Information and Computation, Vol. 135, N. 2, 1997, pp. 69-112.

Disjunctive Stable Models: Unfounded Sets, Fixpoint Semantics, and Computation*

Nicola Leone
Information Systems Dept.
Technical University of Vienna
A-1040 Vienna, Austria
leone@dbai.tuwien.ac.at

Pasquale Rullo
DIMET
Univ. Reggio Calabria
89100 Reggio Calabria, Italy
rullo@si.deis.unical.it

Francesco Scarcello
DEIS
Univ. della Calabria
87030 Rende, Italy
frank@si.deis.unical.it

TR 96-98 CD Lab, TU WIEN
Vienna, April 1996

Abstract

Disjunctive logic programs have become a powerful tool in knowledge representation and commonsense reasoning. This paper focuses on stable model semantics, currently the most widely acknowledged semantics for disjunctive logic programs.

After presenting a new notion of unfounded sets for disjunctive logic programs, we provide two declarative characterizations of stable models in terms of unfounded sets. One shows that the set of stable models coincides with the family of unfounded-free models (i.e., a model is stable iff it contains no unfounded atoms). The other proves that stable models can be defined equivalently by a property of their false literals, as a model is stable iff the set of its false literals coincides with its greatest unfounded set. We then generalize the well-founded $\mathcal{W}_\mathcal{P}$ operator to disjunctive logic programs, give a fixpoint semantics for disjunctive stable models and present an algorithm for computing the stable models of function-free programs. The algorithm's soundness and completeness are proved and some complexity issues are discussed.

1 Introduction

Disjunctive logic programs are logic programs where disjunction is allowed in the heads of the rules and negation may occur in the bodies of the rules. Such programs are now widely recognized as a valuable tool for knowledge representation and commonsense reasoning [3, 37, 33, 30]. Strong interest in enriching logic programming with disjunction is evident in the number of publications (see [37]) and even workshops (e.g., [33]) dedicated to this enterprise. One of

*Some of the results reported in this paper appear in [36]. This work has been supported in part by the *Christian Doppler Laboratory for Expert Systems*; *Istituto per la Sistemistica e l'Informatica, ISI-CNR*; *FWF project P11580-MAT "A Query System for Disjunctive Deductive Databases"*; *EC-US project "DEUS EX MACHINA"*; and a MURST grant (40% share) under the project "Sistemi formali e strumenti per basi di dati evolute."

the attractions of disjunctive logic programming is its ability to naturally model incomplete knowledge [3, 37].

Example 1.1 *(modified from [3, 47])* Consider the following situation: (1) we just saw Max with one broken arm but do not remember which; (2) we know that Max writes with his left hand, so he can write if his left arm is unbroken.

The question is, Can Max write or not ? Because of the uncertainty deriving from our incomplete knowledge about the state of Max's arms, we cannot answer the question definitely. Rather, two *possible* answers can be constructed: (a) "Max's left arm is broken, and he cannot write," and (b) "Max's right arm is broken, and he can write."

In the language of disjunctive logic programs, this situation is represented by rules

$$r_1: \quad la\_broken \vee ra\_broken \leftarrow \qquad\qquad r_2: \quad can\_write \leftarrow \neg la\_broken$$

The semantics of the disjunctive logic program $\mathcal{P}_{Max}$ consisting of the rules r_1 and r_2 is given by the following two models (as will be shown in Section 2):

$$M_1 = \{la\_broken, \neg ra\_broken, \neg can\_write\} \qquad M_2 = \{ra\_broken, \neg la\_broken, can\_write\}$$

M_1 and M_2 are the two possible scenarios, and they correspond exactly to our intended specification.

It is worth noting that the situation can be represented equivalently by a traditional logic program $\mathcal{P}'_{Max}$ where r_1 is replaced by the rules $la\_broken \leftarrow \neg ra\_broken$ and $ra\_broken \leftarrow \neg la\_broken$. However, the unstratified negation in $\mathcal{P}'_{Max}$ makes $\mathcal{P}'_{Max}$ less intuitive than $\mathcal{P}_{Max}$, and it is certainly clear that disjunctive logic programming provides the more natural and intuitive representation of the incomplete knowledge. □

Nonetheless, defining the semantics of a disjunctive logic program is complicated by the presence of disjunction in the rules' heads because it makes disjunctive logic programming inherently nonmonotonic, (i.e., new information can invalidate previous conclusions). Much research has been done on the semantics of disjunctive logic programs, and several alternative semantics have been proposed [10, 21, 30, 44, 52, 50, 51, 56, 58] (see [1, 15, 37] for comprehensive surveys). One widely accepted semantics is the extension to the disjunctive case of the stable model semantics of Gelfond and Lifschitz [31]. According to this semantics [30, 50], a disjunctive logic program may have several alternative models (but possibly none), each corresponding to a possible view of the reality.

Disjunctive logic programs with stable model semantics are very expressive. In [32, 18, 19] it is proved that, under stable model semantics, disjunctive (function-free) logic programs capture the complexity class Σ_2^P (i.e., they allow us to express *every* property that is decidable in nondeterministic polynomial time with an oracle in NP). As Eiter et al. [18] showed, the expressiveness of disjunctive logic programming has practical implications, as real-world situations can be represented by stable model semantics for disjunctive logic programs, while they cannot be expressed by (disjunction-free) logic programs. Stable model semantics for disjunctive logic programs also allows several nonmonotonic logic languages to be translated into disjunctive logic programs (under stable model semantics) [22, 30, 59]. This means that computation of stable models – one of the main objectives of this paper – is at the heart of the computation of several important problems in artificial intelligence.

The main results of the paper can be summarized as follows:

1. We give an original definition of unfounded sets for disjunctive logic programs as an extension of the analogous concept defined for (disjunction-free) logic programs [64]. Unfounded sets for disjunctive logic programs (as for normal logic programs) single out the atoms that are (definitely) not derivable from a given program w.r.t. a fixed interpretation, and thus, according to the closed-world assumption [54], they can be stated to be false. In a disjunctive logic program $\mathcal{P}$ the union of unfounded sets for $\mathcal{P}$ may not be an unfounded set for $\mathcal{P}$ (and the existence of a greatest unfounded set – an unfounded set that contains all other unfounded sets – is not guaranteed). However, for unfounded-free interpretations (i.e., interpretations not containing any unfounded atom), the union of unfounded sets is an unfounded set, and thus there exists the *greatest unfounded set* of $\mathcal{P}$ w.r.t. I, denoted $GUS_{\mathcal{P}}(I)$, which is the union of all unfounded sets. We show that the $GUS_{\mathcal{P}}$ operator is monotonic on its domain.

2. We discover several interesting relationships between stable models and unfounded sets, which lead to a simple yet elegant characterization of disjunctive stable models in terms of unfounded sets. We show that disjunctive stable models coincide with the unfounded-free models of $\mathcal{P}$ and that a model of $\mathcal{P}$ is stable iff the set of false atoms coincides with the greatest unfounded set. Since the elements in the greatest unfounded set are the atoms not derivable from the program, disjunctive stable models can be regarded as those models that are "compatible" with the closed-world assumption [54].

3. We define a fixpoint semantics for disjunctive stable models in terms of a suitable operator $\mathcal{W}_{\mathcal{P}}$ that extends the well-founded operator of Van Gelder et al. [64]. We show that the set of stable models of $\mathcal{P}$ coincides with the set of the (total) fixpoints of $\mathcal{W}_{\mathcal{P}}$.

4. Exploiting the above theoretical results, we design an algorithm for the computation of the stable model semantics of disjunctive deductive databases (i.e., function-free disjunctive logic programs). The key idea is that, since stable models are total interpretations, computing their entire negative portion is superfluous; rather, it is sufficient to restrict the computation to those negative literals that are necessary to derive the positive part. To this end, we introduce the notion of *possibly-true literals*. These play a crucial role in our computation. The algorithm is based on a controlled search in the space of the interpretations implemented by a backtracking technique. The stability of a generated model is tested by checking whether it is unfounded-free by means of a function that runs in polynomial time on *head-cycle-free* (*HCF*) programs [7, 8]. In the general case, our algorithm for the computation of stable models runs in polynomial space and single exponential time. We formally prove both the soundness and the completeness of the proposed method.

5. Finally, we analyze the complexity of the main computational problems related to the concepts we have presented.

It is worth noting that most of the results in this paper generalize analogous results for traditional logic programs.

The paper is organized as follows. In Section 2 we offer basic preliminaries on disjunctive logic programming. In Section 3 we define the notion of unfounded sets and describe some important properties of unfounded sets. In Section 4 we investigate the relationships between stable models and unfounded sets and provide a declarative characterization of the former in terms of the latter. A fixpoint semantics of disjunctive stable models is given in Section 5. Section 6 presents the algorithm for the computation of stable models. Related work is addressed in Section 7. Finally, in Section 8 we draw conclusions and outline ongoing research.

2 Preliminaries on Disjunctive Logic Programming

In this section, we provide an overview of the stable model semantics for disjunctive logic programs. (For further details, see [37].)

The terms of the language are inductively defined. A variable or constant is a *term*; a function symbol with terms as arguments is a term. An *atom* is $a(t_1, ..., t_n)$, where a is a *predicate* of arity n and $t_1, ..., t_n$ are terms. A *literal* is either a *positive literal* p or a *negative literal* $\neg p$, where p is an atom. We use an upper case letter, say L, to denote either a positive or a negative literal. Two literals are *complementary* if they are of the form p and $\neg p$, for some atom p. Given a literal L, $\neg.L$ denotes its complementary literal. Accordingly, given a set A of literals, $\neg.A$ denotes the set $\{\neg.L \mid L \in A\}$.

A *(disjunctive) rule* r is a clause of the form

$$a_1 \vee \cdots \vee a_n \leftarrow b_1, \cdots, b_k, \neg b_{k+1}, \cdots, \neg b_m \qquad n \geq 1,\ m \geq 0$$

where $a_1, \cdots, a_n, b_1, \cdots, b_m$ are atoms. The disjunction $a_1 \vee \cdots \vee a_n$ is the *head* of r, while the conjunction $b_1, ..., b_k, \neg b_{k+1}, ..., \neg b_m$ is the *body* of r. We denote by $H(r)$ the set $\{a_1, ..., a_n\}$ of the head atoms, and by $B(r)$ the set $\{b_1, ..., b_k, \neg b_{k+1}, ..., \neg b_m\}$ of the body literals. $B^+(r)$ and $B^-(r)$ denote, respectively, the set of positive literals and the set of negative literals occurring in $B(r)$. A *(disjunctive) program* is a (finite or countably infinite) set of rules.[1] A $\neg$-free (resp., $\vee$-free) program is called *positive* (resp., *normal* or *disjunction-free*). A term, an atom, a literal, a rule, or a program is *ground* if no variables appear in it. A finite ground program is also called a *propositional* program. A *function-free* program (also called a *disjunctive deductive database*) is a finite program where no function symbol occurs.

