be irreducible in this sense. Note that due to GPPE, any such program should contain only *conditional facts*, that is, rules without positive literals in their bodies. The following result shows that the $\vee$-completion of such a program is conservative, that is, it does not contain new minimal conditional facts.

Auxiliary Lemma 2. *If P is a BD-irreducible program, then any minimal conditional fact belonging to the $\vee$-completion of P is already included in P.*

(A sketch of the proof). For an irreducible program P, let us define a biconsequence relation $\Vdash_*$ as follows:

$$a : b \Vdash_* c : d \quad \equiv \quad \text{either } a : b \Vdash_P c : d \text{ or } \ \Vdash_P b : \text{ or } d \not\subseteq H_P,$$

where H_P is a set of all atoms occurring in the heads of P. It can be shown that $\Vdash_*$ is indeed a biconsequence relation (the only non-trivial case is the the validity of the Negative Cut rule that depends on the irreducibility of P). Moreover, it is easy to check that $\Vdash_*$ is a negative extension of $\Vdash_P$, and bimodels of $\Vdash_*$ are precisely bimodels (u, v) of $\Vdash_P$ satisfying the conditions $\nVdash_P \overline{v}$ and $v \subseteq H_P$.

We are going to show that the circumscription of $\Vdash_*$, that is, $\Vdash^c_*$, is a $\vee$-coherent extension of $\Vdash^c_P$, and hence includes the $\vee$-completion of P[3].

Let us check the coherence of $\Vdash^c_*$. If $\Vdash^c_* a :$, then $\Vdash_* a :$, and hence $\Vdash_P a :$. By the definition of $\Vdash_*$ we immediately conclude $: a \Vdash_*$, and therefore $: a \Vdash^c_*$. Thus, Positive Coherence holds. Assume now that $\nVdash^c_* : A$, which is equivalent to $\nVdash_* : A$. Then $A \in H_P$ by the definition of $\Vdash_*$. Assume that $c, A \leftarrow \textbf{not}\, b$ is a clause of P where A occurs in the head. Then $: b \nVdash_P c :$ and $\nVdash_P b :$ due to the irreducibility of P. Hence $: b \nVdash_* c :$. Let (u, v) be a positively minimal bimodel of $\Vdash_*$ such that $b \subseteq \overline{v}$ and $c \subseteq \overline{u}$. Then $A \in u$, since $: b \Vdash_* c, A :$. Thus, A belongs to at least one minimal bimodel of $\Vdash_*$, and therefore $A : \nVdash^c_*$. Consequently, Negative Coherence also holds.

Since $\Vdash_*$ is a negative extension of $\Vdash_P$, we also have $\Vdash^c_P \subseteq \Vdash^c_*$ by Proposition 3.2 in [12]. Thus, $\Vdash^c_*$ is a coherent extension of $\Vdash^c_P$, and consequently it includes the completion of P, since the latter is a minimal coherent extension of $\Vdash^c$.

Assume now that $: b \Vdash c :$ is a minimal bisequent belonging to the $\vee$-completion of P. Then $: b \Vdash^c_* c :$ and hence $: b \Vdash_* c :$. By the definition of $\Vdash_*$ this implies that either $: b \Vdash_P c :$ or $\Vdash_P b :$. But if $\Vdash_P b :$ holds, we have ${}^c\Vdash^\vee_P b :$ and hence $: b\,{}^c\Vdash^\vee_P$ by Positive Coherence. Consequently, $: b \Vdash c :$ cannot be a minimal bisequent in ${}^c\Vdash^\vee$ (recall that $c \neq \emptyset$). Thus, $: b \Vdash_P c :$. It is easy to show that this can hold only if there is some rule $c_0 \leftarrow \textbf{not}\, b_0$ in P such that $b_0 \subseteq b$ and $c_0 \subseteq c$. But then $: b_0 \Vdash c_0 :$ belongs to the completion of P, and hence $b = b_0$ and $c = c_0$, since $: b \Vdash c :$ is minimal in the completion. Therefore $c \leftarrow \textbf{not}\, b$ belongs to P, and we are done. $\square$

Applying the above two lemmas to the residual program $res(P)$, we obtain that the $\vee$-completion of any disjunctive program P coincides with the $\vee$-completion of $res(P)$, and that the set of minimal conditional facts belonging to the latter completion is precisely $res(P)$. As a consequence, SEM^+_P is a set of disjunctions that are subsumed by disjunctive facts from $res(P)$,

