

# Semantic Forcing in Disjunctive Logic Programs

Marina De Vos\*and Dirk Vermeir  
Dept. of Computer Science  
Free University of Brussels, VUB  
Pleinlaan 2, Brussels 1050, Belgium

## Abstract

We propose a semantics for disjunctive logic programs, based on the single notion of forcing. We show that the semantics properly extends, in a natural way, previous approaches. A fixpoint characterization is also provided. We also take a closer look at the relationship between disjunctive logic programs and disjunctive-free logic programs. We present certain criteria under which a disjunctive program is semantically equivalent with its disjunctive-free (shifted) version.

## 1 Introduction

Disjunctive logic programs are a generalization of logic programs (i.e. Prolog) where disjunctions are allowed in the heads of the rules and negation may occur in the bodies of those rules. Allowing disjunctions in the head, permits a natural specification style in cases where a choice must be made, as in the following example.

Consider a situation where a person is either healthy or sick. We know that, if she is sick, she suffers from either a cold or bronchitis. Moreover, she should go to work unless she has bronchitis.

The knowledge above can be succinctly represented by the following disjunctive logic program .

$$\begin{array}{llll} \textit{healthy} & \vee & \textit{sick} & \leftarrow \\ \textit{cold} & \vee & \textit{bronchitis} & \leftarrow \quad \textit{sick} \\ & & \textit{work} & \leftarrow \neg \textit{bronchitis} \end{array}$$

Intuitively, there are three possibilities:

1. The person is healthy, which means that she is not sick. So we can conclude that she has neither a cold nor a bronchitis and she can go to work.

---

\*Wishes to thank the FWO for its support.

2. The person is sick and by that not healthy but she suffers from a cold and not bronchitis. So she can go to work.
3. The person is sick, not healthy, and has a bronchitis instead of a cold. Hence she has to stay home from work.

The total semantics of the example disjunctive logic program is given by the following three total models, each corresponding to one of the above situations.

$$\begin{aligned}
 M_1 &= \{healthy, \neg sick, \neg cold, \neg bronchitis, work\} \\
 M_2 &= \{\neg healthy, sick, cold, \neg bronchitis, work\} \\
 M_3 &= \{\neg healthy, sick, \neg cold, bronchitis, \neg work\}
 \end{aligned}$$

One should note that the usual way of representing disjunction in ordinary logic programs, using negation, does not always work. This can be seen from the, admittedly contrived, example program.

**Example 1**

$$\begin{array}{lcl}
 a \vee b & \leftarrow & \\
 b & \leftarrow & a \\
 a & \leftarrow & b
 \end{array}$$

Representing the disjunction in the first rule by two disjunction-free rules  $b \leftarrow \neg a$  and  $a \leftarrow \neg b$  does not work as this does not yield any total stable model while the disjunctive version has the natural total stable model [LRS97]  $\{a, b\}$ .

Much research has been done to extend semantics for normal logic programs to disjunctive logic programs and some specific new approaches have been proposed as well [BG94]. One widely acknowledged semantics is the extension to disjunctive logic programs of the stable model semantics of [GL88]. According to this semantics [Prz91, LRS97] a disjunctive logic program may have several alternative models (but possibly none), each corresponding to a possible view of reality, where every proposition is either true or false. This is also the semantics used in the first example. In [LRS97], stable models are expressed using unfounded sets.

Another common semantics for disjunctive logic programs is the three-valued stable semantics introduced by [Prz91]. With three-valued stable models, it is possible for some propositions to remain unknown, i.e. they are neither true nor false. In [ELS97] a characterization of these three-valued stable models is given in terms of unfounded sets, which are different from the unfounded sets in [LRS97]. Also the well-founded semantics for normal logic programs has been extended to disjunctive logic programs, albeit in different ways: [BLM89] uses a generalization of the closed world assumption while [Ros89] allows disjunctions of literals in the interpretation. Both approaches embody the idea that the well-founded model contains those literals, possibly in a disjunction ([Ros89]), which are true or false in every minimal model of the program.

A different approach, called “possible model semantics”, was developed in [Sak89] and extended in [SI94]. It supports treating disjunction as either inclusive or exclusive, depending on the problem at hand. Unlike other semantics, the possible-model semantics is able to detect whether a disjunction should be considered inclusive or exclusive. So this semantics can treat two different disjunctions in the same program totally different, where the other semantics are fixed to either inclusive or exclusive.

The general principle of the possible model semantics is to transform a program in several disjunctive-free programs and calculate their minimal models. Afterwards one keeps those atoms to be true that are present in every possible model. [Cha93] proposes a “possible world semantics” which turns out to be equivalent to the possible model semantics. Another interesting semantics for disjunctive logic programs is the regular extension semantics [YYG98]. Regular extensions are based on maximal alternating fixpoints of the  $\mathcal{F}_P$ -operator. Given a set of “semi-negative atoms”<sup>1</sup>,  $\mathcal{F}_P$  computes all the atoms that are not derivable from the program using the semi-negative information. Then, it returns those derived atoms in semi-negative form. For the derivation, [YYG98] uses the rules from first-order logic augmented with a rule of Assumption Commitment (RAC): given  $not\_a$  and  $a \vee b$  one can derive  $b$ . This allows them to derive disjunctions as exclusive as possible. A set  $S$  of semi-negative information is an alternating fixpoint if  $\mathcal{F}_P^2(S) = S$ .

In this paper, we develop a new model based semantics for disjunctive logic programs, based on a single natural notion of “forcing”. Intuitively, a set  $A$  of atoms is forced by interpretation  $I$  if  $I$  must (be made to) contain part of  $A$  in order to be (come) deductively closed. Put in another way,  $I$  forces those sets of atoms containing some elements that have a good reason to be considered true.

The main results of this paper are summarized as follows :

- Using the notion of forcing, it is natural to consider those sets that are not forced by an interpretation nor by any of its extensions (such sets are “negatively forced”). It turns out that these sets can be characterized by a property that strongly resembles the notion of unfounded set from [LRS97]. In fact, the property coincides with the version in [LRS97] in all but so-called “unnatural” programs, and it appears from examples that our notion is more natural for such programs.
- We call those sets that are not forced by an interpretation assumption sets. It turns out that interpretations are “self-enforcing” iff they are free of assumptions. It can be shown that assumption sets are equivalent to a similar notion defined in [LVZ91] for disjunctive-free programs.
- Based on the above, we define a forced model as an assumption-free set of atoms  $M$  that does not force its complement, i.e. a maximal self-enforcing positive interpretation. We show that such sets are indeed models, i.e. deductively closed; and we provide a fixpoint characterization. It turns out that total forced models are exactly the set of stable models of [LRS97]. For disjunctive-free programs,

---

<sup>1</sup>[YYG98] do not use classical negation, instead they replace negative literals  $\neg p$  by atoms  $not.p$ .

forced models recover most other proposed semantics, from well-founded to partial stable semantics.

- Finally we will see that a weaker form of forcing can be used to detect which disjunctive logic programs can be converted with the shift-operator of [LRS97] to equivalent disjunctive-free programs, thus shedding more light on the power of disjunctive rules.

The rest of this paper is organized as follows. In Section 2, we give a short introduction to disjunctive logic programs. Forcing is introduced in Section 3. In Section 4, we investigate the relationship between unfounded sets and forcing. Section 5 is dedicated to the extension of the notion of assumption sets to disjunctive logic programs. Section 6 presents forced models for disjunctive logic programs. Several equivalent characterizations, e.g. based on fixpoints, are given. Section 7 relates our work with other approaches. We investigate the relationship between stable and forced models, forced models for disjunctive-free programs, and the relationship between logic programs and disjunctive-free programs generated with the shift-operator.

## 2 Preliminaries

The terms of our language are inductively defined. A variable or a constant is a *term*. A function symbol using terms as arguments is a term. An *atom* is a formula  $a(t_1, \dots, t_n)$ , where  $a$  is a predicate symbol and the arguments  $t_1, \dots, t_n$  are terms. A *literal* is either an atom  $p$  or a negated atom  $\neg p$ . The literals  $p$  and  $\neg p$  are each other's complement.

Where  $l$  is any literal, we denote its complement by  $\neg l$ . This notation is extended to sets so that for a set  $X$  of literals,  $\neg X$  denotes  $\{\neg l \mid l \in X\}$ .

**Definition 1** A *disjunctive logic program* [Prz91, RM95, LRS97] is a set of rules of the form  $H \leftarrow B$  where  $H$ , the **head** of the rule, is a finite nonempty set of atoms<sup>2</sup> and  $B$ , the **body** of the rule, is a finite set of literals.

A term, an atom, a literal, a rule or a program is called **ground** if no variables appear in it.

For a program  $P$  we use  $\mathcal{U}_P$  to denote its **Herbrand universe** i.e. the set of ground terms that use the function symbols and the constants appearing in  $P$ . For a program  $P$  we use  $\mathcal{B}_P$  to denote its **Herbrand base**, i.e. the set of all possible ground atoms that can be constructed from the predicates used in  $P$  and the terms occurring in  $\mathcal{U}_P$ .

A *ground instance* of a rule  $r$  is obtained from  $r$  by replacing every variable by a term from  $\mathcal{U}_P$ . The *grounded version* of  $P$ , denoted  $\text{ground}(P)$ , is the program obtained by replacing every rule by all its ground instances.

<sup>2</sup>Note that we hereby exclude constraints, i.e. rules with empty head.

If  $r$  is a rule in a disjunctive logic program  $P$ , we will use  $H_r$  to denote its head and  $B_r$  to denote its body.

Since the order of the atoms or literals in the head or body of (the usual definition of) a rule does not influence the semantics, we simply define them as sets.

Intuitively, the head of the rule should be interpreted as a disjunction while the body should be read as a conjunction. We will use the usual notation where  $a \vee b \leftarrow c, d$  stands for  $\{a, b\} \leftarrow \{c, d\}$  when considering concrete programs.

A grounded set  $X$  of literals is *consistent* if  $X \cap \neg X = \emptyset$ .

For any grounded set of literals  $X$  and a disjunctive logic program  $P$ , we will use  $\overline{\overline{X}}$  to denote the complement of  $X$  w.r.t.  $\mathcal{B}_P \cup \neg\mathcal{B}_P$ , i.e.  $\overline{\overline{X}} = (\mathcal{B}_P \cup \neg\mathcal{B}_P) \setminus X$ . The *positive complement* of  $X$ , denoted  $\overline{X}$  is defined by  $\overline{X} = \mathcal{B}_P \setminus X$ . Suppose a program  $P$  such that  $\mathcal{B}_P = \{a, b\}$ , then the positive complement of  $\{\neg a\}$  equals  $\{a, b\}$  while its complement is  $\{a, b, \neg b\}$ .

From now on we will always work with the grounded version,  $ground(P)$ , of a program  $P$  (note that  $ground(P)$  may be infinite). This will have no effect on any of the definitions or properties presented in this paper.

**Definition 2** Let  $P$  be a disjunctive logic program. A (**partial**) **interpretation** of  $P$  is any consistent subset of  $\mathcal{B}_P \cup \neg\mathcal{B}_P$ . The set of all interpretations of  $P$  is denoted by  $\mathcal{I}_P$ .

For a partial interpretation  $I$  of  $P$ , we use  $I^+$  to denote its positive part, i.e.  $I^+ = I \cap \mathcal{B}_P$ . Similarly, we use  $I^-$  to denote the negative part of  $I$ , i.e.  $I^- = \neg(I \cap \neg\mathcal{B}_P)$  (note that  $I^- \subseteq \mathcal{B}_P$ ). The undefined part  $I^\circ$  of  $P$  is defined as  $I^\circ = \mathcal{B}_P \setminus (I^+ \cup I^-)$ .

An interpretation  $I$  is **total** iff  $I^+ \cup I^- = \mathcal{B}_P$ .

The following is a simple local restriction on disjunctive logic programs.

**Definition 3** Let  $P$  be a disjunctive logic program. A rule  $r \in P$  is called **natural** if  $\neg B_r \cup H_r$  is consistent.  $P$  is **natural** iff all the rules in  $P$  are natural.

The **natural version**  $P_c$ <sup>3</sup> of a disjunctive logic program  $P$  is obtained by removing unnatural rules from  $P$ , i.e.

$$P_c = P \setminus \{r \in P \mid \neg B_r \cup H_r \text{ is inconsistent}\}$$

Atoms in  $\mathcal{B}_P \setminus \mathcal{B}_{P_c}$  are called **doomed** since they appear only in unnatural rules.

Thus, in a natural program we forbid rules such as  $a \leftarrow a$  and  $b \leftarrow a, \neg a$ . This is not a serious restriction since, as we shall see<sup>4</sup>, such rules cannot contribute to a model under any intuitive notion of semantics.

<sup>3</sup>The notion  $P_c$  stand for consistent version of  $P$ , the historical name for natural version. We had to change it because [Sak89, SI94] introduced also a consistent program.

<sup>4</sup>See e.g. Proposition 22.

One reasonable demand on a model is that it is deductively closed, i.e. all applicable rules are applied. This is formalized in the following definition.

**Definition 4** Let  $P$  be a disjunctive logic program. An interpretation  $I$  satisfies<sup>5</sup> a rule  $r \in P$  iff  $(H_r \cap I) \neq \emptyset$  whenever  $B_r \subseteq I$ . An interpretation  $I$  is **deductively closed** iff it satisfies all the rules in  $P$ .

In the following example, the interpretation  $I = \{a, b\}$  is not deductively closed.

$$\begin{array}{lcl} c \vee d & \leftarrow & b \\ a & \leftarrow & \\ b & \leftarrow & \end{array}$$

Note that  $I$  can be extended to a deductively closed interpretation by simply adding  $c$  or  $d$  to the interpretation  $I$ .

### 3 Forcing

In this section, we introduce the main notion of this paper, namely forcing. The choice for the term “forcing” is inspired by a similar notion (with the same name) from classical model theory [Kei91], that was originally proposed by Robinson [Rob70].