Let $\mathcal{P}$ be a program. The *Herbrand universe* $U_\mathcal{P}$ of $\mathcal{P}$ is the set of ground terms that use the function symbols and constants that appear in the program.[2] The *Herbrand base* $B_\mathcal{P}$ of $\mathcal{P}$ is the set of all possible ground atoms that can be constructed from the predicates appearing in the rules of $\mathcal{P}$ and the terms occurring in $U_\mathcal{P}$. If $\mathcal{P}$ is finite and contains no function symbols of positive arity, then both $U_\mathcal{P}$ and $B_\mathcal{P}$ are finite; otherwise, they are countably infinite. Given a rule r occurring in a program $\mathcal{P}$, a *ground instance* of r is a rule obtained from r by replacing every variable X in r by $\sigma(X)$, where σ is a mapping from the variables occurring in r to the terms in $U_\mathcal{P}$. We denote by $ground(\mathcal{P})$ the set of all the ground instances of the rules occurring in $\mathcal{P}$ (note that $ground(\mathcal{P})$ may be infinite).

An *interpretation* for $\mathcal{P}$ is a consistent set of ground literals, that is, an interpretation is a subset I of $B_\mathcal{P} \cup \neg.B_\mathcal{P}$ such that $I \cap \neg.I = \emptyset$. A ground literal L is *true* (resp., *false*) w.r.t. I if $L \in I$ (resp., $L \in \neg.I$). If a ground literal is neither true nor false w.r.t. I, then it is *undefined* w.r.t. I. We denote by I^+ and I^-, respectively, the set of positive literals and the set of negative literals occurring in I. $\bar{I}$ denotes the set of undefined literals w.r.t. I. The interpretation I is *total* if $\bar{I}$ is empty (that is, $I^+ \cup \neg.I^- = B_\mathcal{P}$); otherwise, I is *partial.*

[1] Clearly, user programs are finite sets of rules; we allow a program to be (countably) infinite, as, for technical reasons, it is useful that the instantiation of the program is a program as well.

[2] If no constants appear in the program, then one is added arbitrarily.

Let r be a ground rule in $ground(\mathcal{P})$. The head of r is *true* w.r.t. I if $H(r) \cap I \neq \emptyset$. The body of r is *true* w.r.t. I if $B(r) \subseteq I$ and is *false* w.r.t. I if $B(r) \cap \neg.I \neq \emptyset$ (i.e., some literal in $B(r)$ is false w.r.t. I). The rule r is *satisfied* (or *true*) w.r.t. I if its head is true w.r.t. I or its body is false w.r.t. I.

A *model* for $\mathcal{P}$ is a total interpretation M for $\mathcal{P}$ such that every rule $r \in ground(\mathcal{P})$ is true w.r.t. M. A model M for $\mathcal{P}$ is *minimal* if no model N for $\mathcal{P}$ exists such that N^+ is a proper subset of M^+. The set of all minimal models for $\mathcal{P}$ is denoted by $\mathrm{MM}(\mathcal{P})$.

Note that, under these definitions, the word *interpretation* refers to a possibly partial interpretation, while a model is always a total interpretation.

The first proposal for assigning a semantics to a disjunctive logic program appears in [44], which presents a model-theoretic semantics for positive programs. According to [44], the semantics of a program $\mathcal{P}$ is described by the set $\mathrm{MM}(\mathcal{P})$ of the minimal models for $\mathcal{P}$. Observe that every program $\mathcal{P}$ admits at least one minimal model, that is, for every program $\mathcal{P}$, $\mathrm{MM}(\mathcal{P}) \neq \emptyset$ holds.

Example 2.1 [3] For the positive program $P_1 = \{a \vee b \leftarrow\}$, the (total) interpretations $\{a,\ \neg b\}$ and $\{b,\ \neg a\}$ are its minimal models (i.e., $\mathrm{MM}(\mathcal{P}) = \{\ \{a,\ \neg b\},\ \{b,\ \neg a\}\ \}$).

For the program $P_2 = \{a \vee b \leftarrow;\ b \leftarrow a;\ a \leftarrow b\}$, $\{a, b\}$ is the only minimal model. □

As far as general programs (i.e., programs where negation may appear in the bodies) are concerned, a number of semantics have been recently proposed [10, 30, 44, 52, 50, 51, 56, 58] (see [1, 15, 37] for comprehensive surveys). One generally acknowledged proposal is the extension of the stable model semantics [31] to take into account disjunction [30, 50]. Given a program $\mathcal{P}$ and a total interpretation I, the *Gelfond-Lifschitz (GL) transformation* of $\mathcal{P}$ w.r.t. I, denoted $\mathcal{P}^I$, is the set of positive rules defined as follows:

$$\begin{array}{ll} \mathcal{P}^I = \{a_1 \vee \cdots \vee a_n \leftarrow b_1, \cdots, b_k \mid & a_1 \vee \cdots \vee a_n \leftarrow b_1, \cdots, b_k, \neg b_{k+1}, \cdots, \neg b_m \in ground(\mathcal{P}) \\ & \textit{and } \neg b_i \in I, \text{ for all } k < i \leq m\} \end{array}$$

We note that, in the definition of the GL transformation reported in [31, 50], the condition $\neg b_i \in I$ above is $b_i \notin I$, since total interpretations are represented there by sets of atoms rather than sets of literals as in our case.

Definition 2.2 [50] Let I be a total interpretation for a program $\mathcal{P}$. I is a *(disjunctive) stable model* for $\mathcal{P}$ if $I \in \mathrm{MM}(\mathcal{P}^I)$ (i.e., I is a minimal model of the positive program $\mathcal{P}^I$). The set of all stable models for $\mathcal{P}$ is denoted by $STM(\mathcal{P})$. □

Example 2.3 Let $P = \{a \vee b \leftarrow c;\quad b \leftarrow \neg a, \neg c;\quad a \vee c \leftarrow \neg b\}$.
Consider $I = \{b,\ \neg a,\ \neg c\}$. Then, $P^I = \{a \vee b \leftarrow c;\quad b \leftarrow\}$.
It is easy to verify that I is a minimal model for P^I; thus, I is a stable model for P.

$M_1 = \{la\_broken, \neg ra\_broken, \neg can\_write\}$ and $M_2 = \{ra\_broken, \neg la\_broken, can\_write\}$ are the stable models of the program $\mathcal{P}_{Max}$ of Example 1.1. □

Clearly, if $\mathcal{P}$ is positive, then $\mathcal{P}^I$ coincides with $ground(\mathcal{P})$. It turns out that for a positive program, minimal and stable models coincide.

[3] For simplicity, we use propositional examples, in which the programs coincide with their ground instantiations.

3 Unfounded Sets for Disjunctive Logic Programs

In this section, we extend to disjunctive logic programs the notion of *unfounded sets* given for disjunction-free logic programs in [64], and then investigate some properties of unfounded sets. Although in this paper we use unfounded sets mainly for total interpretations (because they are used to characterize stable models), for the sake of generality and to preserve a close analogy between our definition of unfounded sets and the classical one [64], we define the notion of unfounded sets for general (possibly partial) interpretations.

Definition 3.1 Let I be an interpretation for a program $\mathcal{P}$. A set $X \subseteq B_{\mathcal{P}}$ of ground atoms is an *unfounded set* for $\mathcal{P}$ w.r.t. I if, for each $a \in X$, for each rule $r \in ground(\mathcal{P})$ such that $a \in H(r)$, at least one of the following conditions holds:

1. $B(r) \cap \neg.I \neq \emptyset$, that is, the body of r is false w.r.t. I
2. $B^+(r) \cap X \neq \emptyset$, that is, some positive body literal belongs to X
3. $(H(r) - X) \cap I \neq \emptyset$, that is, an atom in the head of r, distinct from a and other elements in X, is true w.r.t. I.

□

Conditions 1 and 2 are the same as in the classical definition of unfounded sets [64]. Intuitively, the third condition expresses that an atom a, occurring in the head of rule r, is not derivable from r if the head of r is already true; in other words, there exists an atom b in the head of r which is true in I (indeed, inferences follow a minimality criterion). However, the truth of the atom b in the head is not taken into account (for the unfoundedness of a) if b itself is unfounded, that is, $b \in X$ (see example below).

Informally, unfounded atoms are not derivable from the rules of $\mathcal{P}$, and thus, according to the closed-world assumption, they should be considered false.

Example 3.2 Consider the program $\mathcal{P} = \{a \vee b \leftarrow\}$ and let $I = \{b\}$ be a (partial) interpretation for it. The set $X = \{a\}$ is an unfounded set of $\mathcal{P}$ w.r.t. I. Indeed, the unique rule of $\mathcal{P}$ (with a in the head) satisfies Condition 3 of Definition 3.1, as $(H(r) - X) \cap I = \{b\} \neq \emptyset$. The unfoundedness of a meets the intuition that, since b is true, a cannot be derived from the program (and therefore can be assumed false).

Consider the interpretation $I = \{a, b\}$ (again for the program $\mathcal{P} = \{a \vee b \leftarrow\}$). In this case, both set $X = \{a\}$ and set $Y = \{b\}$ are unfounded sets for $\mathcal{P}$ w.r.t. I. Intuitively, this means that the presence either of a or of b is not justified in the interpretation (we should choose a or b, but we cannot accept both).

For the program $\mathcal{P} = \{\ a \vee b \leftarrow\ ;\ a \leftarrow b\ ;\ b \leftarrow a\ \}$ and the interpretation $I = \{a, b\}$, the only unfounded set is $\emptyset$ (i.e., no atom can be assumed false). □

Note that Definition 3.1 generalizes the definition given for disjunction-free logic programs in [64].