[3]Actually, $\Vdash^c_*$ coincides with the completion of P, but we do not need this fact in the proof.

and hence coincides with the positive component of D-WFS. Not also that any nonmonotonic completion is determined by positively minimal bimodels and hence coincides with its own circumscription. This implies, in particular, that $A : \Vdash$ belongs to the completion if and only if there is no minimal bisequent of the form $: b \Vdash c :$ in the completion such that $A \in c$. Consequently, SEM_P^- in our case is a set of all atoms that do not belong to heads of the rules from $res(P)$. But this is precisely the set of negative literals belonging to D-WFS. Thus, SEM_P coincides with D-WFS. □

Theorem 3.1. *If $S(P)$ is a pure stationary completion of a disjunctive program P, then ${}^{c}\Vdash_P^{\{\vee,\wedge\}} = \Vdash^c_{S(P)}$.*

Proof. In order to show the inclusion of left into right, it is sufficient to demonstrate that $\Vdash^c_{S(P)}$ is a $\{\vee, \wedge\}$-coherent biconsequence relation including $\Vdash^c_P$.

If $\Vdash^c_{S(P)} a :$, then $S(P) \models_{min} \bigvee a$. By Proposition 4.1 from [26] this holds if and only if $S(P) \models_{min} \bigvee(\neg\, \mathbf{not}\, a)$, which implies $: a \Vdash^c_{S(P)}$. Similarly, $a : \Vdash^c_{S(P)}$ is equivalent to $S(P) \models_{min} \bigvee(\neg a)$ and hence to $S(P) \models \bigvee(\mathbf{not}\, a)$. But the latter is equivalent to $\Vdash_{S(P)} : a$ and hence to $\Vdash^c_{S(P)} : a$. Consequently, $\Vdash^c_{S(P)}$ is $\{\vee, \wedge\}$-coherent.

Proposition 4.1 from [25] implies also that the stationary completion is obtained by adding propositions of the form $\mathbf{not}\, A$ to a program. Consequently, the corresponding biconsequence relation $\Vdash_{S(P)}$ is a negative extension of $\Vdash_P$ and hence $\Vdash^c_P \subseteq \Vdash_{S(P)}$. Thus, the inclusion of left into right holds.

To show the inverse inclusion, we will prove the following

Auxiliary Lemma 3. *The set of $\{\vee, \neg, \sim\}$-formulas belonging to the $\{\vee, \wedge\}$-completion of P is a stationary expansion of P.*

Proof. Note first that any $\{\vee, \wedge\}$-completion is a circumscribed biconsequence relation (being an extension of $\Vdash^c_P$), and hence it coincides with its own circumscription.

Let P^* denote the set of $\{\vee, \neg, \sim\}$-formulas belonging to the $\{\vee, \wedge\}$-completion of P. Note that the completion coincides with $\Vdash_{P^*}$.

Clearly, P is included in P^*. Moreover, if $P^* \models_{min} \bigvee a$, then $\Vdash_{P^*} a :$, and consequently $: a \Vdash_{P^*}$ be Positive Coherence. Hence, $P^* \models \bigvee(\mathbf{not}\, \neg a)$. Similarly, if $P^* \models \bigvee(\neg a)$, then $a : \Vdash_{P^*}$ and consequently $\Vdash_{P^*} : a$ by Negative Coherence. Thus, $P^* \models \bigvee(\mathbf{not}\, a)$. Therefore P^* includes P and $\{\mathbf{not}\, F \mid P^* \models_{min} \neg F\}$, where F is a pure positive or negative conjunction.