**Definition 5** Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation of  $P$ . We say that  $I$  **forces** a set of atoms  $A \subseteq \mathcal{B}_P$ , denoted  $I \Vdash_P A$  iff there exists a rule  $r \in P$  such that  $H_r \cap A \neq \emptyset$ ,  $B_r \subseteq I$ ,  $B_r \cap A = \emptyset$  and  $(H_r \setminus A) \cap I = \emptyset$ .

For  $A \subseteq \mathcal{B}_P$ , we have that  $I$  **negatively forces**  $A$ , denoted as  $I \Vdash_P \neg A$  iff no extension of  $I$  forces  $A$ , i.e.

$$\forall J \in \mathcal{I}_P \cdot J \supseteq I \Rightarrow J \not\Vdash_P A$$

Intuitively, a set of atoms  $A$  is forced by an interpretation  $I$  if a part of  $A$  must be in  $I$  (or be added to it) in order for  $I$  to be deductively closed. This obligation stems from the simplest of reasoning mechanism, namely (disciplined) rule application (“modus ponens”). Moreover, for  $A$  to be genuinely forced by a rule  $r$ ,  $r$  must be applicable ( $B_r \subseteq I$ ) and have a literal from  $A$  as consequence ( $H_r \cap A \neq \emptyset$ ). To avoid circular inferences, the applicability of  $r$  may not rely on the presence of (part of)  $A$  ( $B_r \cap A = \emptyset$ ). Moreover, rule application must be parsimonious i.e.  $r$  may not have been applied already for a consequence outside of  $A$  ( $(H_r \setminus A) \cap I = \emptyset$ ).

The definition of “negatively forcing” closely follows the original one in [Kei91] (where interpretations play the role of “conditions” in [Kei91]).

Intuitively, a set  $A$  is negatively forced if there is no need to adopt any element of  $A$

<sup>5</sup>Note that this notion of rule satisfaction differs from the three-valued one (see Definition 12).

in  $I$  in order for  $I$  to be (come) deductively closed. Moreover, even if  $I$  were to be extended (e.g. by adding elements from forced sets),  $A$  would still be "superfluous" in the above sense.

Roughly then, a set  $A$  is negatively forced, if there is no good reason (and nor will there be) to adopt any of its elements. This strongly resembles<sup>6</sup> the intuition behind "unfounded set" and the connection between the two notions will be investigated in Sect. 4. Consistent with the notation, we will often say that " $I$  forces  $\neg A$ " when  $I \Vdash_P \neg A$ .

**Example 2** Consider the following disjunctive logic program  $P$ :

$$\begin{array}{l} a \vee b \leftarrow c, \neg d \\ \phantom{a \vee b} c \leftarrow \neg e \\ d \vee e \leftarrow \phantom{c, \neg e} \end{array}$$

and consider the following interpretation  $I$ :

$$I = \{c, a, \neg d, e\}$$

Then,  $I$  forces the following sets:

$$I \Vdash_P \{a\}, \{e\}, \{a, b\}, \{a, e\}, \{a, d\}, \{d, e\}, \{e, b\}, \{e, c\}, \\ \{a, b, e\}, \{a, d, e\}, \{b, d, e\}, \{c, d, e\}, \{b, c, e\}, \{a, b, c, e\}, \\ \{a, b, d, e\}, \{a, c, d, e\}, \{b, c, d, e\}, \{a, b, c, d, e\}.$$

Note that  $I$  is already deductively closed, and that each set  $A$  that is forced by  $I$  has a non-empty intersection with  $I$ :  $I \cap A \neq \emptyset$ . In Proposition 2 will prove that this is generally true.

Still, intuitively,  $I$  is not a good model, e.g. because it contains atoms, such as  $c$ , that can be dropped without losing the deductively closed property of  $I$ .

$I$  negatively forces the following sets:

$$\{b\}, \{c\}, \{d\}, \{b, c\}, \{b, d\}, \{b, c, d\}, \{a, c\}, \{a, b, c\}, \{a, c, d\}, \{a, b, c, d\}$$

Hence,  $I$  contains quite a few unjustified atoms. In fact, only  $\{e\}$  is justified in that it is forced and does not contain any negatively forced atoms.

Note that a set may be forced without any of its elements being forced as in the following example.

**Example 3** Consider the program  $P$  containing the single rule

$$a \vee b \vee c \leftarrow$$

If  $I = \{b, c\}$  then  $I \Vdash_P \{a, b, c\}$  while  $I \not\vdash_P x$  for any  $x \in \{a, b, c\}$ .

<sup>6</sup>However, negatively forced sets are **not** identical to (traditionally) unfounded sets ([LRS97]). In fact, we believe the notion of negatively forcing to be more fundamental: in order to define "what cannot possibly be true", one naturally first defines "what must be true", as in [Kei91].

Conversely, if an atom (singleton) is forced, it may be that a set containing this atom is not forced.

**Example 4** Let  $P = \{b \leftarrow a\}$  with  $I = \{a\}$ . Here  $I \Vdash_P b$  while  $I \not\Vdash_P \{a, b\}$ .

Note that  $I \not\Vdash_P A$  should be understood as “no part of  $A$  can be motivated by  $I$ ”. This is further discussed in section 5.

As for forcing a set of negative literals, the definition is a formalization of the “negation as failure” principle where a literal is presumed false if there is no hope of proving it true. Here we use (interpretation) extension instead of proof and sets instead of single literals. The latter reflects the possibility of choice which is built into disjunctive logic programs, as is illustrated in example 3.

We note that it is impossible to force the empty set.

**Proposition 1** Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation. Then  $I \not\Vdash_P \emptyset$ .

**Proof:** The first condition in Definition 5 cannot be satisfied. □

The next proposition shows that, as forcing reveals those atoms that need to be added to an interpretation in order to make it deductively closed and the ones that are responsible for making it already partially deductively closed, deductively closed interpretations can only force those sets that already contain at least one element from the interpretation.

**Proposition 2** Let  $P$  be a disjunctive logic program and let  $I$  be a deductively closed interpretation. Then,

$$\forall S \subseteq \mathcal{B}_P \text{ such that } I \Vdash_P S \cdot S \cap I \neq \emptyset$$

**Proof:** We proceed by contradiction. Suppose that there exists a set  $S \subseteq \mathcal{B}_P$  such that  $I \Vdash_P S$  and  $S \cap I = \emptyset$ . According to Definition 5, there exists a rule  $r \in P$  such that:

1.  $H_r \cap S \neq \emptyset$ , and
2.  $B_r \subseteq I$ , and
3.  $B_r \cap S = \emptyset$ , and
4.  $(H_r \setminus S) \cap I = \emptyset$

Condition (4) together with  $S \cap I = \emptyset$  yields  $H_r \cap I = \emptyset$ . This is in contradiction with  $I$  being deductively closed as  $B_r \subseteq I$ .

Thus we can conclude that for all sets  $S$  forced by  $I$  holds that  $S \cap I \neq \emptyset$ . □

**Definition 6** Let  $P$  be a disjunctive logic program and let  $M$  be a positive interpretation. We use  $M^\neg$  to denote the union of all sets that are negatively forced by  $M$ , i.e.

$$M^\neg = \cup\{X \mid M \Vdash_P \neg X\}$$

The **closure** of  $M$ , denoted  $M^*$ , is defined by

$$M^* = M \cup \neg M^\neg$$

Note that  $M^*$  does not need to be consistent, as can be seen from the following example.

**Example 5** Let  $P$  contain

$$\begin{array}{l} a \vee b \leftarrow \\ b \leftarrow a \end{array}$$

and let  $M = \{a, b\}$ . It is easy to verify that  $M^\neg = \{a\}$  and thus  $M^*$  is not consistent.

Also,  $M^\neg$  may not itself be negatively forced.

**Example 6** Let  $P$  consist of the single rule  $a \vee b \leftarrow$  and consider  $I = \{a, b\}$ . Then  $M^\neg = \{a, b\}$  but  $M \Vdash_P \{a, b\}$ .

## 4 Unfounded Sets vs. Forcing

The intuition of forcing negative literals strongly resembles the intuition underlying so-called unfounded sets as defined in [GRS88] for disjunctive-free logic programs and in [LRS97, ELS97] for disjunctive logic programs. In this section we will investigate and formalize this connection.

We will use a slightly more general definition than the one defined in [LRS97].

**Definition 7** Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation for  $P$ . A set  $X \subseteq \mathcal{B}_P$  is called an **unfounded set** of  $P$  w.r.t.  $I$ , denoted as  $X \in \mathcal{U}_P(I)$  iff for each  $p \in X$  and for each  $r \in P$  such that  $p \in H_r$  one of the following conditions holds:

1.  $B_r \cup I$  is inconsistent, or
2.  $B_r \cap X \neq \emptyset$ , or
3.  $(H_r \setminus X) \cap (I \cup B_r) \neq \emptyset$

For any interpretation  $I$  we define<sup>7</sup>  $\mathcal{UUS}_P(I) = \cup_{X \in \mathcal{U}_P(I)} X$ .  $I$  is called **unfounded-free** iff  $I \cap \mathcal{UUS}_P(I) = \emptyset$ . The set of all unfounded-free interpretations is denoted  $\mathcal{I}_P^u$ .

The differences between Definition 7 and the definition used in [LRS97] are:

- Instead of requiring  $B_r \cap \neg I \neq \emptyset$  ([LRS97]), we demand that  $B_r \cup I$  be inconsistent. This subsumes  $B_r \cap \neg I \neq \emptyset$  and also covers rules like  $a \leftarrow b, \neg b$  which could never be used to introduce  $a$ , and should therefore not prevent  $a$  from belonging to an unfounded set.
- The original condition  $(H_r \setminus X) \cap I \neq \emptyset$  (“ $r$  has already been used”) is replaced by the more general condition  $(H_r \setminus X) \cap (I \cup B_r) \neq \emptyset$  which has a similar intuition (observe that  $B_r \subseteq I$  will be needed to apply  $r$ ). This change acknowledges that a rule such as  $a \vee c \leftarrow b, c$  is useless since, if it were applicable, it could not be used to introduce new atoms. In particular, it cannot be used to prove  $a$  and therefore should not prevent  $a$  from belonging to an unfounded set. It should be noted that our condition is less general than the corresponding one in [ELS97]. Essentially, [ELS97] allows unfounded atoms to be deduced from unknown antecedents, which is not the case, in general, in our notion.

The following example illustrates the difference between the notion from [LRS97] and Definition 7:

**Example 7** Consider the program  $P$ :

$$\begin{array}{l} c \leftarrow a, \neg a \\ a \leftarrow \neg a \end{array}$$

According to [LRS97],  $I = \emptyset$  has no nonempty unfounded sets, i.e.  $\mathcal{U}_P(I) = \{\emptyset\}$  while, following Definition 7, we obtain that  $\{c\} \in \mathcal{U}_P(I)$ .

This example clearly indicates that our notion of unfounded sets, which we will show to be equivalent to “negatively forced” set, see Corollary 5, is more natural than the original one. Indeed, no extension of  $I$  can ever force  $c$ , so there is no way to “ever” conclude  $c$  and thus, according to the intuition of “minimal undefinedness” [YY95],  $c$  should be in  $\mathcal{U}_P(I)$ .

The relationship between unfounded sets and negatively forced sets is clarified by the following propositions.

**Proposition 3** Let  $P$  be a disjunctive logic program and let  $X \in \mathcal{U}_P(I)$ , for some interpretation  $I$ . Then  $I \not\ll_P X$ .

<sup>7</sup>Note that the union of all unfounded sets  $\mathcal{UUS}_P(I)$  may not be unfounded, i.e. it may be that  $\mathcal{UUS}_P(I) \notin \mathcal{U}_P(I)$ . We have opted for the notation of  $\mathcal{UUS}_P(I)$  instead of the notation  $\mathcal{GUS}$  which is used in normal logic programs, since we cannot guarantee that the union of all unfounded sets is also an unfounded set, as is the case for normal logic programs.

**Proof:** This is a straightforward consequence of Definition 5.  $\square$

**Proposition 4** *Let  $P$  be a disjunctive logic program,  $I$  an interpretation of  $P$ , and let  $A \subseteq \mathcal{B}_P$  be such that  $I \not\vdash_P A$ . Then either*

- $A \in \mathcal{U}_P(I)$ , or
- $\exists J \in \mathcal{I}_P \cdot J \supseteq I \wedge J \Vdash_P A$

**Proof:** See appendix on page 36.  $\square$

Thus if  $I \not\vdash_P A$ , i.e. no part of  $A$  is motivated by  $I$ , then either  $A$  is unfounded or  $A$  can be forced by some extension of  $I$ . This allows us to strengthen Proposition 3.

**Corollary 5** *Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation. Then  $X \in \mathcal{U}_P(I)$  iff  $I \Vdash_P \neg X$ .*

**Proof:** Straightforward from Corollary 3 and Proposition 4.  $\square$

Thus unfounded sets are exactly those sets that cannot be added to any extension of  $I$ . Note that the proof of the above corollary, which formalizes this well-known intuition, relies on the modified Definition 7 of unfounded sets.

**Corollary 6** *Let  $P$  be a disjunctive logic program and let  $M$  be a positive interpretation. Then  $M^*$  is consistent iff  $M$  is unfounded-free.*

**Proof:** Immediate from corollary 5.  $\square$

The following proposition will be needed later on.