Proposition 3.3 *Let $\mathcal{P}$ be a disjunction-free program and I be an interpretation for $\mathcal{P}$. $X \subseteq B_{\mathcal{P}}$ is an unfounded set for $\mathcal{P}$ w.r.t. I according to Definition 3.1 iff it is an unfounded set for $\mathcal{P}$ w.r.t. I according to [64].*

Proof. Conditions 1 and 2 of Definition 3.1 are exactly the same as in [64]. Moreover, for disjunction-free programs, Condition 3 is never satisfied, as $H(r) - X = \emptyset$. Indeed, from Definition 3.1, $H(r) = \{a\}$ and $a \in X$. □

Nevertheless, for disjunctive logic programs, unlike traditional logic programs, the union of two unfounded sets is not necessarily an unfounded set.

Example 3.4 Consider again the program $\mathcal{P} = \{a \vee b \leftarrow\}$ and let $I = \{a, b\}$ be the interpretation. In Example 3.2, we saw that both $\{a\}$ and $\{b\}$ are unfounded sets for $\mathcal{P}$ w.r.t. I, but we can easily verify that the union $X = \{a, b\}$ is not. Indeed, although a (resp., b) is true in I, b (resp., a) is not unfounded because a (resp., b) is also in X (hence Condition 3 of Definition 3.1 does not hold). Intuitively, this represents the fact that we can falsify either a or b, but at least one of them must remain true (to satisfy the rule $a \vee b \leftarrow$). □

In traditional logic programming, the union of all unfounded sets w.r.t. an interpretation I is also an unfounded set w.r.t. I (called the *greatest unfounded set*) that includes all unfounded sets w.r.t. I [64]. Example 3.4 shows that in disjunctive logic programming, this is not generally true. We thus denote by $\mathbf{I}_{\mathcal{P}}$ the set of all interpretations of $\mathcal{P}$ which possess this property. More precisely, an interpretation I of $\mathcal{P}$ is in $\mathbf{I}_{\mathcal{P}}$ iff the union of all unfounded sets for $\mathcal{P}$ w.r.t. I is an unfounded set for $\mathcal{P}$ w.r.t. I as well. Given $I \in \mathbf{I}_{\mathcal{P}}$, in analogy with traditional logic programming, we call the union of all unfounded sets for $\mathcal{P}$ w.r.t. I the *greatest unfounded set* of $\mathcal{P}$ w.r.t. I, and we denote it by $GUS_{\mathcal{P}}(I)$.

Because the existence of the greatest unfounded set is not in general guaranteed, the question of whether an interpretation I is in $\mathbf{I}_{\mathcal{P}}$ naturally comes up. This is an interesting problem from the viewpoint of complexity. The best upper bound we can provide is $\Delta_2^P[O(\log n)]$ – the class of decision problems solvable in polynomial time by a deterministic Turing machine which can use a logarithmic number of calls to an NP oracle. This class is "mildly" harder than NP or co-NP. Completeness for $\Delta_2^P[O(\log n)]$ would entail NP-hardness; however, it is not clear how to reduce an NP-complete problem to this problem.

Proposition 3.5 [4] *Let $\mathcal{P}$ be a propositional program and I an interpretation for $\mathcal{P}$. Deciding whether $I \in \mathbf{I}_{\mathcal{P}}$ (i.e., I admits $GUS_{\mathcal{P}}(I)$) is in $\Delta_2^P[O(\log n)]$.*

Proof. Observe first that deciding whether an unfounded set X for $\mathcal{P}$ w.r.t. I with cardinality greater than a given integer k exists can be done in NP. Indeed, we can proceed as follows: guess $X \subseteq B_{\mathcal{P}}$; verify (i) that X is an unfounded set for $\mathcal{P}$ w.r.t. I and (ii) that the cardinality of X is greater than k (both (i) and (ii) are clearly polynomial).

Now, by binary search on $[0...|B_{\mathcal{P}}|]$, determine the number of elements Σ of the unfounded sets for $\mathcal{P}$ w.r.t. I that are of maximum cardinality. This is done by a logarithmic number of calls to an (NP) oracle deciding whether there exists an unfounded set for $\mathcal{P}$ w.r.t. I with cardinality greater than a given integer k ($k = \frac{|B_{\mathcal{P}}|}{2}$ on the first call; then, if the oracle answers "no," $k = \frac{|B_{\mathcal{P}}|}{4}$; otherwise, k is set to $\frac{3|B_{\mathcal{P}}|}{4}$, and so on, in accordance with standard binary search). Then, we call an NP oracle once to decide whether there exist two unfounded sets X and Y for $\mathcal{P}$ w.r.t. I such that (i) $|X| = \Sigma$ and (ii) $Y - X \neq \emptyset$. A "no" answer from the latter oracle recognizes that I has the greatest unfounded set $GUS_{\mathcal{P}}(I)$.

[4] We analyze the complexity of the propositional case (i.e., for finite ground programs); however, the complexity results extend easily to the data complexity [65] of function-free programs with variables.

Thus, deciding whether I is in $\mathbf{I}_{\mathcal{P}}$ lies in $\Delta_2^P[O(\log n)]$. $\Box$

Next, we show that there is a class of interpretations, called *unfounded-free interpretations*, which always have the greatest unfounded set.

Definition 3.6 Let I be an interpretation for a program $\mathcal{P}$. I is *unfounded-free* if $I \cap X = \emptyset$ for each unfounded set X for $\mathcal{P}$ w.r.t. I. $\Box$

Proposition 3.7 *Let I be an unfounded-free interpretation for a program $\mathcal{P}$. Then*

a. $\mathcal{P}$ has the greatest unfounded set $GUS_{\mathcal{P}}(I)$ (i.e., $I \in \mathbf{I}_{\mathcal{P}}$), and

b. $GUS_{\mathcal{P}}(I)$ is computable in polynomial time if $\mathcal{P}$ is a propositional program.

Proof. *Point a.* We will prove that the union U of a (possibly infinite) family $\mathcal{F}$ of unfounded sets for $\mathcal{P}$ w.r.t. I, is an unfounded set for $\mathcal{P}$ w.r.t. I as well. The statement will then follow immediately.

Since I is unfounded-free, Condition 3 of Definition 3.1 reduces to $H(r) \cap I \neq \emptyset$ (as $(H(r) - X) \cap I = H(r) \cap (I - X)$, which, in turn, is equal to I, since each unfounded set X is disjoint from I).

Now, let a be an atom in U. a must belong to some unfounded set $X \in \mathcal{F}$. Since X is an unfounded set and $X \subseteq U$, we have that (by Definition 3.1) for each rule $r \in ground(\mathcal{P})$ either (1) $B(r)$ is false w.r.t. I or (2) $B^+(r) \cap U \neq \emptyset$ or (3) $H(r) \cap I \neq \emptyset$. Thus, U is an unfounded set for $\mathcal{P}$ w.r.t. I.

Point b. From Point a, I has the greatest unfounded set $GUS_{\mathcal{P}}(I)$. We give a polynomial time construction of the set of ground atoms in $B_{\mathcal{P}} - GUS_{\mathcal{P}}(I)$ (from which $GUS_{\mathcal{P}}(I)$ is efficiently derivable) – a similar proof scheme was adopted in [64] to demonstrate the tractability of the well-founded semantics.

We compute $B_{\mathcal{P}} - GUS_{\mathcal{P}}(I)$ as the least fixpoint of a suitable operator Φ_I. Define Φ_I as follows:

$$\begin{array}{rlll} \Phi_I: & B_{\mathcal{P}} & \rightarrow & B_{\mathcal{P}} \\ & Y & \mapsto & \{a \mid \exists r \in \mathcal{P} \textit{ with } a \in H(r) \textit{ s.t. } B(r) \cap \neg.I = \emptyset \ \ \wedge \ B^+(r) \subseteq Y \\ & & & \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad \wedge \ H(r) \cap I = \emptyset\} \end{array}$$

The Φ_I operator is monotonically increasing in the complete lattice $< B_{\mathcal{P}}, \subseteq>$. Thus, the sequence $\phi_0 = I$, $\phi_k = I \cup \Phi_I(\phi_{k-1})$ is monotonically increasing and converges finitely to a limit ϕ_λ (as it is finitely bound by the Herbrand base). Since each application of Φ_I is feasible in polynomial time, and since the limit ϕ_λ is reached in a number λ of applications of Φ_I which is polynomial in $|B_{\mathcal{P}}|$, ϕ_λ is computable in polynomial time. We next show that $\phi_\lambda = B_{\mathcal{P}} - GUS_{\mathcal{P}}(I)$, and, as a consequence, we derive that the computation of $GUS_{\mathcal{P}}(I)$ is tractable.

$\phi_\lambda \subseteq B_{\mathcal{P}} - GUS_{\mathcal{P}}(I)$. We proceed by induction on the index k of the sequence $\{\phi_k\}_{k \in \mathcal{N}}$. The basis of the induction ($k = 0$) holds since $\phi_0 = I$ and I is unfounded-free (thus, $\phi_0 = I \subseteq B_{\mathcal{P}} - GUS_{\mathcal{P}}(I)$). Assuming now that $\phi_{k-1} \subseteq B_{\mathcal{P}} - GUS_{\mathcal{P}}(I)$ (*inductive hypothesis*), we show that $\phi_k \subseteq B_{\mathcal{P}} - GUS_{\mathcal{P}}(I)$ (i.e., ϕ_k does not contain any unfounded element). We have to show that for every $a \in \phi_k$ there exists a rule with a in the head, which violates all conditions of

Definition 3.1 (for the unfounded set $GUS_{\mathcal{P}}(I)$). Let $a \in \phi_k$; then, $a \in I \cup \Phi_I(\phi_{k-1})$. Now, if $a \in I$, we are done, as I is unfounded-free. Otherwise, by definition of Φ_I, we have that

$$\exists r \in \mathcal{P} \textit{ with } a \in H(r) \textit{ s.t. } B(r) \cap \neg.I = \emptyset \quad \wedge \quad B^+(r) \subseteq \phi_{k-1} \quad \wedge \quad H(r) \cap I = \emptyset$$

Now, $B(r) \cap \neg.I = \emptyset$ violates Condition 1 of Definition 3.1. Moreover, $B^+(r) \subseteq \phi_{k-1}$ violates Condition 2 of Definition 3.1, since by inductive hypothesis, ϕ_{k-1} does not contain any element of $GUS_{\mathcal{P}}(I)$. Furthermore, $H(r) \cap I = \emptyset$ implies that even Condition 3 of Definition 3.1 cannot be satisfied, as $(H(r) - X) \cap I \subseteq H(r) \cap I$. Hence, $\phi_k \subseteq B_{\mathcal{P}} - GUS_{\mathcal{P}}(I)$ and $\phi_\lambda \subseteq B_{\mathcal{P}} - GUS_{\mathcal{P}}(I)$.