To show the inverse inclusion, assume that A is a classical consequence of $P \cup \{\mathbf{not}\, F \mid P^* \models_{min} \neg F\}$. Then there exists a finite set of positive or negative conjunctions F_i such that $P^* \models_{min} \neg F_i$ and $P \models \bigwedge(\mathbf{not}\, F_i) \rightarrow A$. Since P is included in P^*, we have $P^* \models \bigwedge(\mathbf{not}\, F_i) \rightarrow A$. But $P^* \models \mathbf{not}\, F_i$ and hence $P^* \models A$. $\square$

Since the stationary completion is a least stationary expansion, we immediately obtain that it is included in the $\{\vee, \wedge\}$-completion of a program. Consequently, any bimodel (u, v) of the $\{\vee, \wedge\}$-completion is also a bimodel of $\Vdash_{S(P)}$. Hence there must exist a positively minimal bimodel (u_0, v) of $\Vdash_{S(P)}$ such that $u_0 \subseteq u$. However, any positively minimal bimodel of $\Vdash_{S(P)}$ is also a bimodel of the $\{\vee, \wedge\}$-completion (since, as we have shown above, the $\{\vee, \wedge\}$-completion is included in $\Vdash^c_{S(P)}$). But any bimodel of the $\{\vee, \wedge\}$-completion is positively minimal, and hence $u_0 = u$. Therefore any bimodel of the $\{\vee, \wedge\}$-completion is also a positively minimal bimodel of $\Vdash_{S(P)}$. Thus, $\Vdash^c_{S(P)}$ is included in ${}^c\Vdash_P^{\{\vee,\wedge\}}$. This completes the proof of the theorem. $\square$

Theorem 3.3. *Let $\Vdash$ be an affirmative biconsequence relation. A biconsequence relation $\Vdash_0$ is a negative, $\{\vee, \neg\}$-coherent and regular extension of $\Vdash^c$ if and only if there is a stationary expansion $\Vdash_*$ of $\Vdash$ such that $\Vdash_0 = \Vdash^c_*$.*

Proof. Let $\Vdash_0$ be a negative, $\{\vee, \neg\}$-coherent and regular extension of $\Vdash$. We take $\Vdash_*$ to be the least biconsequence relation containing $\Vdash$ and all bisequents of the form $\Vdash: A$ from $\Vdash_0$. Due to the fact that $\Vdash_0$ is a negative extension of $\Vdash^c$, it is easy to show that $\Vdash^c_*$ and $\Vdash_0$ must have the same bimodels, and hence they coincide. As a result, $\Vdash_*$ is also the least biconsequence relation containing $\Vdash$ and all bisequents of the form $\Vdash: A$ from $\Vdash^c_*$. But $\Vdash_0$ is both regular and coherent, and hence $\Vdash^c_*: A$ iff $\Vdash^c_* \neg A :$, for any $\{\vee, \neg\}$-proposition A. Thus, $\Vdash_*$ is a stationary expansion of $\Vdash$. This concludes the proof from left to right.

Assume now that $\Vdash_*$ is a stationary expansion of $\Vdash$. Clearly, any stationary expansion is a negative extension of $\Vdash$. Consequently, $\Vdash^c_*$ is a negative extension of $\Vdash^c$ by Proposition 3.2 in [12]. Moreover, if $\Vdash^c_* \neg A$, for some $\{\vee, \neg\}$-proposition A, then $\Vdash_*: A$ and hence $\Vdash^c_*: A$. Thus, $\Vdash^c_*$ is $\{\vee, \neg\}$-coherent. Finally, if a bisequent $\Vdash: A$ belongs to $\Vdash^c_*$, there must exist a finite set of $\{\vee, \neg\}$-propositions $\{A_1, A_2, \ldots, A_n\}$ such that $\Vdash^c_* \neg A_i :$, for any $1 \leq i \leq n$, and $: A_1, A_2, \ldots, A_n \Vdash: A$. Consequently, $: A_1, A_2, \ldots, A_n \Vdash^c: A$. As is shown in [10], if $\Vdash$ is an affirmative biconsequence relation, then both $\Vdash$ and $\Vdash^c$ are $\{\vee, \neg\}$-regular. Hence $A :\Vdash^c A_1, A_2, \ldots, A_n :$, and therefore

$A :\Vdash^c_* A_1, A_2, \ldots, A_n :$. But $\Vdash^c_* \neg A_i :$, for any $1 \leq i \leq n$, and consequently $A :\Vdash^c_*$. Thus, $\Vdash^c_*$ is $\{\vee, \neg\}$-regular. This concludes the proof of the theorem. □

Theorem 3.5. *If P is a finite normal program, then negative parts of bimodels of any stationary expansion of $\Vdash_P$ form a stable class of P and any stable class of P is determined in this way by some stationary expansion of $\Vdash_P$.*