**Proposition 7** *Let  $P$  be a disjunctive logic program and let  $I$  and  $J$  be interpretations such that  $I \subseteq J$ . Then  $\mathcal{U}_P(I) \subseteq \mathcal{U}_P(J)$ .*

**Proof:** Obvious.  $\square$

**Proposition 8** *Let  $I$  be an unfounded-free interpretation of a disjunctive logic program  $P$  (i.e.  $I \in \mathcal{I}_P^u$ ). Then  $\mathcal{UUS}_P(I)$  is unfounded,*

**Proof:** We show that  $\mathcal{U}_P(I)$  is closed under union. Let  $X_1$  and  $X_2$  be unfounded.  $X_1 \cup X_2$  is unfounded iff for each rule  $r \in P$  such that  $(X_1 \cup X_2) \cap H_r \neq \emptyset$ , we have one of

$$\begin{aligned}
 B_r \cup I \text{ is inconsistent} & \quad \vee & (1) \\
 B_r \cap (X_1 \cup X_2) \neq \emptyset & \quad \vee & (2) \\
 (H_r \setminus (X_1 \cup X_2)) \cap I \neq \emptyset & \quad \vee & (3) \\
 (H_r \setminus (X_1 \cup X_2)) \cap B_r \neq \emptyset & \quad \vee & (4)
 \end{aligned}$$

If any condition corresponding to (1) or (2) holds for  $X_1$  or  $X_2$ , then (1) or (2) also hold. (3) is equivalent to  $H_r \cap I \neq \emptyset$  because  $I$  is unfounded-free. Consequently, if any of the conditions corresponding to (3) hold for  $X_1$  or  $X_2$ , (3) holds. Finally, if neither of the above, (4) reduces to  $H_r \cap B_r \neq \emptyset$  (because both  $B_r \cap X_1 = \emptyset$  and  $B_r \cap X_2 = \emptyset$ ), which must hold for both  $X_1$  and  $X_2$ .  $\square$

**Corollary 9** *Let  $P$  be a disjunctive logic program and let  $M$  be a positive interpretation. If  $M^*$  is consistent then  $M \Vdash_P \neg M^\neg$ .*

**Proof:** Immediate from Proposition 8 and Corollary 5.  $\square$

The following proposition shows that we may as well restrict to sets of atoms when considering unfounded-free interpretations.

**Proposition 10** *Let  $P$  be a disjunctive logic program and let  $I$  be an unfounded-free interpretation. Then  $\mathcal{UUS}_P(I) = \mathcal{UUS}_P(I \cup \neg\mathcal{UUS}_P(I))$ .*

**Proof:** See appendix on page 37.  $\square$

It is straightforward to show that the same property also holds with the definition of unfounded set from [LRS97].

Since negatively forced sets and unfounded sets coincide, we can rephrase the previous proposition in terms of negatively forcing.

**Corollary 11** *Let  $M$  be a positive interpretation such that  $M^*$  is consistent. Then  $M^{*\neg} = M^\neg$ .*

**Proof:**  $M^*$  is consistent implies, by Corollary 9, that  $M$  is unfounded-free. This yields with Proposition 10 that  $\mathcal{UUS}_P(M) = \mathcal{UUS}_P(M \cup \neg\mathcal{UUS}_P(M))$ . Since  $M$  is unfounded-free, by Proposition 8,  $\mathcal{UUS}_P(M) \in \mathcal{U}_P(I)$ . By Corollary 5, this implies  $M^\neg = M^{*\neg}$ .  $\square$

## 5 Assumption Sets

Intuitively, if a set is not forced by an interpretation, then no atom from this set can be motivated from the interpretation and thus, accepting it would mean making an *assumption*. Hence the following definition.

**Definition 8** *Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation. A set  $A \subseteq \mathcal{B}_P$  is called an **assumption set**, w.r.t.  $I$ , denoted  $A \in \mathcal{A}_P(I)$ , iff  $I \not\ll_P A$ .*

*The union of all assumption sets w.r.t.  $I$  is denoted<sup>8</sup>  $\mathcal{GAS}_P(I) = \cup_{A \in \mathcal{A}_P(I)} A$ .  $I$  is called **assumption-free** if  $I \cap \mathcal{GAS}_P(I) = \emptyset$ .*

<sup>8</sup>Note that  $\mathcal{GAS}_P(I)$  need not be an assumption set.

**Example 8** Consider the following disjunctive logic program  $P$ :

$$\begin{array}{l} a \vee b \leftarrow e, \neg d \\ \quad \quad \quad d \leftarrow \neg f \\ e \vee f \leftarrow b \end{array}$$

and assume the following interpretation  $I$ :

$$I = \{\neg a, \neg f, \neg e, \neg d\}$$

This program has the following assumption sets:

$$\begin{aligned} \mathcal{A}_P(I) = & \{\{a\}, \{b\}, \{e\}, \{f\}, \{a, b\}, \{a, e\}, \{a, f\}, \{b, e\}, \\ & \{b, f\}, \{e, f\}, \{a, b, e\}, \{a, b, f\}, \{a, e, f\}, \{b, e, f\}, \{a, b, e, f\}\} \end{aligned}$$

Then,

$$\mathcal{GAS}_P(I) = \{a, b, e, f\}$$

This interpretation is assumption-free ( $I \cap \mathcal{GAS}_P(I) = \emptyset$ ) and  $\mathcal{GAS}_P(I)$  is an assumption set w.r.t.  $I$  ( $\mathcal{GAS}_P(I) \in \mathcal{A}_P(I)$ ).

**Example 9** Reconsider the program  $P$  of example 8 and assume the following interpretation  $I$ :

$$I = \{e, f, b, \neg a, d\}$$

This program has the following assumption sets:

$$\begin{aligned} \mathcal{A}_P(I) = & \{\{a\}, \{b\}, \{e\}, \{f\}, \{d\}, \{a, b\}, \{a, e\}, \{a, f\}, \{a, d\}, \\ & \{b, e\}, \{b, f\}, \{b, d\}, \{e, d\}, \{f, d\}, \{a, b, e\}, \{a, b, f\}, \{a, b, d\}, \\ & \{b, e, f\}, \{a, e, d\}, \{a, f, d\}, \{b, e, d\}, \{b, f, d\}, \{a, b, e, d\}, \{a, b, f, d\}, \\ & \{a, b, e, f\}, \{b, d, e, f\}, \{a, b, d, e, f\}\} \end{aligned}$$

Notice that there is no assumption set that contains both  $e$  and  $f$  without also assuming the element  $b$ . This is a result of the third condition in the definition of forcing (Definition 5) that states that  $(H_r \setminus X) \cap I = \emptyset$ .

Then  $\mathcal{GAS}_P(I)$  equals:

$$\mathcal{GAS}_P(I) = \{a, b, e, f, d\}$$

This interpretation is not assumption-free since  $\mathcal{GAS}_P(I) \cap I = \{e, f, b, d\} \neq \emptyset$ . Note that this does not prevent  $\mathcal{GAS}_P(I)$  to be an assumption set for  $I$ , as  $\mathcal{GAS}_P(I) \in \mathcal{A}_P(I)$ .

The following is an immediate consequence of Proposition 3.

**Corollary 12** Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation. Then  $\mathcal{U}_P(I) \subseteq \mathcal{A}_P(I)$  and  $\mathcal{UUS}_P(I) \subseteq \mathcal{GAS}_P(I)$ . Consequently,  $I$  is unfounded-free if  $I$  is assumption-free.

Note that the inclusion may be strict as in the program  $p \leftarrow \neg p$  with  $I = \emptyset$ , where  $\mathcal{U}_P(I) = \{\emptyset\}$  while  $\mathcal{A}_P(I) = \{\emptyset, \{p\}\}$ .

**Proposition 13** *Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation. If  $I$  is assumption-free then  $\mathcal{A}_P(I)$  is closed under union.*

**Proof:** Let  $A_i \in \mathcal{A}_P(I)$ , for  $i = 1, 2$ . To show that  $A = A_1 \cup A_2 \in \mathcal{A}_P(I)$ , we verify that, for each  $r \in P$ , one of the following holds:

$$\begin{aligned} H_r \cap A &= \emptyset & \vee \\ B_r &\not\subseteq I & \vee \\ B_r \cap A &\neq \emptyset & \vee \\ (H_r \setminus A) \cap I &\neq \emptyset \end{aligned}$$

This is not hard to verify, given that both  $A_1$  and  $A_2$  satisfy a similar condition, and using the fact that  $I \cap (A_1 \cup A_2) = \emptyset$ . (E.g. the last condition reduces to  $H_r \cap I = \emptyset$  since both  $A_1$  and  $A_2$  are assumption-free.)  $\square$

**Corollary 14** *Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation. If  $I$  is assumption-free then  $\mathcal{GAS}_P(I)$  is the unique maximal (according to the subset partial order) assumption set.*

**Proof:** Immediate from Proposition 13.  $\square$

We show that assumption-free interpretations are self-motivating in the sense of the following definition.

**Definition 9** *An interpretation  $I$  for a disjunctive logic program  $P$  is called **self-enforcing** iff  $I \Vdash_P J$  for all  $J \subseteq I^+$ ,  $J \neq \emptyset$ .*

**Proposition 15** *Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation of  $P$ .  $I$  is self-enforcing iff  $I$  is assumption-free.*

**Proof:** Suppose  $I$  is assumption-free. If  $I$  were not self-enforcing, there would be some  $J \subseteq I$  such that  $I \not\Vdash_P J$ , but then, since  $I \cap J \neq \emptyset$ ,  $I$  would not be assumption-free, a contradiction.

Conversely, suppose that  $I$  is self-enforcing. If  $I$  were not assumption-free, there would be some  $A \subseteq \mathcal{B}_P$  such that  $I \not\Vdash_P A$  and  $B = A \cap I \neq \emptyset$ . Since  $I$  is self-enforcing,  $I \Vdash_P B$  and thus there exists some rule  $r \in P$  such that

$$H_r \cap B \neq \emptyset \quad \wedge \quad (5)$$

$$B_r \subseteq I \quad \wedge \quad (6)$$

$$B_r \cap B = \emptyset \quad \wedge \quad (7)$$

$$(H_r \setminus B) \cap I = \emptyset \quad (8)$$

On the other hand, since  $I \not\vdash_P A$ , this same rule  $r$  must also satisfy

$$B_r \not\subseteq I \quad \vee \quad (9)$$

$$B_r \cap A \neq \emptyset \quad \vee \quad (10)$$

$$(H_r \setminus A) \cap I \neq \emptyset \quad (11)$$

Clearly, (6) is inconsistent with (9) and (8) contradicts (11). Thus (10) must hold. But (6) implies that  $B_r \cap A \subseteq B_r \cap B$ , and thus (7) does contradict (10), characterizing  $r$  out of existence.  $\square$

Note that the analogue of Proposition 10 does not hold for  $\mathcal{GAS}$ , i.e.  $\mathcal{GAS}_P(M)$  is not necessarily equal to  $\mathcal{GAS}_P(M^*)$ , as is shown in the next example.

**Example 10** Consider  $P$  with a single rule  $a \vee b \leftarrow \neg c$  and an interpretation  $M = \emptyset$ . Then  $\mathcal{GAS}_P(M) = \{a, b, c\}$  while  $\mathcal{GAS}_P(M^*) = \mathcal{GAS}_P(\{\neg c\}) = \emptyset$ .

Also  $\mathcal{GAS}_P(I)$  is not necessarily equal to  $\mathcal{GAS}_P(M \cup \neg \mathcal{GAS}_P(M))$ , as illustrated by the following example.

**Example 11** Reconsider the one single rule program  $P$  of the previous example and the interpretation  $M = \emptyset$ . Then  $\mathcal{GAS}_P(M) = \{a, b, c\}$  while  $\mathcal{GAS}_P(M \cup \neg \mathcal{GAS}_P(M)) = \mathcal{GAS}_P(\{\neg a, \neg b, \neg c\}) = \{c\}$

## 6 Forced Models

We are now in a position to define a semantics based on forcing. Intuitively, a model  $M \subseteq \mathcal{B}_P$  should both be deductively closed, i.e. every applicable rule should be applied, and "justified", i.e. each atom form  $M$  should be motivated by a legitimate rule application. The latter condition can be rephrased as " $M^*$  is self-enforcing" (Definition 9).

Moreover, while such a definition turns out to be sound and valid (Corollary 19), it can still be simplified by removing the explicit dependence on "deductive closure".

Indeed, "deductive closure" can be formalized, albeit inaccurately, as "any atom that is not an assumption is in  $M$ ", i.e.  $\overline{\mathcal{GAS}_P(M^*)} \subseteq M$ . On the other hand, "self-enforcing" is equivalent (Proposition 15) with assumption-freeness, i.e.  $M \subseteq \overline{\mathcal{GAS}_P(M^*)}$ . Hence the following definition is based solely on the notion of forcing.

**Definition 10** Let  $M$  be a positive interpretation of a disjunctive logic program  $P$ .  $M$  is called a **forced model** of  $P$  iff  $\overline{M} = \mathcal{GAS}_P(M^*)$

Note that we base our semantics on sets of atoms. Negative literals occur in the resulting interpretation only because they need to be negated (they are negatively forced). A similar approach to negation is advocated in e.g. [GL88] and [LVZ91] for disjunctive-free logic programs. Likewise, we do not need an explicitly three-valued approach as "undefined" atoms correspond trivially to  $M \setminus M^\neg$  for any forced model  $M$ .

Let us consider some examples:

**Example 12** Consider the following program:

$$\begin{array}{l} a \vee b \leftarrow \neg d, e \\ \quad \quad \quad d \leftarrow d, f \\ e \vee f \leftarrow \end{array}$$

This program has three forced models:

$$F_1 = \{a, e\}, F_2 = \{b, e\} \text{ and } F_3 = \{f\}$$

This is easy to verify since  $\mathcal{GAS}_P(F_1^*) = \mathcal{GAS}_P(\{a, e, \neg b, \neg f, \neg d\}) = \{b, d, f\}$ ,  
 $\mathcal{GAS}_P(F_2^*) = \mathcal{GAS}_P(\{b, e, \neg f, \neg a, \neg d\}) = \{f, d, a\}$  and for  $F_3$  holds  $\mathcal{GAS}_P(F_3^*) = \mathcal{GAS}_P(\{f, \neg e, \neg d, \neg a, \neg b\}) = \{e, d, a, b\}$ .

**Example 13** Consider the following disjunctive logic program:

$$\begin{array}{l} p \leftarrow \neg q \\ \quad \quad \quad q \leftarrow \neg p \\ a \vee b \leftarrow \end{array}$$

This program has six forced model :  $\{a\}, \{b\}, \{a, p\}, \{a, q\}, \{b, p\}, \{b, q\}$ . Notice that the closures of these forced models coincide with the three-valued stable models.

Unfortunately, the forced model semantics is not universal, as the following example will illustrate.

**Example 14** Consider the following disjunctive logic program:

$$\begin{array}{l} w \vee s \vee t \leftarrow \\ \quad \quad \quad w \leftarrow \neg t \\ \quad \quad \quad s \leftarrow \neg w \\ \quad \quad \quad t \leftarrow \neg s \end{array}$$

This program has no forced models.