$\phi_\lambda \supseteq B_{\mathcal{P}} - GUS_{\mathcal{P}}(I)$. From the maximality of $GUS_{\mathcal{P}}(I)$, it is sufficient to demonstrate that $X = B_{\mathcal{P}} - \phi_\lambda$ is an unfounded set for $\mathcal{P}$ w.r.t. I. Let $a \in X$. We prove that every rule with a in the head satisfies (at least) one of the conditions of Definition 3.1. Since $\phi_\lambda = \phi_{\lambda+1}$ and $a \notin \phi_\lambda$, then $a \notin \Phi_I(\phi_\lambda)$. Hence, $\forall r \in \mathcal{P}$ with $a \in H(r)$: either (i) $B(r) \cap \neg.I \neq \emptyset$ or (ii) $B^+(r) \not\subseteq \phi_\lambda$ or (iii) $H(r) \cap I \neq \emptyset$. Now, if (i) is satisfied, Condition 1 of Definition 3.1 is verified. If $B^+(r) \not\subseteq \phi_\lambda$ ((ii) holds), then $B^+(r) \cap B_{\mathcal{P}} - \phi_\lambda \neq \emptyset$, and Condition 2 of Definition 3.1 is satisfied (as $B_{\mathcal{P}} - \phi_\lambda = X$). Finally, since $I \subseteq \phi_\lambda$, we have that $I = I - X$. Hence, $H(r) \cap I = H(r) \cap (I - X) = (H(r) - X) \cap I$. Thus, $H(r) \cap I \neq \emptyset$ implies Condition 3 of Definition 3.1 holds. (We are done.) □

Thus, an unfounded-free interpretation always admits the greatest unfounded set. This set is efficiently computable, and the proof of Point b of Proposition 3.7 provides an effective algorithm for its computation. Unfortunately, the next proposition shows that, unless P = NP, we cannot efficiently test whether I is unfounded-free.

Proposition 3.8 *Let $\mathcal{P}$ be a propositional program and I be an interpretation for $\mathcal{P}$. Deciding whether I is unfounded-free is* co-NP*-complete.*

Proof. *Membership in* co-NP. The complementary problem of checking whether I is unfounded-free, that is, deciding whether I is not unfounded-free, can be done in NP: guess a subset X of $B_{\mathcal{P}}$ and check in polynomial time that (i) X is an unfounded set for $\mathcal{P}$ w.r.t. I and (ii) $X \cap I \neq \emptyset$. Hence, deciding whether I is unfounded-free belongs to the class co-NP.
Hardness for co-NP. It is well known that deciding whether a model M is stable is co-NP-hard [43, 18, 19, 17]. In the next section (see Theorem 4.6), however, we show that this decision problem reduces to verifying whether M is unfounded-free. Thus, deciding whether an interpretation is unfounded-free is co-NP-hard as well. □

It is worth noting that the containment of the set of unfounded-free interpretations in $\mathbf{I}_{\mathcal{P}}$ is, in general, strict. For instance, for each disjunction-free program, every interpretation admits the greatest unfounded set [64], yet there are interpretations of disjunction-free programs which are not unfounded-free (e.g., for $\mathcal{P} = \{a \leftarrow a\}$ the interpretation $I = \{a\}$ is not unfounded-free and admits the greatest unfounded set $\{a\}$).

Interestingly, the $GUS_{\mathcal{P}}$ operator is monotonic.

Proposition 3.9 *Let I and J be interpretations in $\mathbf{I}_{\mathcal{P}}$. If $I \subseteq J$, then $GUS_{\mathcal{P}}(I) \subseteq GUS_{\mathcal{P}}(J)$.*

Proof. We first show that if X is an unfounded set for $\mathcal{P}$ w.r.t. I, then X is an unfounded set for $\mathcal{P}$ w.r.t. J as well.

From the definition of unfounded sets and the condition $I \subseteq J$, we have that, for each $a \in X$ and for each $r \in \mathit{ground}(\mathcal{P})$ such that $a \in H(r)$, one of the following conditions holds: either (1) the body $B(r)$ is false w.r.t. I and, hence, is false w.r.t. J; or (2) $B(r) \cap X \neq \emptyset$; or (3) $(H(r) - X) \cap I \neq \emptyset$ and, hence, $(H(r) - X) \cap J \neq \emptyset$. Thus, X is an unfounded set for $\mathcal{P}$ w.r.t. J.

Now, by definition, $\mathit{GUS}_{\mathcal{P}}(I)$ is an unfounded set for $\mathcal{P}$ w.r.t. I. Hence, from the property proven above, $\mathit{GUS}_{\mathcal{P}}(I)$ is an unfounded set for $\mathcal{P}$ w.r.t. J as well. Therefore, $\mathit{GUS}_{\mathcal{P}}(I)$ is included in $\mathit{GUS}_{\mathcal{P}}(J)$, the greatest unfounded set for $\mathcal{P}$ w.r.t. J (as $\mathit{GUS}_{\mathcal{P}}(J)$ is the union of all unfounded sets for $\mathcal{P}$ w.r.t. J), that is, $\mathit{GUS}_{\mathcal{P}}(I) \subseteq \mathit{GUS}_{\mathcal{P}}(J)$ holds. □

We conclude this section by providing an equivalent characterization of the unfounded-free property in the domain of total interpretations.

Proposition 3.10 *Let I be a total interpretation for a program $\mathcal{P}$. I is unfounded-free iff no nonempty set of atoms contained in I is an unfounded set for $\mathcal{P}$ w.r.t. I.*

Proof. ($\Longleftarrow$) We prove the contrapositive, that is, if I is not unfounded-free, then there exists a non-empty subset of I which is an unfounded set for $\mathcal{P}$ w.r.t. I. To this end, assume that I is not unfounded-free. Then, from Definition 3.6, there exists an unfounded set X for $\mathcal{P}$ w.r.t. I such that $X \cap I \neq \emptyset$. We now show that the set $Y = X \cap I$ is an unfounded set for $\mathcal{P}$ w.r.t. I. Let a be an element of Y. Since X is an unfounded set for $\mathcal{P}$ w.r.t. I, for each rule $r \in \mathit{ground}(\mathcal{P})$ such that $a \in H(r)$, by Definition 3.1 at least one of the following conditions holds: (1) the body of r is false w.r.t. I (i.e., $B(r) \cap \neg.I \neq \emptyset$); (2) $B^+(r) \cap X \neq \emptyset$; (3) $(H(r) - X) \cap I \neq \emptyset$. Consider now Condition 2. Since I is total, for each literal $L \in B^+(r)$ either $L \in I^+$ or $\neg.L \in I^-$. Therefore, $B^+(r) \cap X \neq \emptyset$ implies either $B^+(r) \cap Y \neq \emptyset$ or $B^+(r) \cap \neg.I^- \neq \emptyset$ (i.e., $B(r)$ is false w.r.t. I). Concerning Condition 3, it is obvious that it is equivalent to $(H(r) - Y) \cap I \neq \emptyset$. Therefore, we have that, for each rule $r \in \mathit{ground}(\mathcal{P})$ such that $a \in H(r)$, either $B(r)$ is false in I or $B(r) \cap Y \neq \emptyset$ or $(H(r) - Y) \cap I \neq \emptyset$ holds. Thus, we may conclude that Y is an unfounded set for $\mathcal{P}$ w.r.t. I.
($\Longrightarrow$) If a nonempty subset Y of I is an unfounded set for $\mathcal{P}$ w.r.t. I, then I is clearly not unfounded-free, since Y violates the unfounded-free condition of Definition 3.6. □

4 Stable Models and Unfounded Sets

In this section, we provide some characterizations of (stable) models in terms of unfounded sets. We wish to emphasize that, since for disjunction-free programs the definitions of stable models and unfounded sets given in this paper coincide with the classical definitions given in [31] and [64], respectively, all results proven here hold also for disjunction-free programs (under the classical definitions of stable models and unfounded sets).

The next proposition shows that models are characterized by the property that all false literals are unfounded.

Proposition 4.1 *Let M be a total interpretation for a program $\mathcal{P}$. Then M is a model for $\mathcal{P}$ iff $\neg.M^-$ is an unfounded set for $\mathcal{P}$ w.r.t. M.*

Proof. ($\Longleftarrow$) We prove that if M is not a model, then $\neg.M^-$ is not an unfounded set for $\mathcal{P}$ w.r.t. M. Assume that M is not a model. Then a rule $r \in \mathit{ground}(\mathcal{P})$ exists such that

$B(r) \subseteq M$ and $H(r)$ is not true in M, that is, $H(r) \cap M^+ = \emptyset$. Because M is total, it turns out that $H(r) \subseteq \neg.M^-$ holds. It is easy to verify that r does not satisfy any of the unfoundedness conditions of Definition 3.1 for the set $\neg.M^-$. Indeed, the body of r is not false w.r.t. M, nor does it contain any atom in $\neg.M^-$ (as $B(r) \subseteq M$). Further, $(H(r) - \neg.M^-) \cap M = \emptyset$ holds, as $H(r) \subseteq \neg.M^-$. Therefore, $\neg.M^-$ is not an unfounded set for $\mathcal{P}$ w.r.t. M.
($\Longrightarrow$) Assume that $\neg.M^-$ is not an unfounded set for $\mathcal{P}$ w.r.t. M. Thus, there is an atom $a \in \neg.M^-$ such that there exists a rule $r \in ground(\mathcal{P})$ having a in its head for which none of the unfoundedness conditions of Definition 3.1 hold. In particular, $B(r)$ is not false in M, which implies that $B(r)$ is true in M ($B(r) \subseteq M$), as M is total by hypothesis. Moreover, $(H(r) - \neg.M^-) \cap M = \emptyset$ holds; thus, $H(r) \cap M = \emptyset$. Indeed, $(H(r) - \neg.M^-) \cap M = H(r) \cap (M - \neg.M^-) = H(r) \cap M$. Therefore, M is not a model, as r is not satisfied in M since its body is true while its head is false ($B(r) \subseteq M$ and $H(r) \cap M = \emptyset$). $\Box$

Next we prove that the class of stable models for $\mathcal{P}$ coincides with that of the unfounded-free models for $\mathcal{P}$. To this end, we need some preliminary results.