Proof. In view of Theorem 3.3, the circumscription of any stationary expansion $\Vdash_*$ is a negative, $\{\vee, \neg\}$-coherent and regular extension of $\Vdash^c_P$. Note also that any negative part of a bimodel of $\Vdash^c_*$ is also a positive part of some bimodel of $\Vdash^c_*$ and vice versa (due to regularity and coherence). Consequently, the above correspondence amounts to correspondence between stable classes and *positive* parts of bimodels of negative, $\{\vee, \neg\}$-coherent and regular extensions of $\Vdash^c_P$. Note also that if P is normal, then any extension of $\Vdash^c_P$ can be easily shown to be a negative extension. So, we need to establish the correspondence between stable classes and (positive parts of bimodels of) coherent and regular extensions of $\Vdash^c_P$.

Let S be a stable class of a normal program P. Note that in the finite case any stable class of a program should be finite, since all its elements are *minimal* models of some reducts of P.

We define a biconsequence relation $\Vdash^S_P$ determined by S as follows:

> $a : b \Vdash^S_P c : d$ iff $c \cap s \neq \emptyset$ or $d \cap \overline{s}' \neq \emptyset$, for any pair s, s' from S such that $s = \mathbf{F}(s')$, $a \subseteq s$ and $b \subseteq \overline{s}'$.

As we mentioned, a pair (s, s') satisfies the condition $s = \mathbf{F}(s')$ iff it is a positively minimal bimodel of $\Vdash_P$. Consequently, $\Vdash^P_S$ is determined by some set of positively minimal bimodels and hence is an extension of the corresponding circumscribed biconsequence relation, that is, $\Vdash^c_P \subseteq \Vdash^S_P$.

Now we will show that $\Vdash^S_P$ is both coherent and regular with respect to $\{\vee, \neg\}$.

If $a :\nVdash^S_P b :$, then there exists $s \in S$ such that $a \subseteq s$, $b \subseteq \overline{s}$. But then $\mathbf{F}(s) \in S$, because S is a stable class, and we obtain $: b \nVdash^S_P : a$, since there is a pair $(\mathbf{F}(s), s)$ such that $a \subseteq s$ and $b \subseteq \overline{s}$. Thus, $\Vdash^S_P$ satisfies $\{\vee, \neg\}$-regularity.

In the other direction, if $: b \nVdash^S_P : a$, then there exists $s \in S$ such that $a \subseteq s$, $b \subseteq \overline{s}$. But then $s = \mathbf{F}(s')$, for some $s' \in S$, and we immediately obtain $a :\nVdash^S_P b :$. Hence, $\{\vee, \neg\}$-coherence also holds.

A set u coincides with a positive part of some bimodel of $\Vdash_P^S$ iff $u :\nVdash_P^S \overline{u} :$. The latter condition means that, for any finite sets a, b such that $a \subseteq u$ and $b \subseteq \overline{u}$, there is $s \in S$ such that $a \subseteq s$ and $b \subseteq \overline{s}$. Since S is finite, this can hold only if u coincides with some set from S. Consequently, the positive parts of all bimodels of $\Vdash_P^S$ coincide with elements of S.

Assume now that $\Vdash^*$ is a $\{\vee, \neg\}$-coherent and regular extension of $\Vdash_P^c$ and S is a set of all positive parts of bimodels from $\Vdash^*$. Since P is finite, it is easy to show that all bimodels of $\Vdash^*$ should be positively minimal bimodels of $\Vdash_P$ (see [9]). Moreover, $u \in S$ implies that there exists v such that (u, v) is a bimodel of $\Vdash^*$, and hence $: \overline{v} \nVdash^* : v$ and $u = \mathbf{F}(v)$. Due to coherence we have $v :\Vdash^* \overline{v} :$, and hence $v \in S$. In the other direction, if $u \in S$, then $u :\nVdash^* \overline{u}$. By $\{\vee, \neg\}$-regularity, we infer that $: \overline{u} \nVdash : u$, and consequently there is v such that (v, u) is a bimodel of $\Vdash^*$. Therefore, $v \in S$. Moreover, since (v, u) is a positively minimal model of $\Vdash_P$, we have $v = \mathbf{F}(u)$, and hence $S = \mathbf{F}(S)$, that is, S is a stable class of P. □