Although the forced model semantics may not be universal, it has still some interesting properties.

**Corollary 16** Let  $M$  be a forced model of a disjunctive logic program  $P$ .

- $M$  is unfounded-free and therefore  $M^*$  is consistent and  $M \Vdash_P \neg M^*$ .
- $M^*$  is assumption-free and thus  $M^* \not\ll_P \overline{M}$ .

**Proof:** Obvious. □

The positive complement of any assumption-free and deductively closed interpretation is an assumption set for this interpretation.

**Proposition 17** *Let  $I$  be an interpretation of a disjunctive logic program  $P$  which is both assumption-free and deductively closed. Then  $\bar{I} \in \mathcal{A}_P(I)$ , i.e.  $\bar{I}$  is an assumption set.*

**Proof:** We proceed by contradiction. Suppose  $\bar{I} \notin \mathcal{A}_P(I)$ , i.e.  $I \not\models_P \bar{I}$ . Following Definition 5, there exists a rule  $r \in P$  such that, among other conditions,  $(H_r \setminus \bar{I}) \cap I = \emptyset$ . Hence, since  $\bar{I} \cap I = \emptyset$ ,  $H_r \cap I = \emptyset$ , contradicting that  $I$  is deductively closed. □

Of course the closure of any forced model is deductively closed.

**Proposition 18** *Let  $M$  be a forced model of a disjunctive logic program  $P$ . Then  $M^*$  is deductively closed.*

**Proof:** We proceed by contradiction, i.e. suppose that

$$\exists r \in P \cdot B_r \subseteq M^* \wedge H_r \cap M = \emptyset$$

We show that  $\mathcal{GAS}_P(M^*) \cap H_r = \emptyset$ . Indeed, if  $\mathcal{GAS}_P(M^*) \cap H_r$  were not empty, we would have one of

1.  $B_r \not\subseteq M^*$ , which is false since  $B_r \subseteq M^*$ ; or
2.  $(H_r \setminus \mathcal{GAS}_P(M^*)) \cap M \neq \emptyset$ , which is false since  $H_r \cap M = \emptyset$ ; or
3.  $B_r \cap \mathcal{GAS}_P(M^*) \neq \emptyset$ , which is false since  $B_r \subseteq M^*$  and  $M^*$  is assumption-free.

Hence  $\mathcal{GAS}_P(M^*) \cap H_r = \emptyset$ , and, since  $M$  is forced,  $H_r \cap M \neq \emptyset$ , a contradiction. □

Note that it is not enough that  $\bar{M} \subseteq \mathcal{GAS}_P(M^*)$  for  $M$  to be deductively closed, as is illustrated below.

**Example 15** *Consider  $P$  containing*

$$\begin{array}{l} c \leftarrow a \\ a \leftarrow a \end{array}$$

*and  $M = \{a\}$ . Then  $\{a, c\} \in \mathcal{A}_P(M)$  while  $M^*$  is not deductively closed and even not consistent.*

**Corollary 19** *Let  $M$  be a positive interpretation of a disjunctive logic program  $P$ .  $M$  is a forced model iff  $M^*$  is self-enforcing and deductively closed.*

**Proof:** If  $M$  is forced then  $M^*$  is assumption-free and thus Proposition 15 implies that  $M^*$  is self-enforcing. Also, Proposition 18 implies that  $M^*$  is deductively closed.

Conversely, if  $M^*$  is self-enforcing then, by Proposition 15, it is assumption-free. Combined with Proposition 17, this implies that  $\overline{M} = \mathcal{GAS}_P(M^*)$ , i.e.  $M$  is forced.  $\square$

From the above corollary we know that closures of forced models are deductively closed. This can be strengthened further as in the following proposition.

**Proposition 20** *Let  $M$  be a positive interpretation. Then  $M$  is a forced model iff  $M^*$  is a minimal deductively closed interpretation containing  $\mathcal{UUS}_P(M)$ .*

**Proof:** We will first prove the “if-part”. We proceed by contradiction. Suppose  $J$  is a deductively closed interpretation such that  $J^- = \mathcal{UUS}_P(M)$  and  $J^+ \subset M^+$ . Let  $X = M^+ \setminus J^+$ . We will demonstrate that  $M^* \not\llcorner_P X$  which contradicts the self-enforcing property of  $M^*$ , as  $M$  is a forced model (Corollary 19). Because  $M^{*+} = M$  we obtain  $X \subseteq M$ . As  $J$  is deductively closed we have:

$$\forall r \in P \text{ with } H_r \cap X \neq \emptyset \cdot (H_r \setminus X) \cap J^+ \neq \emptyset \text{ or } B_r \not\subseteq J$$

The former yields  $(H_r \setminus X) \cap M \neq \emptyset$ , as  $J^+ \subset M$ , while the latter implies either  $B_r \not\subseteq M^*$  or  $B_r \cap X$ , as  $J = M^{*+} \cup X \cup \neg M^{*-}$ . With this we can conclude, following Definition 5, that  $M^* \not\llcorner_P$ .

Thus,  $M^*$  has to be a minimal deductively closed interpretation with  $M^{*-} = \mathcal{UUS}_P(M)$ . To prove the “only-if-part”, we also proceed by contradiction by assuming that  $M^*$  is not forced. As  $M^*$  is already deductively closed, we have, according to Corollary 19, that  $M^*$  cannot be self-enforcing. Following Definition 9, this implies that  $\exists X \subseteq M$  such that  $\forall r \in P$  with  $H_r \cap C \neq \emptyset$  the following conditions hold:

- $B_r \not\subseteq M^*$ , and
- $B_r \cap X \neq \emptyset$ , and
- $(H_r \setminus X) \cap M \neq \emptyset$ .

Construct now the interpretation  $J$  such that  $J^+ = M \setminus X$  and  $J^- = \mathcal{UUS}_P(M) = M^{*-}$ . Now it is easy to see that  $J$  is deductively closed. However, this is in contradiction with  $M^*$  being minimal, as  $J^+ \subset M$  and  $J^- = M^{*-}$ . Thus,  $M$  has to be a forced model.  $\square$

**Corollary 21 (Equivalence summary)** *Let  $P$  be a disjunctive logic program and let  $M$  be a positive interpretation for it. Then, the following statement are equivalent:*

1.  $M$  is a forced model.

2.  $M^*$  is self-enforcing and deductively closed.
3.  $M^*$  is a minimal deductively closed interpretation containing  $\mathcal{UUS}_P(M)$ .

**Proof:** (1)  $\Leftrightarrow$  (2): Corollary 19.

(1)  $\Leftrightarrow$  (3): Proposition 20. □

As expected, forced models are not influenced by unnatural rules.

**Lemma 1** *Let  $P$  be a disjunctive logic program and let  $I, I_c, A, A_c$  be subsets of  $\mathcal{B}_P \cup \neg\mathcal{B}_P$  such that  $I_c = I \cap (\mathcal{B}_{P_c} \cup \neg\mathcal{B}_{P_c})$ <sup>9</sup> and  $A_c = A \cap (\mathcal{B}_{P_c} \cup \neg\mathcal{B}_{P_c})$ . Then*

$$I \Vdash_P A \quad \text{iff} \quad I_c \Vdash_{P_c} A_c \quad (12)$$

$$I \Vdash_P \neg A \quad \text{iff} \quad I_c \Vdash_{P_c} \neg A_c \quad (13)$$

**Proof:** See appendix on page 39 □

**Proposition 22** *Let  $P$  be a disjunctive logic program.  $M$  is a forced model of  $P$  iff  $M$  is a forced model of  $P_c$ .*

**Proof:** Suppose that  $M$  is a forced model of  $P$ . It is easy to show that  $M^* \not\llcorner_P a$  for any  $a \in \mathcal{B}_P \setminus \mathcal{B}_{P_c}$  and thus we can conclude from Proposition 19 that  $M \subseteq \mathcal{B}_{P_c}$ .

We use  $M_c^*$  to denote the closure of  $M$  w.r.t.  $P_c$ . Using Lemma 1, in particular (13), it is straightforward to show that  $M_c^* = M^* \cap (\mathcal{B}_{P_c} \cup \neg\mathcal{B}_{P_c})$ . From Lemma 1(12), we also obtain that  $M_c^*$  is self-enforcing. Obviously,  $M$  is also deductively closed w.r.t.  $P_c$  and thus, by Proposition 19,  $M$  is a forced model of  $P_c$ .

The reverse can be shown in a similar manner. □

Next we provide a simple and natural fixpoint characterization for forced models, based on singleton forcing.

**Definition 11** *Let  $P$  be a disjunctive logic program. We define the  $\mathcal{D}_P$  operator as follows:*

$$\begin{aligned} \mathcal{D}_P & : 2^{\mathcal{B}_P} \rightarrow 2^{\mathcal{B}_P} \\ I & \mapsto \{a \in \mathcal{B}_P \mid I^* \Vdash_P a\} \end{aligned}$$

**Lemma 2** *Let  $P$  be a disjunctive logic program and let  $M$  be an assumption-free fixpoint of  $\mathcal{D}$ . Then  $M$  is a forced model.*

**Proof:** See appendix on page 40. □

**Lemma 3** *Let  $P$  be a disjunctive logic program and let  $M$  be a forced model. Then  $M$  is a fixpoint of  $\mathcal{D}_P$ .*

---

<sup>9</sup>Recall that  $P_c$  is the natural version of  $P$ , obtained by deleting every rule  $r$  in  $P$  with  $H_r \cap B_r \neq \emptyset$  or  $B_r \cap \neg B_r \neq \emptyset$

**Proof:** See appendix on page 40. □

**Proposition 23**  *$M$  is a forced model iff  $M$  is an assumption-free fixpoint of  $\mathcal{D}_P$ .*

**Proof:** This follows immediately from Lemmas 2 and 3. □

The assumption-freeness condition is necessary, as can be seen from the following example.

**Example 16** *Consider  $P$*

$$\begin{array}{l} b \leftarrow a \\ a \leftarrow b \end{array}$$

$\mathcal{D}_P(\{a, b\}) = \{a, b\}$  but  $\{a, b\}$  is not assumption-free.

## 7 Relationship with Other Approaches

### Stable vs forced models

In [LRS97], the notion of (*total*) *stable model* is defined, based on a definition of unfounded-set which is slightly different from the one derived in the present paper, as illustrated in example 7 (page 10).

The program in example 7 is not natural<sup>10</sup>. It turns out that this is necessary to obtain a collection of unfounded sets that differs from the one obtained using the definition in [LRS97].

**Proposition 24** *Let  $P$  be a natural disjunctive logic program and let  $I$  be an interpretation of  $P$ . Then,  $X \in \mathcal{U}_P(I)$  iff  $X$  is unfounded according to [LRS97].*

**Proof:** Since  $P$  is natural, we have, according to Definition 3, that each rule  $r \in P$  satisfies both  $B_r \cap \neg B_r = \emptyset$  and  $H_r \cap B_r = \emptyset$ . Thus the first condition ( $B_r \cup I$  is inconsistent) from Definition 7 reduces to  $B_r \cap \neg I \neq \emptyset$ , because  $B_r$  is consistent. Similarly, the third condition,  $(H_r \setminus X) \cap (I \cup B_r) \neq \emptyset$  reduces to  $(H_r \setminus X) \cap I \neq \emptyset$  because  $H_r \cap B_r = \emptyset$ . As the second condition is identical in both definitions, this completes the proof of the proposition. □

Both notions also coincide for total interpretations.

**Proposition 25** *Let  $P$  be a disjunctive logic program and let  $I$  be a total interpretation of  $P$ . Then, both definitions of unfounded set are equivalent.*

**Proof:** We will first show that every unfounded set  $X$  according to Definition 7 is also an unfounded set according to [LRS97]. Hence for every rule  $x \in X$  and every rule  $r \in P$  such that  $x \in H_r$  at least one of the following conditions hold:

<sup>10</sup>See Definition 3 on page 5

1.  $B_r \cup I$  is inconsistent, or
2.  $B_r \cap X \neq \emptyset$ , or
3.  $(H_r \setminus X) \cap (I \cup B_r) \neq \emptyset$

The first condition can be rewritten as  $B_r \cap \neg I \neq \emptyset$  since  $I$  is consistent and if  $B_r$  is inconsistent, then also  $B_r \cap \neg I \neq \emptyset$  because  $I$  is total. Suppose the first condition is false, i.e.  $B_r \cup I$  is consistent. This implies that  $B_r \subseteq I$  because  $I$  is total. In this case, the third condition can be rewritten as  $(H_r \setminus X) \cap I \neq \emptyset$  or  $(H_r \setminus X) \cap B_r \neq \emptyset$ . The latter implies the former since  $B_r \subseteq I$ .

So in every case at least one of the conditions for unfounded sets according [LRS97] is satisfied.

The second inclusion is also obvious since Definition 7 is a generalization of the Definition in [LRS97].  $\square$

**Proposition 26** *Let  $P$  be a disjunctive logic program and let  $I$  be a total interpretation.  $I$  is a stable model iff  $I = M^*$  for some forced model  $M$ .*

**Proof:** From Theorem 4.8 in [LRS97], it follows that  $I \in \mathcal{I}_P$  is stable iff  $\neg I^- = \mathcal{UUS}_P(I)$  (where we use Proposition 25 to justify using Definition 7 instead of the one in [LRS97]).

Now let  $M$  be a “total” forced model (in the sense that  $M^*$  is total).

Then  $M^{*-} = \mathcal{UUS}_P(M) = \mathcal{UUS}_P(M^*)$ , by Definition 10 and Proposition 10. Hence  $M^*$  is a stable model.