Lemma 4.2 *Let M be a model for a program $\mathcal{P}$. If M is unfounded-free, then M is a minimal model for $\mathcal{P}$.*

Proof. We show that if M is not minimal, then it is not unfounded-free. If M is not minimal, then the existence of another model M_1 such that $X = M^+ - M_1^+ \neq \emptyset$ is implied. Since M_1 is a model, for each rule r whose head contains some element in X, the following holds: either (i) $H(r) \cap M_1^+ \neq \emptyset$ (the head is true); or (ii) $B^+(r) \cap \neg.M_1^- \neq \emptyset$ (some positive literal in the body is false); or (iii) $B^-(r) \cap \neg.M_1^+ \neq \emptyset$ (some negative literal is false). Hence each rule r, whose head contains some element in X, satisfies at least one condition of Definition 3.1: If $H(r) \cap M_1^+ \neq \emptyset$, then Condition 3 is satisfied, as $(H(r) - X) \cap M^+ \neq \emptyset$. If $B^+(r) \cap \neg.M_1^- \neq \emptyset$, then either Condition 1 or Condition 2 is verified, as $\neg.M_1^- = \neg.M^- \cup \neg.X$. Finally, if $B^-(r) \cap \neg.M_1^+ \neq \emptyset$, then Condition 2 is clearly satisfied, as $M_1^+ \subset M^+$. Therefore, M is not unfounded-free since X is an unfounded set for $\mathcal{P}$ w.r.t. M. $\Box$

The converse of Lemma 4.2 does not hold in general, since some minimal models may not be unfounded-free (even for normal programs, as pointed out in [41]).

Example 4.3 Let $\mathcal{P} = \{a \leftarrow \neg b\}$. It is easy to see that the model $M = \{b, \neg a\}$ is minimal but not unfounded-free. $\Box$

Nevertheless, for positive logic programs, the converse of Lemma 4.2 does hold. The minimal model semantics proposed by Minker for positive programs [44] is thus exactly characterized by the unfounded-free condition.

Proposition 4.4 *Let M be a model for a positive program $\mathcal{P}$. M is a minimal model for $\mathcal{P}$ iff it is unfounded-free.*

Proof. By Lemma 4.2, it remains to show that every minimal model for $\mathcal{P}$ is unfounded-free.

We proceed by contradiction. Assume M is not unfounded-free, and let $X \subseteq M^+$ be a nonempty unfounded set for $\mathcal{P}$ w.r.t. M. We show that the total interpretation $M_1 = (M - X) \cup \neg.X$ is a model for $\mathcal{P}$ (contradicting the minimality of M). To this end, it clearly suffices to prove that the rules whose heads are true in M but not in M_1 are satisfied in M_1

(i.e., have a false body w.r.t. M_1). Let $r \in \mathit{ground}(\mathcal{P})$ be any such rule and let a be an atom in the head of r such that $a \in X$ (such an a exists since $H(r) \cap M \neq \emptyset$ while $H(r) \cap M_1 = \emptyset$). Since X is an unfounded set for $\mathcal{P}$ w.r.t. M, for each rule with a in the head, either $B(r)$ is false in M (i.e., since $\mathcal{P}$ is positive, $B(r) \cap \neg.M^- \neq \emptyset$) or $B^+(r) \cap X \neq \emptyset$ holds (note that Condition 3 of Definition 3.1 cannot hold as $(H(r) - X) \cap M = \emptyset$ by assumption). In the former case, by construction of M_1, $B(r)$ is false w.r.t. M_1 as well (since $\neg.M^- \subset \neg.M_1^-$). In the second case, $B(r)$ is clearly false w.r.t. M_1, as $B^+(r) \cap X \neq \emptyset$ implies that $B^+(r) \cap \neg.M^- \neq \emptyset$. Hence, in every case, r is satisfied w.r.t. M_1, since its body is false w.r.t. M_1. Therefore, we can conclude that M_1 is a model, thus contradicting the minimality of M. □

Before giving the characterization of stable models in terms of unfounded sets, we need a further lemma stating that the unfounded-free property is preserved under the GL transformation.

Lemma 4.5 *Let M be a total interpretation for a program $\mathcal{P}$. M is unfounded-free for $\mathcal{P}$ iff M is unfounded-free for $\mathcal{P}^M$.*

Proof. Recall that the GL transformation $\mathcal{P}^M$ of $\mathcal{P}$ is obtained from $\mathit{ground}(\mathcal{P})$ by dropping all rules where some negative body literal is false w.r.t. M and then eliminating all negative literals from the bodies of the remaining rules.
($\Longrightarrow$) If X is not an unfounded set for $\mathcal{P}$ w.r.t. M, then for each $a \in X$, there exists a rule $r \in \mathit{ground}(\mathcal{P})$ that violates all conditions of Definition 3.1. In particular, from the violation of Condition 1, the body of r is true; thus, the image r' of r (under the GL transformation) is in $\mathcal{P}^M$. Clearly, r' violates all conditions of Definition 3.1 for $\mathcal{P}^M$ w.r.t. M. Therefore, if X is not an unfounded set for $\mathcal{P}$ w.r.t. M then X is not an unfounded set for $\mathcal{P}^M$ w.r.t. M as well. Now, if M is unfounded-free for $\mathcal{P}$, then, from Proposition 3.10, every nonempty subset X of M^+ is not an unfounded set for $\mathcal{P}$ w.r.t. M. As a consequence, from the property proven above, we derive that every nonempty subset X of M^+ is not an unfounded set for $\mathcal{P}^M$ w.r.t. M, that is, M is unfounded-free for $\mathcal{P}^M$.
($\Longleftarrow$) Let X be an unfounded set for $\mathcal{P}$ w.r.t. M. Since every rule in $\mathcal{P}^M$ with a in its head is (under the GL transformation) the image r' of a rule $r \in \mathit{ground}(\mathcal{P})$, X is an unfounded set for $\mathcal{P}^M$ w.r.t. M if for each $a \in X$, for each rule $r \in \mathit{ground}(\mathcal{P})$ with head a, the image r' of r satisfies at least one condition of Definition 3.1 for $\mathcal{P}^M$.

Now, since X is an unfounded set for $\mathcal{P}$ w.r.t. M, for each $a \in X$, for each rule $r \in \mathit{ground}(\mathcal{P})$ either (i) the body of r is false w.r.t. M, that is, (i.1) $B^+(r) \cap \neg.I^- \neq \emptyset$ or (i.2) $B^-(r) \cap \neg.I^+ \neq \emptyset$; or (ii) $B^+(r) \cap X \neq \emptyset$; or (iii) $(H(r) - X) \cap I \neq \emptyset$. Case (i.2) implies that r has no image in $\mathcal{P}^M$. If Condition (i.1), Condition (ii), or Condition (iii) holds for r, however, then the same condition is verified for r', as $B^+(r) = B^+(r')$ and $H(r) = H(r')$.

Therefore, if X is an unfounded set for $\mathcal{P}$ w.r.t. M, then X is an unfounded set for $\mathcal{P}^M$ w.r.t. M as well. Thus, if M is not unfounded-free for $\mathcal{P}$, then M is not unfounded-free for $\mathcal{P}^M$. As a consequence, M unfounded-free for $\mathcal{P}^M$ implies M unfounded-free for $\mathcal{P}$. □

We are now in a position to prove the main result of this section: in the general case of programs with negation, the unfounded-free condition singles out the stable models.

Theorem 4.6 *Let M be a model for a program $\mathcal{P}$. M is stable iff M is unfounded-free.*

Proof. ($\Longrightarrow$) Let M be a stable model for $\mathcal{P}$. By Definition 2.2, M is a minimal model of (the positive) program $\mathcal{P}^M$. Consequently, from Proposition 4.4, M is an unfounded-free model for

$\mathcal{P}^M$. Therefore, from Lemma 4.5, M is an unfounded-free model for $\mathcal{P}$.
($\Longleftarrow$) Let M be an unfounded-free model for $\mathcal{P}$. It is easily recognized that M is a model for $\mathcal{P}^M$. Moreover, since M is an unfounded-free model for $\mathcal{P}$, from Lemma 4.5 M is an unfounded-free model for $\mathcal{P}^M$. Therefore, by Lemma 4.2, M is a minimal model for $\mathcal{P}^M$. Thus, M is a stable model according to Definition 2.2. □

Thus, both minimal models for positive programs and stable models for general programs are exactly the unfounded-free models.

From Theorem 4.6, it immediately follows that a stable model has associated with it the greatest unfounded set (by virtue of Proposition 3.7). Moreover, the minimality of stable models is immediately derived.

Corollary 4.7 *Every stable model for $\mathcal{P}$ is a minimal model for $\mathcal{P}$.*

Proof. Immediate from Theorem 4.6 and Lemma 4.2. □

The next theorem supplies another interesting declarative characterization of stable models.

Theorem 4.8 *Let M be a total interpretation for a program $\mathcal{P}$. M is a stable model for $\mathcal{P}$ iff $\neg.M^- = GUS_{\mathcal{P}}(M)$.*[5]

Proof. ($\Longleftarrow$) By Proposition 4.1, M is a model. Moreover, since $\mathcal{P}$ has the greatest unfounded set w.r.t. M, every unfounded set w.r.t. M is included in $GUS_{\mathcal{P}}(M)$, that is, every unfounded set w.r.t. M is included in $\neg.M^-$. Therefore, M^+ does not contain any nonempty unfounded set w.r.t. M, as it is disjoint from $\neg.M^-$. Hence, from Proposition 3.10, M is unfounded-free. It follows that, by virtue of Theorem 4.6, M is a stable model, since it is an unfounded-free model.
($\Longrightarrow$) Since M is a stable model, it is unfounded-free (by Theorem 4.6). Therefore, from Theorem 3.7, M has the greatest unfounded set $GUS_{\mathcal{P}}(M)$. From Definition 3.6, every unfounded set is included in $B_{\mathcal{P}} - M^+$, which coincides with $\neg.M^-$, as M is total. The union $GUS_{\mathcal{P}}(M)$ of all unfounded sets for $\mathcal{P}$ w.r.t. M is hence contained in $\neg.M^-$. Moreover, since M is a model, by Proposition 4.1, the whole set $\neg.M^-$ is an unfounded set for $\mathcal{P}$ w.r.t. M (i.e., $\neg.M^- \subseteq GUS_{\mathcal{P}}(M)$). In sum, $\mathcal{P}$ has the greatest unfounded set w.r.t. M, and it coincides with $\neg.M^-$, that is, $\neg.M^- = GUS_{\mathcal{P}}(M)$. □

Observe that (disjunctive) stable models have been defined classically by a property of their true literals: a model M is stable if all its positive literals are derivable from the program (assuming M^-) [31, 50]. Theorem 4.8 gives an (equivalent) characterization of stable models which is "dual" to the classical one: a total interpretation M is a stable model if all and only its false literals cannot be derived from the program (assuming M), that is, all and only its false literals are unfounded.