Theorem 4.1. *(u, v) is a p-stable bimodel of a biconsequence relation $\Vdash$ if and only if $u \subseteq v$ and both (u, v) and (v, u) are positively minimal bimodels of $\Vdash$.*

Proof. Follows immediately from the fact that the invariant completion of a biconsequence relation is determined by all the invariant bimodels of the circumscribed biconsequence relation, while all the bimodels of the latter are positively minimal. □

Theorem 4.2. *If P is a normal program, then p-stable bimodels of $\Vdash_P$ are exactly partial stable models of P.*

Proof. We will make use of a representation of normal logic programs in terms of default consequence relations suggested in [8] (see also [5]). As is shown in these papers, partial stable models of a normal program can be characterized as consistent interpretations (i, j) such that i is the set of atoms provable by taking $\mathbf{not}\, j$ as a set of negative premises, while $\overline{j}$ is the set of atoms provable if $\mathbf{not}\, \overline{i}$ is taken as a set of negative premises. Using our present terminology, this amounts to saying that both $(i, \overline{j})$ and $(\overline{j}, i)$ are positively minimal bimodels. Now the result follows immediately from the preceding theorem. □

Theorem 4.3. *If P is a disjunctive program, then any p-stable bimodel of $\Vdash_P$ is a disjunctive partial stable model of P.*

Proof. We are assuming that the reader is familiar with a definition of disjunctive partial stable models, given in [24]. To begin with, we will gradually translate Przymusinski's key notions into our terminology.

- A partial (three-valued) interpretation I can be identified with a consistent pair (u, v), where u (v) is a set of atoms that are true (respectively, not false) in I.

- Recursive definitions of the connectives $\wedge, \vee, \sim$ (but not $\leftarrow$), given in [24], coincide with the corresponding definitions of these connectives in our framework.

- An interpretation $I = (u, v)$ is a *model* of a program in the sense of [24] if and only if both (u, v) and (v, u) are bimodels of the program in our sense (this correspondence is due to a special meaning assigned to the connective $\leftarrow$ by Przymusinski).

- A model (u, v) is *minimal* iff there there is no other model (u', v') such that $u' \subseteq u$ and $v' \subseteq v$.

- An interpretation (u', v') is a model of the *quotient* $\dfrac{P}{I}$ of a program P modulo an interpretation $I = (u, v)$ iff both (u', v) and (v', u) are bimodels of P.

Now, a disjunctive partial stable model is defined as any interpretation $I = (u, v)$ that coincides with a minimal model of $\dfrac{P}{I}$. Making use of the above correspondences, it is easy to show that this definition is equivalent to the condition that (u, v) is an invariant bimodel of the program such that there is no other interpretation (u', v') for which $u' \subseteq u$, $v' \subseteq v$, and both (u', v) and (v', u) are bimodels. It immediately follows from this description that if (u, v) is consistent and both (u, v) and (v, u) are positively minimal bimodels, then (u, v) is a partial stable model in the sense of Przymusinski. $\square$

Theorem 4.4. *Let $\Vdash^{cc}$ be the least ordered invariant biconsequence relation containing $\Vdash^{c}$. Then a $\{\vee, \sim\}$-proposition belongs to all p-stable bimodels of $\Vdash$ iff it is provable in $\Vdash^{cc}$.*

Proof. It is easy to show that bimodels of $\Vdash^{cc}$ are exactly consistent or complete invariant bimodels of $\Vdash^{c}$. Note also that if a $\{\vee, \sim\}$-proposition A is true in a consistent bimodel (u, v), it is also not false with respect to it (this can be shown by induction on the complexity of A). Consequently, if A is true in all p-stable bimodels of $\Vdash$, it will be also true in all *complete* invariant bimodels of $\Vdash^c$, and therefore in all bimodels of $\Vdash^{cc}$. This means that A will be provable in $\Vdash^{cc}$. In the other direction, if A is provable in $\Vdash^{cc}$, then it is obviously true in all consistent invariant bimodels of $\Vdash^c$, that is, in all p-stable bimodels. $\square$

Theorem 4.5. *1. A $\{\vee, \sim\}$-proposition A belongs to some p-stable bimodel of a biconsequence relation $\Vdash$ iff it can be classically true with respect to $\Vdash^{cc}$, that is, if $A : \nVdash^{cc} : A$.*

2. A $\{\vee, \sim\}$-proposition A is false in some p-stable bimodel of a biconsequence relation $\Vdash$ iff it can be classically false with respect to $\Vdash^{cc}$, that is, if $: A \nVdash^{cc} A :$.