To show the reverse, assume that  $I$  is a stable model, i.e.  $\neg I^- = \mathcal{UUS}_P(I)$ . Define  $M = I^+$ . By Corollary 12,  $\overline{M} = \mathcal{UUS}_P(I) \in \mathcal{A}_P(I)$  and thus it remains to show that  $M^*$  is assumption-free. Assume the contrary, e.g. there exists an assumption set  $X \in \mathcal{A}_P(M^*)$  such that  $M \cap X \neq \emptyset$ . Since  $M$  is unfounded-free (because  $I = M^*$  is consistent),  $X$  cannot be an unfounded set. By Definition 7, there is a rule  $r \in P$  such that  $X \cap H_r \neq \emptyset$  for which

$$\begin{aligned}
B_r \cup M^* \text{ is consistent} & \quad \wedge \\
B_r \cap X = \emptyset & \quad \wedge \\
(H_r \setminus X) \cap (M^* \cup B_r) = \emptyset & \quad (14)
\end{aligned}$$

Since  $B_r \cup M^*$  is consistent and  $M^*$  is total, it follows that  $B_r \subseteq M^*$ . Using this, (14) reduces  $(H_r \setminus X) \cap M^* = \emptyset$ . Together, these conditions prevent any of the conditions associated with Definition 8 to hold for  $r$ , yielding a contradiction.  $\square$

The following proposition shows that the difference between the unfounded sets of an arbitrary program  $P$  and those of its consistent version  $P_c$  reduces to doomed atoms.

**Proposition 27** *Let  $P$  be a disjunctive logic program such that  $\mathcal{B}_P = \mathcal{B}_{P_c}$ , i.e.  $P$  has no doomed literals. Then  $\mathcal{U}_P(I) = \mathcal{U}_{P_c}(I)$  for any interpretation<sup>11</sup>  $I$ .*

<sup>11</sup>Note that  $P$  and  $P_c$  have the same interpretations.

**Proof:** Clearly, since  $P_c$  has fewer rules than  $P$  and  $\mathcal{B}_P = \mathcal{B}_{P_c}$ ,  $\mathcal{U}_P(I) \subseteq \mathcal{U}_{P_c}(I)$ .

To show the reverse, assume that, on the contrary, there is some  $X \in \mathcal{U}_{P_c}(I) \setminus \mathcal{U}_P(I)$ . Hence, for any rule  $r \in P_c$  which is such that  $H_r \cap X \neq \emptyset$ , we have that either  $B_r \cup I$  is inconsistent, or  $B_r \cap X \neq \emptyset$ , or  $(H_r \setminus X) \cap (I \cup B_r) \neq \emptyset$ .

On the other hand, since  $X \notin \mathcal{U}_P(I)$ , we know that there exists rules  $r' \in P$  such that none of the above conditions holds. Thus, for such  $r'$  we have that

1.  $B_{r'} \cup I$  is consistent, and
2.  $B_{r'} \cap X = \emptyset$ , and
3.  $(H_{r'} \setminus X) \cap (I \cup B_{r'}) = \emptyset$

Because  $r' \in P \setminus P_c$ ,  $\neg B_{r'} \cup H_{r'}$  must be inconsistent. It follows that either  $B_{r'}$  is inconsistent or  $B_{r'} \cap H_{r'} \neq \emptyset$ .

If  $B_{r'}$  is inconsistent then so is  $I \cup B_{r'}$ , contradicting (1) above.

If, on the other hand,  $B_{r'} \cap H_{r'} \neq \emptyset$ , then  $r'$  is of the form  $H_{r'} \cup U_{r'} \leftarrow B_{r'} \cup U_{r'}$  for some  $U_{r'} \neq \emptyset$ .

Then (3) can be rewritten as  $((H_{r'} \cup U_{r'}) \setminus X) \cap (I \cup B_{r'} \cup U_{r'}) = \emptyset$ . It is not difficult to see that the above equality cannot hold if  $U_{r'} \setminus X \neq \emptyset$ . On the other hand, if  $U_{r'} \setminus X = \emptyset$ , i.e.  $U_{r'} \subseteq X$ , then  $B_{r'} \cap X \supseteq U_{r'} \neq \emptyset$ , contradicting (2) above.  $\square$

**Proposition 28** *Let  $P$  be a disjunctive logic program with an interpretation  $I$  and let  $a$  be a doomed atom of  $P$ . Then  $\{a\} \in \mathcal{U}_P(I)$ .*

**Proof:** If  $a$  does not appear in the head of any rule, then Definition 7 holds vacuously. Let  $r$  be such that  $a \in H_r$ . Since  $r$  is inconsistent, we know that either  $B_r$  is inconsistent or  $B_r \cap H_r \neq \emptyset$ .

If  $B_r$  is inconsistent, so is  $I \cup B_r$  (condition 1 in Definition 7). If  $B_r \cap H_r \neq \emptyset$ , we consider two subcases:

- If  $a \in B_r$ ,  $B_r \cap \{a\} \neq \emptyset$ , i.e. condition 2 in Definition 7.
- If  $a \notin B_r$ , it is easy to see that  $(H_r \setminus \{a\}) \cap (I \cup B_r) \neq \emptyset$ , which is condition 3 of Definition 7.

$\square$

## Assumption sets in disjunctive-free logic programs

In [LVZ91], the notion of assumption set was defined for disjunctive-free logic programs and positive interpretations. We show that our definition properly extends the one from [LVZ91].

**Proposition 29** *Let  $P$  be a natural disjunctive-free logic program and let  $M \subseteq \mathcal{B}_P$  be an unfounded-free positive interpretation. Then  $\mathcal{A}_P(M)$  contains exactly the assumption sets w.r.t  $M$  according to [LVZ91].*

**Proof:** According to [LVZ91],  $X \subseteq \mathcal{B}_P$  is an assumption set iff for each rule  $r \in P$  such that  $H_r \in X$ , one of the following holds

$$\begin{aligned} B_r \cap \neg M &\neq \emptyset \vee \\ B_r \cap X &\neq \emptyset \vee \\ B_r &\not\subseteq M \cup \neg \mathcal{UUS}_P(M) \end{aligned}$$

The last condition is correct because of Proposition 24.

On the other hand, the conditions for  $X \in \mathcal{A}_P(M)$  can be rewritten as (note that  $(H_r \setminus X) \cap M = \emptyset$  is vacuously true since  $H_r$  is a singleton):

$$\begin{aligned} B_r &\not\subseteq M \cap \mathcal{UUS}_P(M) \vee \\ B_r \cap X &\neq \emptyset \end{aligned}$$

Thus  $X \in \mathcal{A}_P(M)$  implies that  $X$  is unfounded according to [LVZ91]. To show the reverse, it suffices to consider the case where  $B_r \cap \neg M \neq \emptyset$ . Because  $M \subseteq \mathcal{B}_P$ , this implies that  $B_r \not\subseteq M \cup \mathcal{UUS}_P(M)$  since otherwise,  $\neg M \cup \neg \mathcal{UUS}_P(M) \neq \emptyset$ , contradicting our assumption that  $M$  is unfounded-free.  $\square$

The above proposition does not necessarily hold for unnatural programs.

**Example 17** *Let  $P$  have the rules*

$$\begin{aligned} c &\leftarrow a, \neg a \\ d &\leftarrow \neg c \\ a &\leftarrow \neg a \end{aligned}$$

*and consider the interpretation  $M = \emptyset$ . Clearly,  $\mathcal{UUS}_P(M) = \{c\}$  and  $\mathcal{GAS}_P(M^*) = \mathcal{GAS}_P(\neg c) = \{a, c\}$  while, according to [LVZ91], the  $\mathcal{GAS}_P(\emptyset)$  is  $\{a, d, c\}$ .*

**Corollary 30 ([LVZ91])** *Let  $P$  be a natural disjunctive-free logic program.*

- *The forced models of  $P$  are its three-valued stable models which are equivalent with its grounded models.*
- *The unique minimal forced model of  $P$  exists and is equal to the well-founded partial model [GRS88] of  $P$ .*
- *The maximal forced models of  $P$  are its stable partial models [SZ90] which are equivalent with the regular models [YY90] and preferred extensions. They all correspond with the regular extensions [YYG98]*

- The greatest lower bound of all maximal forced models of  $P$  exists and is equal to the maximal deterministic model [SZ90].

**Proof:** Immediate from Proposition 29 and theorems in [LVZ91], [YY95], [KM92], [WB93] and [YYG98].  $\square$

## Forced models and three-valued partial stable models

In [Prz91], three-values (partial) stable models are defined as follows.

**Definition 12 ([Prz91])** Define three truth values  $\mathbf{T3} = \{\mathbf{f}, \mathbf{u}, \mathbf{t}\}$  where  $\mathbf{t}$  stands for “true”,  $\mathbf{f}$  for “false” and  $\mathbf{u}$  for “unknown” with the order  $\mathbf{f} < \mathbf{u} < \mathbf{t}$ . Let  $P$  be a disjunctive logic program. Given a truth assignment  $\phi : \mathcal{B}_P \rightarrow \mathbf{T3}$ , we define the truth value of a conjunction, disjunction and negation by

$$\begin{aligned}\phi(a \wedge b) &= \min\{\phi(a), \phi(b)\} \\ \phi(a \vee b) &= \max\{\phi(a), \phi(b)\} \\ \phi(\neg a) &= \begin{cases} \mathbf{f} & \text{if } \phi(a) = \mathbf{t} \\ \mathbf{t} & \text{if } \phi(a) = \mathbf{f} \\ \mathbf{u} & \text{if } \phi(a) = \mathbf{u} \end{cases}\end{aligned}$$

A rule  $H \leftarrow B$  is satisfied by a truth assignment if  $\phi(H) \geq \phi(B)$ , where  $H$  is considered a disjunction and  $B$  a conjunction. A **three-valued model** of a disjunctive logic program  $P$  is a truth assignment that satisfies all rules in  $P$ . Truth assignments are ordered using

$$\phi_1 \leq \phi_2 \text{ iff } \forall a \in \mathcal{B}_P \cdot \phi_1(a) \leq \phi_2(a)$$

**Definition 13 ([Prz91])** Let  $P$  be a disjunctive logic program. Given two three-valued models  $M$  and  $N$ , we can order them in the following way:

$$I \preceq_t J \text{ iff } \mathcal{V}_I(a) \leq_t \mathcal{V}_J(a) \text{ for all } a \in \mathcal{B}_P$$

or equivalently

$$M \preceq N \text{ iff } M^+ \subseteq N^+ \text{ and } M^- \supseteq N^-$$

We call this the knowledge or standard ordering.

A three-valued model  $M$  is minimal iff there exists no three-valued model  $N$  such that  $N \preceq M$ .

**Definition 14 ([Prz91])** Let  $P$  be a disjunctive logic program and let  $\phi$  be a three-valued model of  $P$ . The **Gelfond-Lifschitz transformation**  $P^\phi$  of  $P$  w.r.t.  $\phi$  is obtained by replacing every negative literal  $\neg a$  in  $P$  by  $\phi(\neg a)$ .  $\phi$  is a **three-valued partial stable model** of  $P$  if  $\phi$  is a minimal model of  $P^\phi$ .

We denote a three-valued stable model by a subset  $M \subseteq \mathcal{B}_P \cup \neg\mathcal{B}_P$  containing the true atoms and the negation of the false atoms.

Before we examine the relation between three-valued stable models and forced models, we will first prove that the former are self-enforcing. According to Corollary 19 we have that this is a necessary but insufficient condition for forced models.

**Proposition 31** *Let  $P$  be a natural disjunctive logic program and let  $M$  be a three-valued stable model for  $P$ . Then,  $M$  is self-enforcing.*

**Proof:** Because  $M$  is a three-valued stable model of  $P$ , we have that  $M$  is also a minimal three-valued model of  $P$  (Proposition 3.1 in [Prz91]).

Assume now, by means of contradiction, that  $M$  is not self-enforcing. By Definition 9, this implies that there exists a  $J \subseteq M^+$  such that for every rule  $r \in P$  with  $H_r \cap J \neq \emptyset$  the following conditions hold:

1.  $B_r \not\subseteq M$ , and
2.  $B_r \cap J \neq \emptyset$ , and
3.  $(H_r \setminus J) \cap M \neq \emptyset$ .

This implies that  $N$  with  $N^+ = M^+ \setminus J$  and  $N^- = M^- \cup J$  would be a model contradicting the minimality of  $M$ . Thus  $M$  must be self-enforcing.  $\square$

Now we prove that three-valued stable models are deductively closed.

**Proposition 32** *Let  $P$  be a disjunctive logic program and let  $M$  be a three-valued stable model for  $P$ . Then,  $M$  is deductively closed.*

**Proof:** Since  $M$  is a three-valued stable model for  $P$ ,  $M$  is also a minimal three-valued model for  $P$ . By definition of a three-valued minimal model holds that for every rule  $r \in P$   $\mathcal{V}_M(H_r) >_t \mathcal{V}_M(B_r)$ . If  $\mathcal{V}_M(B_r) = true$  or equivalently  $B_r \subseteq M$ , then  $\mathcal{V}_M(H_r) = true$ . This implies that  $H_r \cap M \neq \emptyset$ . By definition, this yields that  $M$  is deductively closed.  $\square$

According to Corollary 19, we would only need to prove for a three-valued stable model  $M$  that  $M^- = \mathcal{UUS}_P(M^+)$  to obtain that  $M^+$  is a forced model, since we have already that  $M$  is self-enforcing by Proposition 31, and that  $M$  is deductively closed, by Proposition 32. The following example illustrates that, in general, this is not possible.

**Example 18** *Let us consider the following disjunctive logic program  $P$  :*

$$\begin{array}{l} p \leftarrow \neg p \\ a \vee b \leftarrow p \end{array}$$

*This program has two three-valued stable models:*

$$\{\neg b\} \text{ and } \{\neg a\}.$$

For both models the greatest unfounded set is empty. This means that the negative part of the three-valued stable model is not necessarily an unfounded set.

The above example shows that three-valued stable models are not necessarily forced models. The next example shows that the other inclusion does not hold either.

**Example 19** Consider the program  $P$  of example 18. The unique forced model for  $P$  is the empty set while this is not a three-valued partial stable model.

The reason that no forced model sanctions e.g.  $\neg a$  is that there is a consistent extension of  $I = \emptyset$ , namely  $\{p, \neg b\}$ , that forces  $a$ , so we are not allowed to conclude  $\neg a$  on the basis of  $\emptyset$ .

This example indicates that forced models are more conservative (“skeptical”) than partial-stable models which are extremely “credulous”, concluding negations from unknown antecedents. This credulous approach results in a blind application of minimality without any epistemic justification.