We next derive that partial interpretations cannot be extended to stable models if they are not unfounded-free.

Corollary 4.9 *Let I be a partial interpretation for a program $\mathcal{P}$. If I is not unfounded-free, then no stable model for $\mathcal{P}$ contains I.*

[5]Observe that this condition implicitly requires that M has the greatest unfounded set, i.e., $M \in \mathbf{I}_{\mathcal{P}}$.

Proof. By Definition 3.6, if I is not unfounded-free, then there exists an unfounded set X for $\mathcal{P}$ w.r.t. I such that $X \cap I^+ \neq \emptyset$. Now, if M is a model containing I ($I \subseteq M$), then it is easy to see that X is an unfounded set for $\mathcal{P}$ w.r.t. M as well. Therefore, M is not unfounded-free (as $X \cap M^+ \neq \emptyset$). Thus, by Theorem 4.6, M is not a stable model. $\Box$

Corollary 4.9 can be helpful when computing the stable models for $\mathcal{P}$: whenever one realizes that a partial interpretation is not unfounded-free, the interpretation can be discarded, as it will not lead to any stable model.

5 A Fixpoint Semantics for Stable Models

In this section, we extend the $\mathcal{W}_\mathcal{P}$ operator, defined in [64] for disjunction-free programs, to the class of disjunctive logic programs. Then we show that the stable models of a disjunctive logic program coincide exactly with the (total) fixpoints of $\mathcal{W}_\mathcal{P}$ (i.e., M is stable iff it is a fixpoint of $\mathcal{W}_\mathcal{P}$). The results in this section generalize the analogous results shown in [64] for disjunction-free programs (namely, Theorem 5.4, Corollary 5.6, and Corollary 5.7).

We start by providing an extension to disjunctive logic programs of the immediate consequence operator $T_\mathcal{P}$ defined in [64] for three-valued interpretations of normal logic programs.

Definition 5.1 Let $\mathcal{P}$ be a program. Define the $\mathcal{T}_\mathcal{P}$ operator as follows:

$$\begin{array}{lll} \mathcal{T}_\mathcal{P}: & 2^{B_\mathcal{P} \cup \neg.B_\mathcal{P}} & \rightarrow 2^{B_\mathcal{P}} \\ & I & \mapsto \{a \in B_\mathcal{P} \mid \exists r \in ground(\mathcal{P}) \ s.t. \ a \in H(r), \ H(r) - \{a\} \subseteq \neg.I, \\ & & \qquad and \ B(r) \subseteq I\}. \end{array}$$ $\Box$

Intuitively, given an interpretation I, $\mathcal{T}_\mathcal{P}$ derives a set of atoms belonging to every model containing I (i.e. atoms that are surely needed to extend I to a model). Note that, unlike other extensions of $T_\mathcal{P}$ to disjunctive logic programs, $\mathcal{T}_\mathcal{P}$ is deterministic, that is, its result is a single set of atoms rather than a family of sets of atoms.

Definition 5.2 Let $\mathcal{P}$ be a program. Define the $\mathcal{W}_\mathcal{P}$ operator as follows:[6]

$$\begin{array}{lll} \mathcal{W}_\mathcal{P}: & \mathbf{I}_\mathcal{P} & \rightarrow 2^{B_\mathcal{P} \cup \neg.B_\mathcal{P}} \\ & I & \mapsto \mathcal{T}_\mathcal{P}(I) \cup \neg.\mathit{GUS}_\mathcal{P}(I). \end{array}$$ $\Box$

The next proposition confirms the intuition that Definition 5.2 extends to disjunctive logic programs the $\mathcal{W}_\mathcal{P}$ operator defined in [64] for disjunction-free programs (whose least fixpoint is the well-founded model).

Proposition 5.3 *Let $\mathcal{P}$ be a disjunction-free program. Then the $\mathcal{W}_\mathcal{P}$ operator of Definition 5.2 exactly coincides with $\mathcal{W}_\mathcal{P}$ operator defined in [64].*

Proof. First observe that the two definitions of $\mathcal{W}_\mathcal{P}$ assign the same domain to the operator: because $\mathcal{P}$ is a disjunction-free program, its every interpretation admits the greatest unfounded set. Thus, $\mathbf{I}_\mathcal{P}$ coincides with the set of all (three valued) interpretations for $\mathcal{P}$.

[6]Recall that $\mathbf{I}_\mathcal{P}$ is the set of interpretations having the greatest unfounded set.

We must prove that given an interpretation I, the two versions of $\mathcal{W}_{\mathcal{P}}(I)$ coincide. In both versions, the positive part of $\mathcal{W}_{\mathcal{P}}(I)$ is the result of the application of the immediate consequence operator $\mathcal{T}_{\mathcal{P}}$, while the negative part of $\mathcal{W}_{\mathcal{P}}(I)$ is the greatest unfounded set for $\mathcal{P}$ w.r.t. I. $\mathcal{P}$ being a disjunction-free program, $\mathcal{T}_{\mathcal{P}}$ as given in Definition 5.1 coincides with $T_{\mathcal{P}}$ of [64], since, $\forall r \in \mathcal{P}$, $H(r)$ is a singleton and, as a consequence, the condition $H(r)-\{a\} \subseteq \neg.I$ is trivially satisfied. From Proposition 3.3, it immediately follows that the two versions of the greatest unfounded set also coincide. The proposition is therefore proven. □

The next theorem relates the stable models of $\mathcal{P}$ to the fixpoints of $\mathcal{W}_{\mathcal{P}}$. Note that, because $\mathcal{W}_{\mathcal{P}}$ is defined on the domain $\mathbf{I}_{\mathcal{P}}$, every fixpoint of $\mathcal{W}_{\mathcal{P}}$ by definition admits the greatest unfounded set (since each fixpoint of $\mathcal{W}_{\mathcal{P}}$ must belong to the domain $\mathbf{I}_{\mathcal{P}}$ of $\mathcal{W}_{\mathcal{P}}$).

Theorem 5.4 *Let M be a total interpretation for program $\mathcal{P}$. M is a stable model for $\mathcal{P}$ iff M is a fixpoint of $\mathcal{W}_{\mathcal{P}}$.*

Proof. ($\Longrightarrow$) Since M is a stable model, by Theorem 4.6 M is unfounded-free. Thus, M is in the domain $\mathbf{I}_{\mathcal{P}}$ of $\mathcal{W}_{\mathcal{P}}$, by virtue of Proposition 3.7.

Now, according to Definition 5.2, we have to prove that $M^+ = \mathcal{T}_{\mathcal{P}}(M)$ and that $M^- = \neg.\mathit{GUS}_{\mathcal{P}}(M)$. The latter condition holds by Theorem 4.8, as M is a stable model, so only the former condition remains to be proven.

$\mathcal{T}_{\mathcal{P}}(M) \supseteq M^+$: Observe that, by Theorem 4.6, M is unfounded-free, and thus, from Proposition 3.10, no subset of M is an unfounded set for $\mathcal{P}$ w.r.t. M. This is true for any subset of the form $\{a\}$. Hence, for each $a \in M^+$, there exists a rule $r \in ground(\mathcal{P})$ with a in its head, which violates all unfoundedness conditions of Definition 3.1. In particular, $B(r)$ is not false w.r.t. M, and thus, given that M is total, $B(r) \subseteq M$. Moreover, $(H(r) - \{a\}) \cap M = \emptyset$ holds; thus, as M is total, $H(r) - \{a\} \subseteq \neg.M$. Therefore, for each $a \in M^+$, $a \in \mathcal{T}_{\mathcal{P}}(M)$, that is, $\mathcal{T}_{\mathcal{P}}(M) \supseteq M^+$ holds (see Definition 5.1).

$M^+ \supseteq \mathcal{T}_{\mathcal{P}}(M)$: Let $a \in \mathcal{T}_{\mathcal{P}}(M)$. From Definition 5.1, $\exists r \in ground(\mathcal{P})$ with a in the head such that: (i) $B(r) \subseteq M$, and (ii) $H(r) - \{a\} \subseteq \neg.M$. It follows that a must be in M^+; otherwise M would not be a model.

($\Longleftarrow$) Let M be a fixpoint of $\mathcal{W}_{\mathcal{P}}$. Then, by Definition 5.2, M admits the greatest unfounded set $\mathit{GUS}_{\mathcal{P}}(M)$ (since $M \in \mathbf{I}_{\mathcal{P}}$), which coincides with $\neg.M^-$ (i.e., $M^- = \neg.\mathit{GUS}_{\mathcal{P}}(M)$). Therefore, by Theorem 4.8, M is a stable model for $\mathcal{P}$. □

Observe that our Theorem 5.4 is a generalization of Theorem 5.4 of [64] to the class of disjunctive logic programs. It is also worth noting that the $\mathcal{W}_{\mathcal{P}}$ operator is a skeptical operator and preserves the "correctness" of the interpretations, as it does not make any unjustified choice. In other words, if I is an interpretation contained in a stable model M, then $\mathcal{W}_{\mathcal{P}}(I)$ may add to I some literals of M but it never "contradicts" M (i.e., no inconsistency with M is introduced by $\mathcal{W}_{\mathcal{P}}(I)$).

Proposition 5.5 *Let I be an interpretation for a program $\mathcal{P}$, and let M be a stable model of $\mathcal{P}$. If $I \subseteq M$, then*

(a) I belongs to the domain $\mathbf{I}_{\mathcal{P}}$ of $\mathcal{W}_{\mathcal{P}}$, and

(b) $\mathcal{W}_{\mathcal{P}}(I) \subseteq M$.