Proof. As was mentioned in the proof of the preceding theorem, if a $\{\vee, \sim\}$-proposition A is true in a consistent bimodel (u, v), it is also not false with respect to it. Thus, it is classically true in (u, v). Therefore, if A is true in some p-stable bimodel, we will obtain that it is classically true in it and hence $A : \nVdash^{cc} : A$. Similarly, if A is false in a p-stable bimodel, it should be classically false in it, and hence $: A \nVdash^{cc} A :$.

Assume now that $A : \nVdash^{cc} : A$ holds for some $\{\vee, \sim\}$-proposition A. Then A should be true in some bimodel (u, v) of $\Vdash^{cc}$. But either (u, v) or (v, u) should be consistent, and hence be a p-stable bimodel. Consequently, A is true in some p-stable bimodel. The proof for falsity is completely analogous. $\square$

Theorem 4.6. *A $\{\vee, \sim\}$-semantics for any biconsequence relation coincides with a restriction of the $\{\vee, \neg, \sim\}$-semantics to $\{\vee, \sim\}$-propositions.*

Proof. We need to show that if a $\{\vee, \sim\}$-proposition is provable or refutable in the invariant completion, it is already provable or refutable in the $\{\vee, \sim\}$-completion. In other words, we need to show that adding the Invariance rule to the $\{\vee, \sim\}$-completion does not change the set of provable or refutable $\{\vee, \sim\}$-propositions.

It can be shown (see [5]) that $\{\vee, \sim\}$-coherence rules amount to the following structural rules:

$$\frac{\Vdash a : b}{b : a \Vdash} \qquad \frac{a : b \Vdash}{\Vdash b : a}$$

A *k-order* on bimodels is defined as follows:

$$(u, v) \leq_k (u', v) \quad \equiv \quad u \subseteq u' \text{ and } v' \subseteq v.$$

Then it can be shown that above structural rules correspond to semantic conditions saying that bimodels that are minimal or maximal in the k-order are invariant. Furthermore, since $\{\vee, \sim\}$-propositions are persistent with respect to the k-order, it is easy to check that such a proposition is provable in a biconsequence relation if and only if it is true in all k-minimal bimodels. Similarly, it is refutable iff it is not true in all k-maximal bimodels. Consequently, adding the Invariance rule does not change the set of provable or refutable $\{\vee, \sim\}$-propositions, since the restriction to invariant bimodels does not change in this case the set of k-maximal and k-minimal bimodels. This completes the proof. $\square$

Theorem 4.7. *If $\Vdash$ is a locally finite biconsequence relation, then $\Vdash_{K^i(\Vdash)}$ coincides with the invariant completion of $\Vdash$.*

Proof. Let $\Vdash^i$ denote the invariant completion of $\Vdash$. Clearly, $K^i(\Vdash)$ is included in $\Vdash^i$, and hence $\Vdash_{K^i(\Vdash)} \subseteq \Vdash^i$. In addition, it is easy to show that if a bisequent $a : b \Vdash c : d$ is provable from $K^i(\Vdash)$, then $d : c \Vdash b : a$ is also provable from this set (by induction on proofs). Consequently, Invariance is an admissible rule in $\Vdash_{K^i(\Vdash)}$. Thus, $\Vdash_{K^i(\Vdash)}$ is an invariant biconsequence relation containing the circumscription of $\Vdash$, and therefore $\Vdash^i \subseteq \Vdash_{K^i(\Vdash)}$. This completes the proof. $\square$

Theorem 5.2. *Stable models of any program P coincide with bimodels of the nonmonotonic completion of P in the full classical language.*

Proof. As can be seen, bimodels of the classical completion of a biconsequence relation $\Vdash$ are precisely positively minimal bimodels of $\Vdash$ of the form (i, i). As was shown in [6], such sets correspond to nonmonotonic models of Lifschitz' Modal Logic of Belief and Negation as Failure (MBNF), as defined in [20]. Then the result immediately follows from the the representation of the stable semantics in the framework of MBNF, as given in [20]. $\square$