The above example illustrates that, in general, forced models do not correspond with three-valued stable models, but the following proposition will prove that they always correspond with some three-valued model.

**Proposition 33** Let  $P$  be a natural disjunctive logic program and let  $M$  be a forced model for  $P$ . Then,  $M^*$  is a three-valued model for  $P$ .

**Proof:** We distinguish three different cases that correspond with the three possible truth values:

1. Let  $a \in M$   
 Since  $M^*$  is self-enforcing, there exists a rule  $r \in P$  with  $a \in H_r$  such that:
  - (a)  $B_r \subseteq M^*$ , and
  - (b)  $B_r \cap \{a\} = \emptyset$ , and
  - (c)  $(H_r \setminus \{a\}) \cap M = \emptyset$

The first condition implies that  $\mathcal{V}_{M^*}(H_r) \geq_t \mathcal{V}_{M^*}(B_r) = true$

2. Let  $a \in M^{*-}$   
 Since  $M^{*-} = \mathcal{UUS}_P(M)$  and  $M$  is a natural program, for every rule  $r \in P$  with  $a \in H_r$  at least one of the following conditions must be satisfied:
  - (a)  $B_r \cap \neg M^* \neq \emptyset$ , or
  - (b)  $B_r \cap M^{*-} \neq \emptyset$ , or
  - (c)  $(H_r \setminus M^{*-}) \cap M \neq \emptyset$

The first two conditions imply that  $\mathcal{V}_{M^*}(B_r) = false$  and the last one implies that  $\mathcal{V}_{M^*}(H_r) = true$ . So in either case  $\mathcal{V}_{M^*}(H_r) \geq_t \mathcal{V}_{M^*}(B_r)$ .

3. Let  $a \in M^{*\mathbf{u}}$ . Since  $\overline{M} = \mathcal{GAS}_P(M^*)$ , this implies that for every rule  $r \in P$  with  $a \in H_r$  at least one of following conditions must be fulfilled:

- (a)  $B_r \not\subseteq M^*$ , or
- (b)  $B_r \cap \overline{M} \neq \emptyset$ , or
- (c)  $(H_r \setminus \overline{M}) \cap M \neq \emptyset$

The first two conditions yield :  $\mathcal{V}_{M^*}(B_r) \leq_t \text{unknown}$ . and the last condition implies that  $\mathcal{V}_{M^*}(H_r) = \text{true}$ . Since  $a \in H_r$ ,  $\mathcal{V}_{M^*}(H_r) \geq_t \text{unknown}$ . So in either case,  $\mathcal{V}_{M^*}(H_r) \geq_t \mathcal{V}_{M^*}(B_r)$

Thus, in every circumstance,  $\mathcal{V}_{M^*}(H_r) \geq_t \mathcal{V}_{M^*}(B_r)$ , which implies by definition, that  $M^*$  is a partial model.  $\square$

## 8 D-Unfoundedness

In [LRS97]), disjunctive logic programs are transformed into disjunctive-free ones using a operator called the *shift* operator. In this section we introduce a class of programs that preserve the stable model semantics under the shift-operator.

**Definition 15 (The shift operator ([LRS97]))** *Given a disjunctive logic program  $P$ , we denote by  $sh(P)$  the disjunction-free program obtained from  $P$  by substituting every rule of the form  $a_1 \vee \dots \vee a_n \leftarrow b_1, \dots, b_k, \neg c_1, \dots, \neg c_l$  ( $n > 0$ ) by the  $n$  rules (one for each  $1 \leq i \leq n$ )*

$$a_i \leftarrow b_1, \dots, b_k, \neg c_1, \dots, \neg c_l, \neg a_1, \dots, \neg a_{i-1}, \neg a_{i+1}, \dots, \neg a_n$$

**Example 20** *Consider the program  $P$  from example 1:*

$$\begin{array}{l} a \vee b \leftarrow \\ a \leftarrow b \\ b \leftarrow a \end{array}$$

Then  $sh(P)$  contains

$$\begin{array}{l} b \leftarrow \neg a \\ a \leftarrow \neg b \\ b \leftarrow a \\ a \leftarrow b \end{array}$$

A natural question then is how the semantics of  $sh(P)$  relate to the semantics of  $P$ . Clearly, answering this question is fundamental for understanding the precise power of disjunctions in logic programs. E.g. in [LRS97] it was shown that the well-founded model of  $sh(P)$  ([GRS91]) coincides with the least fixpoint operator  $\mathcal{W}_P^w(\emptyset)$  of the  $\mathcal{W}_P$ -operator<sup>12</sup> applied on the original program  $P$ .

<sup>12</sup>Informally, this operator takes an unfounded-free interpretation  $I$  and returns the union of the result of the immediate consequence operator  $\mathcal{T}_P(I)$  and the negation of  $\mathcal{UUS}_P(I)$ . For more information and formal definitions see [LRS97].

**Proposition 34 (Theorem 5.11 in [LRS97])** *Let  $P$  be a function-free disjunctive logic program. Then,  $\mathcal{W}_P^w(\emptyset)$  coincide with the well-founded model of  $sh(P)$ .*

For forced (and, by Proposition 26, stable) models, the situation is not so simple. To show that the stable models of  $P$  and  $sh(P)$  coincide, it would suffice to prove the following:

**Proposition 35 (Lemma 5.9 in [LRS97])** *Let  $P$  be a disjunctive logic program and let  $I$  be an unfounded-free interpretation for  $P$ . Then  $\mathcal{U}_P(I) = \mathcal{U}_{sh(P)}(I)$ .*

From propositions 35 and 25 and Theorem 4.8<sup>13</sup> in [LRS97], the desired equivalence would follow.

Unfortunately, Proposition 35 is false. E.g. in example 20, we can take  $I = \{a, b\}$  for which  $\mathcal{U}_P(I) = \emptyset$  while  $\{a, b\} \in \mathcal{U}_{sh(P)}(I)$ .

We can however show that unfounded sets are always shifted unfounded sets.

**Proposition 36** *Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation for  $P$ . Then  $\mathcal{U}_P(I) \subseteq \mathcal{U}_{sh(P)}(I)$ .*

**Proof:** We have to prove that every unfounded set  $X$  for  $P$  w.r.t.  $I$  is also an unfounded set for  $sh(P)$  w.r.t.  $I$ . Let  $a \in X$  and let  $r \in P$  such that  $a \in H_r$ . There exists exactly one  $r' \in sh(P)$  with  $a = H_{r'}$  that corresponds with this  $r$ . Since  $X$  is an unfounded set for  $P$  w.r.t.  $I$ , one of the following conditions must hold:

1.  $B_r \cup I$  is inconsistent, or
2.  $B_r \cap X \neq \emptyset$ , or
3.  $(H_r \setminus X) \cap (I \cup B_r) \neq \emptyset$

The first condition yields  $B_{r'} \cup I$  is also inconsistent, since  $B_r \subseteq B_{r'}$ . The second condition corresponds to  $B_{r'} \cap X \neq \emptyset$ , since  $B_r^+ = B_{r'}^+$ . The third condition can be rewritten as  $(H_r \setminus \{a\}) \cap (I \cup B_r) \neq \emptyset$ . In the corresponding shifted program this yields  $B_{r'} \cap \neg(I \cup B_r) \neq \emptyset$  or  $B_r \cup I$  is inconsistent, since  $B_r \subseteq B_{r'}$ . This means that in every case one of the conditions of unfounded set is fulfilled for  $X$ . So  $X$  is also an unfounded set for  $sh(P)$  w.r.t.  $I$ .  $\square$

Using results from [LRS97], we immediately obtain.

**Corollary 37** *Let  $P$  be a disjunctive logic program and let  $M$  be a stable model of  $sh(P)$ . Then  $M$  is a stable model of  $P$ .*

Moreover, if  $I$  is unfounded-free for  $sh(P)$ , then Proposition 35 does hold.

<sup>13</sup>Let  $M$  be a total interpretation for a disjunctive logic program  $P$ .  $M$  is a stable model for  $P$  iff  $M^- = \mathcal{U}US_P(M)$ .

**Proposition 38** *Let  $P$  be a disjunctive logic program and let  $I$  be an unfounded-free interpretation for  $sh(P)$ . Then  $\mathcal{U}_P(I) = \mathcal{U}_{sh(P)}(I)$ .*

**Proof:** In Proposition 36 we already proved that  $\mathcal{U}_P(I) \subseteq \mathcal{U}_{sh(P)}(I)$ , so we only need to prove the other inclusion to have equivalence. We will show that every unfounded set  $X$  for  $sh(P)$  is also an unfounded set w.r.t.  $I$  for  $P$ . Let  $a \in X$  and let  $r \in sh(P)$  such that  $a = H_r$ . With this  $r$  there corresponds exactly one rule  $r' \in P$  such that  $a \in H_{r'}$ . Because  $X$  is an unfounded set w.r.t.  $I$  for  $sh(P)$ , one of the following conditions must hold:

$$B_r \cup I \text{ is inconsistent, or} \quad (15)$$

$$B_r \cap X \neq \emptyset \quad (16)$$

In case of the first condition, equation 15 there are two possibilities:

$$B_r \cap \neg B_r \neq \emptyset \quad (17)$$

$$B_r \cap \neg I \neq \emptyset \quad (18)$$

Equation 17 yields also two alternatives:

$$B_r \cap \neg B_r \subseteq B_{r'} \quad (19)$$

$$B_r \cap \neg B_r \subseteq H_{r'} \quad (20)$$

Equation 19 because  $B_{r'} \subseteq B_r$ . With 19, we have  $B_{r'} \cup I$  is inconsistent, with 20  $H_{r'} \cap B_{r'} \neq \emptyset$ , since  $B_r^+ = B_{r'}^+$ . This gives rise to again two possibilities:

$$(H_{r'} \setminus X) \cap B_{r'} = \emptyset \quad (21)$$

$$(H_{r'} \setminus X) \cap B_{r'} \neq \emptyset \quad (22)$$

The former, equation 21, yields  $B_{r'} \cap X \neq \emptyset$  and the latter, equation 22, yields  $(H_{r'} \setminus X) \cap (I \cup B_{r'}) \neq \emptyset$ .

Equation 18,  $B_r \cap \neg I \neq \emptyset$ , has also two alternatives:

$$B_r \cap \neg I \subseteq B_{r'}, \text{ or} \quad (23)$$

$$B_r \cap \neg I \subseteq \neg(H_{r'} \setminus \{a\}) \quad (24)$$

The former, equation 23 yields  $B_{r'} \cup I$  is inconsistent. The latter, equation 24 can be rewritten as  $(H_{r'} \setminus \{a\}) \cap I \neq \emptyset$ . Because  $I$  is unfounded-free for  $sh(P)$  this means that  $I \setminus X = \emptyset$ . So  $(H_{r'} \setminus \{a\}) \cap I \neq \emptyset$  becomes  $(H_{r'} \setminus \{a\}) \cap (I \setminus X) \neq \emptyset$  or  $(H_{r'} \setminus X) \cap (I \cup B_{r'}) \neq \emptyset$ .

The second condition or equation 16, yields in the original program  $B_{r'} \cap X \neq \emptyset$ , since  $B_r^+ = B_{r'}^+$ .

So in every possible case we have that one of the conditions of unfounded sets holds for  $X$ . So  $X$  is an unfounded set w.r.t  $I$  for  $P$ .  $\square$

Note the difference between Propositions 38 and 35: since  $\mathcal{U}_P(I) \subseteq \mathcal{U}_{sh(P)}(I)$ ,  $I$  may be unfounded-free for  $P$  but not for  $sh(P)$ , as witnessed by  $\{a, b\}$  in example 20.

Thus shifted programs generate more unfounded sets, and, by corollary 3, such programs force fewer sets than “full” disjunctive programs.

We isolate the forcing power of disjunctive rules in the following definition.

**Definition 16** Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation for  $P$ . A set of atoms  $X \subseteq \mathcal{B}_P$  is said to be **d-unfounded** w.r.t.  $I$  iff  $I \Vdash_{P_V} \neg X$  where<sup>14</sup>  
 $P_V = \{r \in P \mid \#H_r > 1\}$

Thus, a set is d-unfounded w.r.t.  $I$  if it is unfounded relative to the strictly disjunctive rules of  $P$ . In example 20, we have that  $\{a, b\} \Vdash_{P_V} \{a, b\}$  which prevents  $\{a, b\} \in \mathcal{U}_P(I)$ .

It turns out that the d-unfounded sets characterize precisely the difference between a shifted program and its original version.

**Proposition 39** Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation for  $P$ . Then  $\mathcal{U}_{sh(P)}(I) \cap \mathcal{U}_{P_V}(I) = \mathcal{U}_P(I)$  where  $P_V = \{r \in P \mid \#H_r > 1\}$

**Proof:** With Proposition 3, this can be rewritten as:

- Every unfounded set  $X$  for  $sh(P)$  w.r.t.  $I$  who is d-unfounded for  $P$  w.r.t.  $I$ , is also an unfounded set for  $P$  w.r.t.  $I$ .
- Every unfounded set for  $P$  w.r.t.  $I$  is d-unfounded for  $P$  w.r.t.  $I$  and is an unfounded set for  $sh(P)$  w.r.t.  $I$ .

The last statement can easily be proven since we have already shown, in proposition 36, that every unfounded set for  $P$  w.r.t.  $I$  is also an unfounded set for  $sh(P)$ , and because d-unfoundedness is unfoundedness reduced to a part of the original program, which does not reduce the number of unfounded sets.