Proof. *Point* (a). Since M is a stable model, by Theorem 4.6 it is unfounded-free. Therefore, I is unfounded-free. Indeed, from Definition 3.1, every unfounded set X w.r.t. I is also an unfounded set w.r.t. M. Therefore, for each unfounded set X w.r.t. I, we have that $X \cap I \subseteq X \cap M = \emptyset$. Thus, from Proposition 3.7, I belongs to the domain $\mathbf{I}_{\mathcal{P}}$ of $\mathcal{W}_{\mathcal{P}}$, which proves Point (a).

Point (b). M is a stable model, so, by Theorem 4.8, $M^- = \neg.\mathit{GUS}_{\mathcal{P}}(M)$ and, by Proposition 3.9, $\mathit{GUS}_{\mathcal{P}}(I) \subseteq \mathit{GUS}_{\mathcal{P}}(M)$ (as $I \subseteq M$). Hence, $\mathcal{W}_{\mathcal{P}}(I)^- = \neg.\mathit{GUS}_{\mathcal{P}}(I) \subseteq \neg.\mathit{GUS}_{\mathcal{P}}(M) = M^-$.

To complete the proof of Point (b), only $\mathcal{W}_{\mathcal{P}}(I)^+ \subseteq M^+$ remains to be demonstrated. Let $a \in \mathcal{W}_{\mathcal{P}}(I)^+$. Since $\mathcal{W}_{\mathcal{P}}(I)^+ = \mathcal{T}_{\mathcal{P}}(I)$, by Definition 5.1 $\exists r \in \mathit{ground}(\mathcal{P})$ such that $a \in H(r)$, $H(r) - \{a\} \subseteq \neg.I$, and $B(r) \subseteq I$. Consequently, as $I \subseteq M$, $\exists r \in \mathit{ground}(\mathcal{P})$ such that $a \in H(r)$, $H(r) - \{a\} \subseteq \neg.M$, and $B(r) \subseteq M$. Hence, a must belong to M^+; otherwise M would not be a model. Thus, $\forall a \in \mathcal{W}_{\mathcal{P}}(I)^+$, $a \in M^+$, that is, $\mathcal{W}_{\mathcal{P}}(I)^+ \subseteq M^+$. □

The elegant fixpoint characterization of stable models given by Theorem 5.4 does not provide suggestions for constructively building the stable models. Because the computation of the stable models of function-free programs is an important issue that will be dealt with in the next section, we will investigate how the $\mathcal{W}_{\mathcal{P}}$ operator can be employed to construct the stable models of function-free programs.

To start we prove that, as in normal logic programming, the (possibly partial) least fixpoint $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ of $\mathcal{W}_{\mathcal{P}}$ is contained in every stable model.

Proposition 5.6 *Given a function-free program $\mathcal{P}$, let $\{W_n\}_{n \in \mathcal{N}}$ be the sequence whose nth term is the n-fold application of the $\mathcal{W}_{\mathcal{P}}$ operator on the empty set (i.e., $W_0 = \emptyset$, $W_n = \mathcal{W}_{\mathcal{P}}(W_{n-1})$). Then*

(a) $\{W_n\}_{n \in \mathcal{N}}$ converges to a limit $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$, and

(b) for each stable model M for $\mathcal{P}$, $M \supseteq \mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$.

Proof. Observe that we have to prove that the sequence $\{W_n\}_{n \in \mathcal{N}}$ is well-defined (i.e., for each n in $\mathcal{N}$, W_n is an interpretation in $\mathbf{I}_{\mathcal{P}}$).

In the case where $\mathcal{P}$ admits some stable model, the statement follows immediately from Proposition 5.5. Indeed, from Proposition 5.5 by applying simple induction, it is easy to see that $\{W_n\}_{n \in \mathcal{N}}$ is well defined and that, for each stable model M for $\mathcal{P}$, $M \supseteq W_n$ for every element W_n of the sequence. The convergence of the sequence is a consequence of the monotonicity of $\mathcal{W}_{\mathcal{P}}$ ($\mathit{GUS}_{\mathcal{P}}$ is monotonic by Proposition 3.9, and $\mathcal{T}_{\mathcal{P}}$ is clearly monotonic) and of the finiteness of $B_{\mathcal{P}}$ (which is an upper bound for $\{W_n\}_{n \in \mathcal{N}}$).

Consider now the case where the program $\mathcal{P}$ has no stable model. Point (b) is trivially true. To prove Point (a), we demonstrate that, for each $n \in \mathcal{N}$, the following two conditions hold: (i) W_n is an interpretation (i.e., it is consistent), and (ii) W_n is unfounded-free.

As expected, we proceed by induction on the index n of the sequence. The basis of the induction is trivial. Assuming now that both conditions hold for each $k \leq n-1$ (*inductive hypothesis*), we demonstrate that they hold for W_n.
(i) Consistency of W_n. By contradiction, assume that W_n is not consistent (i.e., $W_n \cap \neg.W_n \neq \emptyset$). Let q be an atom in $W_n \cap \neg.W_n$. By the definition of W_n, $q \in \mathcal{T}_{\mathcal{P}}(W_{n-1}) \cap \mathit{GUS}_{\mathcal{P}}(W_{n-1})$ (recall that $W_n = \mathcal{W}_{\mathcal{P}}(W_{n-1}) = \mathcal{T}_{\mathcal{P}}(W_{n-1}) \cup \neg.\mathit{GUS}_{\mathcal{P}}(W_{n-1})$). On the one hand, since $q \in \mathit{GUS}_{\mathcal{P}}(W_{n-1})$, for each rule $r \in \mathit{ground}(\mathcal{P})$ with q in its head, at least one of the following

conditions holds: (1) $B(r) \cap \neg.W_{n-1} \neq \emptyset$, (2) $B(r) \cap GUS_{\mathcal{P}}(W_{n-1}) \neq \emptyset$, or (3) $(H(r) - GUS_{\mathcal{P}}(W_{n-1})) \cap W_{n-1} \neq \emptyset$ (see Definition 3.1). On the other hand, since $q \in \mathcal{T}_{\mathcal{P}}(W_{n-1})$, we have that $\exists r' \in ground(\mathcal{P})$ such that $B(r') \subseteq W_{n-1}$ and $(H(r') - \{q\}) \subseteq \neg.W_{n-1}$ (see Definition 5.1). Now, for rule r' we find that none of the conditions hold. Condition (1) does not hold for r', as otherwise $W_{n-1} \cap \neg.W_{n-1} \neq \emptyset$ (contradicting the consistency of W_{n-1} assumed in the inductive hypothesis). Condition (2) does not hold for r', as otherwise $W_{n-1} \cap GUS_{\mathcal{P}}(W_{n-1}) \neq \emptyset$ (contradicting the unfounded-free property of the inductive hypothesis). Condition (3) does not hold for r', as $(H(r') - GUS_{\mathcal{P}}(W_{n-1})) \subseteq (H(r') - \{q\}) \subseteq \neg.W_{n-1}$ (thus, $(H(r') - GUS_{\mathcal{P}}(W_{n-1})) \cap W_{n-1} \neq \emptyset$ would contradict the consistency of W_{n-1} assumed in the inductive hypothesis). Therefore, $q \notin GUS_{\mathcal{P}}(W_{n-1})$ (i.e., a contradiction arises). Hence, W_n is consistent (point (i) is proven).
(ii) W_n is unfounded-free. By contradiction, assume that there exists an unfounded set X for $\mathcal{P}$ w.r.t. W_n such that $X \cap W_n \neq \emptyset$. Then, by Definition 3.1, for each rule $r \in ground(\mathcal{P})$ with an atom from X in the head, at least one of the following holds: (1) $B(r) \cap \neg.W_n \neq \emptyset$, (2) $B(r) \cap X \neq \emptyset$, or (3) $(H(r) - X) \cap W_n \neq \emptyset$. Now, let m be the least index such that $X \cap W_m \neq \emptyset$ (note that $X \cap W_{m-1} = \emptyset$). Given $q \in X \cap W_m$, every rule with q in its head must verify at least one of the three conditions above (as q belongs to the unfounded set X w.r.t. W_n).

In contrast, since $W_m = \mathcal{W}_{\mathcal{P}}(W_{m-1})$, then $\exists r' \in ground(\mathcal{P})$ such that $B(r') \subseteq W_{m-1}$ and $(H(r') - \{q\}) \subseteq \neg.W_{m-1}$ (see Definition 5.1). For rule r' we observe that $W_{m-1} \subseteq W_n$ and that none of the conditions hold. Condition (1) does not hold for r', as otherwise $W_n \cap \neg.W_n \neq \emptyset$ (contradicting the consistency of W_n proven in point (i) above). Condition (2) does not hold for r', as $B(r')$ is included in W_{m-1} which does not intersect X (we assumed that W_m is the first element of $\{W_n\}_{n \in \mathcal{N}}$ intersecting X). Condition (3) does not hold for r', as $(H(r') - X) \subseteq (H(r') - \{q\}) \subseteq \neg.W_{m-1} \subseteq \neg.W_n$ (thus, $(H(r') - X) \cap W_n \neq \emptyset$ would contradict the consistency of W_n proven in point (i) above). Therefore, X is not an unfounded set w.r.t. W_n (i.e., a contradiction arises), since rule r' with an element from X in its head does not satisfy any condition of Definition 3.1. Hence, W_n is unfounded-free (point (ii) is proven).

Once we know that $\{W_n\}_{n \in \mathcal{N}}$ is well defined, as in the case where $\mathcal{P}$ admits some stable model, we easily recognize that $\{W_n\}_{n \in \mathcal{N}}$ (finitely) converges to a limit $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$. □

It is worth noting that the iterated applications of the $\mathcal{W}_{\mathcal{P}}$ operator starting from the empty set yield unfounded-free interpretations. Nevertheless, in general, $\mathcal{W}_{\mathcal{P}}(I)$ may not be unfounded-free even if I is unfounded-free. For instance, consider program $\mathcal{P} = \{b \leftarrow \neg a;\ a \leftarrow b\}$ and interpretation $I = \{b, \neg a\}$. It is easy to see that I is unfounded-free, while $\mathcal{W}_{\mathcal{P}}(I) = \{a, b\}$ is not.

In disjunction-free logic programming, whenever $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is total, it is the unique stable model (see Corollary 5.6 of [64]). The next corollary generalizes this result to disjunctive logic programs.