We continue with the first statement. So let  $X$  be an unfounded set for  $sh(P)$  and let  $X$  be also a d-unfounded set for  $P$  w.r.t.  $I$ . We will prove that  $X$  must be an unfounded set for  $P$  as well. Since  $X$  is an unfounded set for  $P$  w.r.t.  $I$ , we know that for every rule  $r \in sh(P)$ , with  $H_r \in X$  at least one of the following conditions must hold:

1.  $B_r \cup I$  is inconsistent, or
2.  $B_r \cap X \neq \emptyset$

By definition of the shift-operator, we have that there exists exactly one rule  $r' \in P$  that corresponds with  $r$ . If this  $r'$  is not a disjunctive rule, we are finished since in that case  $r = r'$ . Suppose now that  $r'$  is a disjunctive rule such that  $r' \in P_V$ . Since  $X$  is a d-unfounded set for  $P$ , or with other words a unfounded set for  $P_V$ , we get that at least one of the conditions must be fulfilled:

1.  $B_{r'} \cup I$  is inconsistent, or
2.  $B_{r'} \cap X \neq \emptyset$ , or
3.  $H_{r'} \cap (I \cup B_{r'}) \neq \emptyset$

---

<sup>14</sup>We use  $\#X$  to denote the cardinality of a set  $X$ .

Thus we can conclude that in every case at least one of the conditions for unfounded set is satisfied, So  $X$  is an unfounded set for  $P$  w.r.t.  $I$ .  $\square$

In example 20,  $\mathcal{U}_{P_V}(\{a, b\}) = \mathcal{U}_P(I) = \{\emptyset\}$  while  $\mathcal{U}_{sh(P)}(\{a, b\}) = 2^{\{a, b\}}$ .

The definition of d-unfounded sets can actually be tightened.

**Proposition 40** *Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation for  $P$ . Then  $\mathcal{U}_{sh(P)}(I) \cap \mathcal{U}_Q(I) = \mathcal{U}_P(I)$  where  $Q = \{r \in P \mid \#(H_r \cap I) > 1\}$*

**Proof:** We will start by proving that only rules from  $Q$  can cause that an unfounded set  $X$  for  $sh(P)$  is not an unfounded set for  $P$ . Suppose such an  $X$ .

With each rule  $r \in P$  there exist  $n$  rules  $r' \in sh(P)$ , with  $n$  the number of atoms in the head of  $r$ . Take a rule  $r'$  for which holds that  $H_{r'} \in X$ . Since  $X$  is an unfounded set, at least one of following the conditions for unfounded sets must hold for  $r'$ :

1.  $B_{r'} \cup I$  is inconsistent, or
2.  $B_{r'} \cap X \neq \emptyset$ .

Since  $X$  is not an unfounded set for  $P$  we have for  $r$ :

1.  $B_r \cup I$  is consistent, and
2.  $B_r \cap X = \emptyset$ , and
3.  $(H_r \setminus X) \cap (I \cup B_r) = \emptyset$

This means, with the definition of the shift-operator, that the conditions  $B_{r'} \cap X \neq \emptyset$  and  $B_{r'} \cap \neg B_{r'} \neq \emptyset$  can not be true. So the only possibility is that  $B_{r'} \cap \neg I = S \neq \emptyset$ . But since  $(H_r \setminus X) \cap I = \emptyset$  we know that  $S \subseteq X$ . For each element  $s \in S$  there exists a  $r'' \in sh(P)$  also generated from  $r$  such that  $(H_r \setminus X) \cap I = \emptyset$  and  $B_{r''} \cap \neg I = D \neq \emptyset$ . By construction with the shift-operator we know that  $D \neq S$ . Since they are neither empty we have that  $\#(H_r \cap I) > 1$  and that  $r \in Q$ .

Now we have proven that only the rules from  $Q$  can cause problems for unfounded set of  $sh(P)$ . If we use the unfounded sets that can be produced with only the rules from  $Q$  and use them as a filter for the unfounded sets of  $sh(P)$  then we receive the unfounded sets of  $P$ .  $\square$

Proposition 39 illustrates the role of disjunction (and disjunctive rules) in addition to the “generating” power which is also present in the shifted version of a program: they act as constraints. Indeed, an unfounded set for  $I$  of  $sh(P)$  must be rejected if it can be forced (by an extension of  $I$ ) using only disjunctive rules. Proposition 40 makes it quite clear that such rules do not contribute to the model in the sense that rules from  $Q$  do not prevent any element of  $I$  from being added to an unfounded set. Summarizing, disjunctive rules may be thought of as extra stimuli for the program to consider disjunctions inclusively true, where the transformed program forces them to be treated exclusive.

**Corollary 41** *Let  $P$  be a disjunctive logic program and let  $M$  be an interpretation for  $P$ . If  $\mathcal{UUS}_{sh(P)}(I) \in \mathcal{U}_{P_v}(I)$  then,  $M$  is a stable model of  $P$  iff  $M$  is a stable model of  $sh(P)$ .*

**Proof:** Suppose  $M$  is a stable model of  $P$ , then from [LRS97] we know that  $M^- = \mathcal{UUS}_P(I)$ . If we combine  $\mathcal{UUS}_{sh(P)}(I) \in \mathcal{U}_{P_v}(I)$  and Proposition 36, we can only conclude that  $\mathcal{UUS}_P(I) = \mathcal{UUS}_{sh(P)}(I)$ . This means that  $M$  is also a stable model for  $sh(P)$ . The other implication was already proven in corollary 37  $\square$

Theorem 3.7 [LRS97] gives a class of interpretations for which the  $\mathcal{UUS}_P(I)$  is also an unfounded set for the program  $P$ . In the following corollary we give another class of interpretations which possesses this property.

**Corollary 42** *Let  $P$  be a disjunctive logic program and let  $I$  be an interpretation for  $P$ . If  $\mathcal{UUS}_{sh(P)}(I) \in \mathcal{U}_{P_v}(I)$  then,  $\mathcal{UUS}_P(I) \in \mathcal{U}_P(I)$ .*

This last result should be contrasted with Theorem 3.7 from [LRS97] which requires that  $I$  is unfounded-free in order for  $\mathcal{UUS}_P(I) \in \mathcal{U}_P(I)$ .

**Example 21** *Consider the interpretation  $I = \{a\}$  for the program*

$$\begin{array}{l} a \vee b \leftarrow d \\ d \leftarrow \neg a \end{array}$$

*Clearly  $\mathcal{UUS}_{P_v}(I) = \{d, a, b\}$ , and  $I$  is not unfounded-free. On the other hand, the shifted version is*

$$\begin{array}{l} a \leftarrow \neg b, d \\ b \leftarrow \neg a, d \\ d \leftarrow \neg a \end{array}$$

*and  $\mathcal{UUS}_{sh(P)}(I) = \{d, a, b\}$ . Corollary 42 implies that  $\mathcal{UUS}_P(I) \in \mathcal{U}_P(I)$ .*

It is easy to verify that the results of this section are also valid for [LRS97]'s definition of unfounded set. This permits us to adapt the proof of Proposition 34 which in [LRS97] used the erroneous Proposition 35.

**Proposition 34(Theorem 5.11 in [LRS97])** *Let  $P$  be a function-free disjunctive logic program. Then,  $\mathcal{W}_P^w(\emptyset)$  coincide with the well-founded model of  $sh(P)$ .*

**Proof:** The well-founded model for any disjunctive-free logic program is unfounded-free. If we apply Proposition 38 to  $\mathcal{W}_{sh(P)}^w(\emptyset)^- = \mathcal{UUS}_{sh(P)}(\mathcal{W}_{sh(P)}^w(\emptyset))$  then  $\mathcal{UUS}_{sh(P)}(\mathcal{W}_{sh(P)}^w(\emptyset)) = \mathcal{UUS}_P(\mathcal{W}_{sh(P)}^w(\emptyset))$ . Because, according to [LRS97]'s Lemma 5.10,  $\mathcal{T}_{sh(P)}(I) = \mathcal{T}_P(I)$ , we have that  $\mathcal{W}_P^w(\emptyset) = \mathcal{W}_{sh(P)}^w(\emptyset)$ .  $\square$

## 9 Conclusions and Direction for Further Research

With this paper we have shown that forced models can be seen as worthy alternatives for three-valued stable models. The definitions are elegant and intuitive, based on the

single notion of forcing. Models are simply positive interpretations for which the other truth values can be computed by means of unfounded sets. Forced models can be computed, based on a straightforward fixpoint characterization. The closure of a forced model is deductively closed and self-enforcing and is a valid three-valued model. It may be seen as a drawback that the closures of forced models and three-valued stable models no longer coincide, but we actually consider this as an advantage since, in our opinion, three-valued stable models are too credulous, deriving negations from unknown facts.

In the last section we have shown that, in general, the shift-operator is unable to preserve the stable model semantics of a disjunctive logic program. We investigated a class of programs for which the shift-operator fulfilled the expectations and we found a new category of interpretations, besides the unfounded-free interpretations, that have an "unfounded"  $UUS_P$ .

As we already mentioned in the section concerning relationships with other approaches, there exists a one-to-one mapping between maximal forced models and regular extensions of disjunctive-free programs (Corollary 30). For the future we wish to elaborate on this by investigating the disjunctive case. It might also be interesting to investigate whether three-valued stable models can be recovered from a different notion of negatively forcing, without altering the basic intuition behind forcing. Because of the results reported in [ELS97], this would lead to an equivalence between "negatively forced" and [ELS97]'s unfounded set.

## Acknowledgments

The authors wish to thank the anonymous referees who have helped improving the paper and who gave some interesting ideas for further research in this area. The first author also wishes to thank the Flemish National Science Foundation (FWO) for its financial support.

## References

- [BG94] C. Baral and M. Gelfond. Logic programming and knowledge representation. *Journal of Logic Programming*, 19/20:73–148, 1994.
- [BLM89] C. Baral, J. Lobo, and J. Minker. Generalised well-founded semantics for logic programs. In M.E. Stickel, editor, *Proc. of tenth Intl. Conf. on Automated deduction*, pages 112–116, Kaiserlautern, Germany, July 1989. Springer-Verslag.
- [Cha93] E. P. F. Chan. A possible world semantics for disjunctive databases. *IEEE Transactions on Knowledge and Data Engineering*, 5(2):282–292, April 1993.

- [ELS97] Thomas Eiter, Nicola Leone, and Domenico Sacca. On the Partial Semantics for Disjunctive Deductive Databases. *Annals of Mathematics and Artificial Intelligence*, 19(1–2):59–96, 1997.
- [GL88] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proc. of fifth logic programming symposium*, pages 1070–1080. MIT PRESS, 1988.
- [GRS88] A. Van Gelder, K. Ross, and J. S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *Proceedings of the Seventh ACM Symposium on Principles of Database Systems*, pages 221–230, Austin, Texas, 1988. Association for Computing Machinery.
- [GRS91] Allen Van Gelder, Kenneth A. Ross, and John S. Schlipf. The well-founded semantics for general logic programs. *Journal of ACM*, 38(3):620–650, July 1991.
- [Kei91] H. Jerome Keisler. *Handbook of mathematical logic*, chapter Fundamentals of Model Theory, pages 89–94. North Holland, 1991.
- [KM92] A.e Kakas and P. Marcarella. Preferred extensions are partial stable models. *Journal of Logic Programming*, pages 341–348, 1992.
- [LRS97] Nicola Leone, Pasquale Rullo, and Francesco Scarello. Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation. *Journal of Information and Computation*, 135(2):69–112, June 1997.
- [LVZ91] Els Laenens, Dirk Vermeir, and Carlo Zaniolo. Pure models for logic programs: a simplification and unification of logic programming semantics. article, May 1991.
- [Prz91] Theodor C. Przymusiński. Stable semantics for disjunctive programs. In *New Generation Computing*, volume 9, pages 401–424. OHMSHA, LTD. and Springer-Verslag, 1991.
- [RM95] C. Ruiz and J. Minker. Computing stable and partial stable models of extended disjunctive logic programs. *Lecture Notes in Computer Science*, 927:205–??, 1995.
- [Rob70] A. Robinson. Forcing in model theory. In *Symposia Math*, volume 5, pages 69–82, 1970.
- [Ros89] Kenneth A. Ross. The well-founded semantics for disjuncted logic programs. In W. Kim, J.N. Nicolas, and S. Nishio, editors, *Proc. of the first Intl. Conf. on Deductive and Object Oriented Databases*, pages 337–351, Kyoto, Japan, December 1989. Elsevier Science Publishers B.V. (North-Holland).
- [Sak89] C. Sakama. Possible model semantics for disjunctive databases (preliminary report). In *Proc. First Int’l. Conf. on Deductive and Object-Oriented Databases*, Kyoto, Japan, December 4–6 1989.

- [SI94] Chiaki Sakama and Katsumi Inoue. An alternative approach to the semantics of disjunctive logic programs and deductive databases. *Journal of Automated Reasoning*, 13:145–172, 1994.
- [SZ90] D. Sacca and C. Zaniolo. Stable models and non-determinism in logic programs with negation. In *Proc. of the 9th ACM PODS*, pages 205–217, 1990.
- [WB93] C. Witteveen and G. Brewka. Skeptical reason maintenance and belief revision. *Journal of Artificial Intelligence*, 61:1–36, 1993.
- [YY90] Jia-Huai You and Li Yan Yuan. Three-valued formalization of logic programming: Is it needed? In *PODS'90*, pages 172–182. Association of Computing Machinery, 1990.
- [YY95] Jia-Huai You and Li Yan Yuan. On the equivalence of semantics for normal logic programs. *Journal of Logic Programming*, 22(3):211–222, 1995.
- [YYG98] Jia-Huai You, Li Yan Yuan, and Randy Goebel. Regular extension semantics and disjunctive eshghi-kowalski procedure. In *JICSLP'98*, 1998.

## A Proofs

**Proof of Proposition 4:** According to Definition 5,  $I \not\ll_P A$  is equivalent with

$$\forall r \in P \cdot \neg(H_r \cap A \neq \emptyset \wedge B_r \subseteq I \wedge B_r \cap A = \emptyset \wedge (H_r \setminus A) \cap I = \emptyset)$$

or, equivalently,

$$\forall r \in P \cdot H_r \cap A = \emptyset \vee B_r \not\subseteq I \vee B_r \cap A \neq \emptyset \vee (H_r \setminus A) \cap I \neq \emptyset$$

This can be rewritten as

$$\forall r \in P \cdot H_r \cap A = \emptyset \vee B_r \cap \bar{I} \neq \emptyset \vee B_r \cap A \neq \emptyset \vee (H_r \setminus A) \cap I \neq \emptyset \quad (25)$$

Using the fact that  $\bar{I} = \neg I \cup (I^\circ \cup \neg I^\circ)$ , the second part of the disjunction in (25) can be rewritten as

$$B_r \cap \neg I \neq \emptyset \vee B_r \cap (I^\circ \cup \neg I^\circ) \neq \emptyset$$

Thus (25) becomes

$$\forall r \in P \cdot \alpha_r \vee \beta_r \vee \gamma_r \vee \delta_r \vee \epsilon_r \quad (26)$$

where

$$\begin{aligned} \alpha_r &\equiv H_r \cap A = \emptyset \\ \beta_r &\equiv B_r \cap \neg I \neq \emptyset \\ \gamma_r &\equiv B_r \cap (I^\circ \cup \neg I^\circ) \neq \emptyset \\ \delta_r &\equiv B_r \cap A \neq \emptyset \\ \epsilon_r &\equiv (H_r \setminus A) \cap I^+ \neq \emptyset \end{aligned}$$

Next we define

$$R = \{r \in P \mid \neg\alpha_r \wedge \neg\beta_r \wedge \neg\delta_r \wedge \neg\epsilon_r\}$$

We consider several cases:

- $R = \emptyset$ . Then, clearly,  $A \in \mathcal{U}_P(I)$  and we are done.
- If  $R \neq \emptyset$  then we define, for each  $r \in R$ ,

$$J_r = I \cup (B_r \cap (I^\circ \cup \neg I^\circ)) \cup \neg((H_r \setminus A) \cap I^\circ)$$

Again we consider two cases:

- $J_r$  is consistent for some  $r \in R$ . But then  $J_r \Vdash_P A$ , because of the construction of  $J_r$ , so we are done.

–  $J_r$  is inconsistent for each  $r \in R$ .

Consider such an  $r$ . Since  $I$  is consistent, it follows from the definition of  $J_r$  that there are but two possibilities for  $J_r$  to be inconsistent.

The first possibility is that  $B_r$  is itself inconsistent. In this case we have that  $B_r \cup I$  is inconsistent, and this is one of the conditions for  $A$  to be an unfounded set w.r.t.  $I$ .

The other possibility for  $J_r$  to be inconsistent is that

$$U_r = (H_r \setminus A) \cap I^o \cap B_r \neq \emptyset$$

In this case,  $r$  must be of the form

$$H'_r \cup U_r \leftarrow B'_r \cup U_r$$

This yields  $(H_r \setminus A) \cap (I \cup B_r) \neq \emptyset$  which is also one of the conditions for  $A$  to be an unfounded set w.r.t.  $I$ .

So in every case we have that for each  $r \in P$ , at least one of the conditions for  $A$  to be an unfounded set is satisfied so  $A \in \mathcal{U}_P(I)$ .

This completes the proof of the proposition. □

**Proof of Proposition 10:** From proposition 7 we know that  $\mathcal{U}_P(I) \subseteq \mathcal{U}_P(I \cup \neg \mathcal{U}\mathcal{S}_P(I))$  and thus  $\mathcal{U}\mathcal{S}_P(I) \subseteq \mathcal{U}\mathcal{S}_P(I \cup \neg \mathcal{U}\mathcal{S}_P(I))$ .

To show the reverse, let  $X \in \mathcal{U}_P(I \cup \neg \mathcal{U}\mathcal{S}_P(I))$ . We will show that

$$X \cup \mathcal{U}\mathcal{S}_P(I) \in \mathcal{U}_P(I) \tag{27}$$

from which it follows that  $X \subseteq \mathcal{U}\mathcal{S}_P(I)$  and thus  $\mathcal{U}\mathcal{S}_P(I \cup \neg \mathcal{U}\mathcal{S}_P(I)) \subseteq \mathcal{U}\mathcal{S}_P(I)$ , which implies the proposition.

To show (27), let  $X \in \mathcal{U}_P(I \cup \neg \mathcal{U}\mathcal{S}_P(I))$ . If  $X \in \mathcal{U}_P(I)$ , (27) follows immediately.

Otherwise, we know that for each  $r \in P$  for which  $H_r \cap X \neq \emptyset$ , one of the following holds:

$$B_r \cap \neg B_r \neq \emptyset \vee \tag{28}$$

$$B_r \cap \neg(I \cup \neg \mathcal{U}\mathcal{S}_P(I)) \neq \emptyset \vee$$

$$B_r \cap X \neq \emptyset \vee \tag{29}$$

$$(H_r \setminus X) \cap (I \cup \neg \mathcal{U}\mathcal{S}_P(I)) \neq \emptyset \vee$$

$$(H_r \setminus X) \cap B_r \neq \emptyset \tag{30}$$

The second condition reduces to the disjunction of the following two conditions:

$$B_r \cap \neg I \neq \emptyset \vee \quad (31)$$

$$B_r \cap \mathcal{UUS}_P(I) \neq \emptyset \quad (32)$$

$(H_r \setminus X) \cap (I \cup \neg \mathcal{UUS}_P(I)) \neq \emptyset$  reduces to

$$(H_r \setminus X) \cap I \neq \emptyset \quad (33)$$

because  $H_r \cap \mathcal{UUS}_P(I) \subseteq \mathcal{B}_P$ .

We also know that  $\mathcal{UUS}_P(I)$  is unfounded, thus for each  $r \in P$ , if  $H_r \cap \mathcal{UUS}_P(I) \neq \emptyset$ , one of the following must hold:

$$B_r \cap \neg B_r \neq \emptyset \quad \vee \quad (\text{i.e. (28)})$$

$$B_r \cap \neg I \neq \emptyset \quad \vee \quad (\text{i.e. (31)})$$

$$B_r \cap \mathcal{UUS}_P(I) \neq \emptyset \quad \vee \quad (\text{i.e. (32)})$$

$$(H_r \setminus \mathcal{UUS}_P(I)) \cap I \neq \emptyset \quad \vee$$

$$(H_r \setminus \mathcal{UUS}_P(I)) \cap B_r \neq \emptyset \quad (34)$$

$(H_r \setminus \mathcal{UUS}_P(I)) \cap I \neq \emptyset$  reduces to

$$H_r \cap I \neq \emptyset \quad (35)$$

because  $I$  is unfounded-free.

To show (27), we need to show that each rule  $r \in P$  for which  $H_r \cap (X \cup \mathcal{UUS}_P(I)) \neq \emptyset$  satisfies one of

$$B_r \cap \neg B_r \neq \emptyset \vee \quad (36)$$

$$B_r \cap \neg I \neq \emptyset \vee \quad (37)$$

$$B_r \cap (X \cup \mathcal{UUS}_P(I)) \neq \emptyset \vee \quad (38)$$

$$(H_r \setminus (X \cup \mathcal{UUS}_P(I))) \cap I \neq \emptyset \vee$$

$$(H_r \setminus (X \cup \mathcal{UUS}_P(I))) \cap B_r \neq \emptyset \quad (39)$$

$(H_r \setminus (X \cup \mathcal{UUS}_P(I))) \cap I \neq \emptyset$  is equivalent with

$$(H_r \setminus X) \cap I \neq \emptyset \wedge (H_r \setminus \mathcal{UUS}_P(I)) \cap I \neq \emptyset$$

The second part of this conjunction reduces to  $H_r \cap I \neq \emptyset$  since  $I \cap \mathcal{UUS}_P(I) = \emptyset$  because  $I$  is unfounded-free. So  $(H_r \setminus (X \cup \mathcal{UUS}_P(I))) \cap I \neq \emptyset$  reduces to

$$(H_r \setminus X) \cap I \neq \emptyset \quad (40)$$

Let  $r \in P$  be a rule for which  $H_r \cap (X \cup \mathcal{UUS}_P(I)) \neq \emptyset$ .

First we note that (28) is identical to (36), so we can ignore this case.

If (30) then, if (34), we get (39). If (34) is false, i.e.  $(H_r \setminus \mathcal{UUS}_P(I)) \cap B_r = \emptyset$ , then, since  $H_r \cap B_r \neq \emptyset$  because of (30), we obtain  $\mathcal{UUS}_P(I) \cap B_r \neq \emptyset$  and thus (38). A similar reasoning holds if (34).

For the remaining possibilities, we consider three cases:

- $H_r \cap \mathcal{UUS}_P(I) \neq \emptyset$  and  $H_r \cap X = \emptyset$ . If  $B_r \cap \neg I \neq \emptyset$ , we have (37). If  $B_r \cap \mathcal{UUS}_P(I) \neq \emptyset$ , we obviously get (38). Finally, if (35), since  $X \cap H_r = \emptyset$ , we obtain (40).
- $H_r \cap \mathcal{UUS}_P(I) \neq \emptyset$  and  $H_r \cap X \neq \emptyset$ . Again, if  $B_r \cap \neg I \neq \emptyset$ , we have (37) and  $B_r \cap \mathcal{UUS}_P(I) \neq \emptyset$  implies (38). If (35) holds for  $\mathcal{UUS}_P(I)$ , a similar reasoning holds if  $X$  satisfies either (31) or (32) or (29). Finally, if (35) and (33), we note that (33) is identical to (40).
- $H_r \cap \mathcal{UUS}_P(I) = \emptyset$  and  $H_r \cap X \neq \emptyset$ : (31) is identical to (37), (32) implies (38), as does (29). Finally, (33) is identical to (40).

□

**Proof of Lemma 1:** We first prove (12). Suppose  $I \Vdash_P A$ . According to Definition 5, there is a rule  $r \in P$  such that

$$H_r \cap A \neq \emptyset \quad \wedge \quad (41)$$

$$B_r \subseteq I \quad \wedge \quad (42)$$

$$B_r \cap A = \emptyset \quad \wedge \quad (43)$$

$$(H_r \setminus A) \cap I = \emptyset \quad (44)$$

(42) implies that  $B_r$  is consistent and that

$$(H_r \setminus A) \cap I \supseteq (H_r \setminus A) \cap B_r$$

From (43) and (44) it then follows that

$$(H_r \setminus A) \cap B_r = H_r \cap B_r = \emptyset$$

and thus  $r$  must be consistent. Consequently  $r \in P_c$  and

$$(H_r \cup B_r) \subseteq \mathcal{B}_{P_c} \cup \neg \mathcal{B}_{P_c} \quad (45)$$

Using (45) and the fact that both  $I_c \subseteq I$  and  $A_c \subseteq A$ , we easily obtain

$$H_r \cap A_c \neq \emptyset \quad \wedge \quad (46)$$

$$B_r \subseteq I_c \quad \wedge \quad (47)$$

$$B_r \cap A_c = \emptyset \quad \wedge \quad (48)$$

$$(H_r \setminus A_c) \cap I_c = \emptyset \quad (49)$$

from which it follows that  $I_c \Vdash_{P_c} A_c$ .

To show the reverse, assume that  $I_c \Vdash_{P_c} A_c$ , i.e. there exists  $r \in P_c$  such that (46) through (49). Note that  $r \in P$  because  $P_c \subseteq P$ . It is easy to see that (46) implies (41) and that (42) follows from (47).

From (47) we also get that  $B_r \subseteq I_c \subseteq \mathcal{B}_{P_c} \cup \neg \mathcal{B}_{P_c}$  and thus  $B_r \cap A_c = B_r \cap A$ , which implies (43).

Finally, since  $H_r \subseteq \mathcal{B}_{P_c}$ ,  $H_r \cap I_c = H_r \cap I$  and  $H_r \setminus A_c = H_r \setminus A$  from which (44) readily follows. From (41) through (44) we can conclude, with Definition 5, that  $I \Vdash_P A$ .

It is straightforward to prove (13) from (12).  $\square$

**Proof of Lemma 2:** Since  $M$  is assumption-free, all we need to show is that  $\overline{M} \in \mathcal{A}_P(M)$ . Suppose, on the contrary, that  $M^* \Vdash_P \overline{M}$ . By definition 5, there exists some  $r \in P$  that satisfies all the following conditions.

$$\begin{aligned} H_r \cap \overline{M} &\neq \emptyset && \wedge \\ B_r &\subseteq M^* && \wedge \\ B_r \cap \overline{M} &= \emptyset && \wedge \\ (H_r \setminus \overline{M}) \cap M^* &= \emptyset \end{aligned}$$

As  $M^* \cap \overline{M} = \emptyset$ , the last condition reduces to  $H_r \cap M = \emptyset$ . But this implies that  $M^* \Vdash_P a$  for any  $a \in H_r \cap \overline{M}$  and thus that  $\mathcal{D}_P(M) \cap \overline{M} \neq \emptyset$ , contradicting the fact that  $M$  is a fixpoint of  $\mathcal{D}$ .  $\square$

**Proof of Lemma 3:** First we show that  $M \subseteq \mathcal{D}_P(M)$ . Suppose not, i.e. there is some  $a \in M$  such that  $M^* \not\Vdash_P a$ . But then  $\{a\} \in \mathcal{A}_P(M^*)$ , contradicting the fact that  $M$  is assumption-free.

Next we show that  $\mathcal{D}_P(M) \subseteq M$ . Again we proceed by contradiction: assume that  $b \in \overline{M}$  is such that  $M \Vdash_P b$ . By Definition 5, there is a rule  $r$  in  $P$  that satisfies all of the following conditions.

$$\begin{aligned} H_r \cap \{b\} &\neq \emptyset && \wedge \\ B_r &\subseteq M^* && \wedge \\ B_r \cap \{b\} &= \emptyset && \wedge \\ (H_r \setminus \{b\}) \cap M^* &= \emptyset \end{aligned}$$

The last condition reduces to  $H_r \cap M^* = \emptyset$  because  $b \notin M$ .

Now, because  $M^* \not\ll_P \overline{M}$  and  $H_r \cap \overline{M} \neq \emptyset$ ,  $r$  must also satisfy one of the following conditions.

$$\begin{aligned} B_r &\not\subseteq M^* && \vee \\ B_r \cap \overline{M} &\neq \emptyset && \vee \\ (H_r \setminus \overline{M}) \cap M &\neq \emptyset \end{aligned}$$

where the last condition reduces to  $H_r \cap M^* = \emptyset$  because  $b \notin M$ .

It is easy to verify that  $r$  cannot satisfy all the above conditions and thus  $\mathcal{D}_P(M) \subseteq M$ .  $\square$