Corollary 5.7 *Let $\mathcal{P}$ be a function-free program. If $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is a total interpretation, then it is the unique stable model for $\mathcal{P}$.*

Proof. If $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is a total interpretation, then, by Theorem 5.4, it is a stable model. Furthermore, from Proposition 5.6, for each stable model M, $M \supseteq \mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$. Hence, since $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is total, for each stable model M, $M = \mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$, that is, $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is the unique stable model for $\mathcal{P}$. □

As we intuitively expected, for disjunction-free programs, $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is the traditional well-founded model.

Corollary 5.8 *Let $\mathcal{P}$ be a function-free program. If $\mathcal{P}$ is disjunction-free, then $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is the well-founded model for $\mathcal{P}$ as defined in [64].*

Proof. Immediate from Propositions 5.6 and 5.3. $\square$

To enhance understanding of $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$, we conclude by showing that $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ coincides with the well-founded model of a disjunction-free program $\mathcal{P}'$ obtained by "shifting" some head atoms to the bodies of the rules.

Given a (disjunctive) program $\mathcal{P}$, we denote by $sh(\mathcal{P})$ the disjunction-free program obtained from $\mathcal{P}$ by substituting every rule of the form $a_1 \vee \ldots \vee a_k \leftarrow b_1, \ldots, b_m, \neg c_1, \ldots, \neg c_n$ by the k rules
$a_i \leftarrow b_1, \ldots, b_m, \neg c_1, \ldots, \neg c_n, \neg a_1, \ldots, \neg a_{i-1}, \neg a_{i+1}, \ldots, \neg a_k \quad (1 \leq i \leq k)$.

Lemma 5.9 *Let I be an unfounded-free interpretation for a function-free program $\mathcal{P}$. Then X is an unfounded set for $\mathcal{P}$ w.r.t. I iff X is an unfounded set for $sh(\mathcal{P})$ w.r.t. I.*

Proof. ($\Longrightarrow$) Let $a \in X$ and consider a rule $r \in ground(\mathcal{P})$ with $a \in H(r)$. Notice that there is only one rule r' with head a in $ground(sh(\mathcal{P}))$ corresponding to r in $ground(\mathcal{P})$. Since X is an unfounded set for $\mathcal{P}$ w.r.t. I, at least one condition of Definition 3.1 is verified. If $B(r) \cap \neg.I \neq \emptyset$, then $B(r') \cap \neg.I \neq \emptyset$, since $B(r) \subseteq B(r')$. If $B^+(r) \cap X \neq \emptyset$, then $B^+(r') \cap X \neq \emptyset$, since $B^+(r) = B^+(r')$. Finally, consider the case $(H(r) - X) \cap I \neq \emptyset$. Since I is unfounded-free, the previous condition is equivalent to $(H(r) - \{a\}) \cap I \neq \emptyset$. As $\neg.(H(r) - \{a\}) \subseteq B(r')$, then $B(r') \cap \neg.I \neq \emptyset$ because at least $\neg.(H(r) - \{a\}) \cap \neg.I \neq \emptyset$.

This proves the result for any $a \in X$ and arbitrary rule r and therefore for every rule with head a in $ground(sh(\mathcal{P}))$.
($\Longleftarrow$) The proof proceeds along the same lines as in ($\Longrightarrow$) above. The first case for $sh(\mathcal{P})$ reduces to either the first or the third case for $\mathcal{P}$. The second case always reduces to the second case. $\square$

Lemma 5.10 *Let I be an interpretation for a function-free program $\mathcal{P}$. Then, $\mathcal{T}_{\mathcal{P}}(I) = \mathcal{T}_{sh(\mathcal{P})}(I)$.*

Proof. By Definition 5.1, $a \in \mathcal{T}_{\mathcal{P}}(I)$ iff there is a rule r in $ground(\mathcal{P})$ such that $a \in H(r)$, $H(r) - \{a\} \subseteq \neg.I$ and $B(r) \subseteq I$. This is equivalent to having a rule r' in $ground(sh(\mathcal{P}))$ with head a such that $B(r') \subseteq I$ since $B(r') = B(r) \cup \neg.(H(r') - \{a\}) \subseteq I$. $\square$

Theorem 5.11 *Let $\mathcal{P}$ be a function-free program. Then, $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ coincides with the well-founded model of $sh(\mathcal{P})$.*

Proof. From Lemmas 5.9 and 5.10, it follows that $\mathcal{W}_{\mathcal{P}}(I) = \mathcal{W}_{sh(\mathcal{P})}(I)$ for every unfounded-free interpretation I. From the proof of Proposition 5.6, the iterated application of $\mathcal{W}_{\mathcal{P}}$ starting from the empty set yields unfounded-free interpretations. Thus, $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset) = \mathcal{W}^{\omega}_{sh(\mathcal{P})}(\emptyset)$. $\square$

As an important consequence, we obtain that the computation of $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is tractable for disjunctive logic programs.

Proposition 5.12 *Given a propositional program $\mathcal{P}$, $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is computable in polynomial time.*

Proof. From Theorem 5.11, $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ coincides with the well-founded model of the (polynomially computable) program $\mathcal{P}'$. The result thus follows from [64]. □

It is worth noting that in $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$, no positive conclusion is drawn from disjunctive rules, since $\mathcal{W}_{\mathcal{P}}$ is a deterministic operator and makes no choice in the presence of a disjunct. Thus, as is not true for normal logic programs, where $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is a (partial) model and can be thought of as the intended meaning of $\mathcal{P}$ [64], for disjunctive logic programs $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is not in general a model and cannot be seen as a meaningful semantics for the program. For instance, consider the simple program $\mathcal{P} = \{a \vee b\}$. $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is $\emptyset$, so it is not a model, and it cannot be seen as the semantics of $\mathcal{P}$ since too many atoms are left undefined. Nevertheless, there are simple cases where $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ provides a reasonable meaning for the program. For instance, consider the program $\{b \leftarrow \neg c;\ a \vee b \leftarrow\}$. We have $\mathcal{W}_{\mathcal{P}}(\emptyset) = \{\neg c\}$, $\mathcal{W}_{\mathcal{P}}(\{\neg c\}) = \{b, \neg c\}$, and $\mathcal{W}_{\mathcal{P}}(\{b, \neg c\}) = \{\neg c, b, \neg a\} = \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$. $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is total and is thus the unique stable model of the program capturing the intended meaning of $\mathcal{P}$. In this case, $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ works well because the disjunctive rule of $\mathcal{P}$ is satisfied since some head atom is derivable from disjunction-free rules. Indeed, $a \vee b \leftarrow$ is satisfied by the atom b derived from $b \leftarrow \neg c$; the $\mathit{GUS}_{\mathcal{P}}$ operator hence is able to infer the falsity of a.

Thus, from the semantic viewpoint, only the total fixpoints of $\mathcal{W}_{\mathcal{P}}$ are meaningful, as they coincide with the stable models of $\mathcal{P}$. Nevertheless, since $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is contained in every stable model (see Proposition 5.6) and is efficiently computable (see Proposition 5.12), the computation of $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ can be a good starting point for algorithms computing the stable models.

6 Computation of Stable Models

In this section, we study the computation of stable models. Exploiting the results presented in the previous sections, we design an algorithm that computes the stable models of any disjunctive deductive database (i.e., function-free disjunctive program). We limit ourselves to the function-free case because for general disjunctive programs (where the presence of function symbols makes the Herbrand Base infinite), the problem is known to be infeasible in general. There may exist continuum many stable models, and the problem of deciding whether a recursively enumerable interpretation is a stable model is Π_2^0-hard [42, 62].

Remark. All the programs considered in this section are function-free. For simplicity, we refer to them simply as *programs* instead of *function-free programs.* □

To start, consider the naive algorithm for the computation of stable models shown in Figure 1. This algorithm, based on a trivial guess-and-check strategy, reveals the presence of two main sources of complexity in the computation of the stable models which interact, in a sense, orthogonally: (1) the exponential number of interpretations that are "candidates" to be stable models, and (2) the check of the unfounded-free condition (or, equivalently, the stability condition), which has been shown to be a co-NP-hard decision problem (see Proposition 3.8).

The next two subsections focus on these sources of complexity. For clarity of exposition, we start in Section 6.1 with the check of the unfounded-free property; then, Section 6.2 provides a method that allows us to reduce the number of candidate interpretations to be considered in practice.

Input: A disjunctive logic program $\mathcal{P}$.
Output: The stable models of $\mathcal{P}$.
begin
  for each interpretation I of $\mathcal{P}$ **do**
    if I is an unfounded-free model
      then output *"I is a stable model of $\mathcal{P}$"*;
  end_for;
end.

Figure 1: *Naive Guess-and-Check Algorithm for the Computation of Stable Models*

Combining the results in the two subsections yields an effective algorithm for the computation of the stable models.

6.1 Checking the Unfounded-Free Property

Checking the unfounded-free property is difficult in general. Indeed, by virtue of Theorem 4.6, this task is equivalent to verifying the stability condition, which is known to be a co-NP-hard decision problem [43, 18, 19, 17] (see Proposition 3.8). Recent studies [7, 8], however, have proven that minimal model checking — the hardest part of stable model checking — can be efficiently performed for a relevant class of programs, called *head-cycle-free* (*HCF*) programs. We next provide an algorithm for checking the unfounded-free property which runs in polynomial time for HCF programs and for general programs limits the inefficient part of the computation to only the components of the program that are not HCF. Our algorithm pays attention to head-cycle-free programs for the following main reasons:

- A fundamental efficiency requirement for algorithms solving untractable problems is that they be polynomial on the main classes of instances of the problem which are known to be tractable. To our knowledge, the set of HCF programs is the most relevant class of programs for which stable model checking is known to be tractable (note that all disjunction-free programs are contained in this set). Good algorithms for the computation of disjunctive stable models should therefore handle HCF programs efficiently.

- It has recently been proven that under brave (resp., cautious) reasoning, HCF programs (with stable model semantics) express *all* properties in NP (resp., co-NP) [20]. More important, several problems in NP can be expressed in a very simple and natural way by HCF programs [20, 7, 8] – a number of rules with HCF disjunction may "guess" the solution which is then "checked" by disjunction-free rules. For instance, *all* problems suggested as benchmarks for nonmonotonic reasoning systems in [12] are naturally expressed by HCF programs.

With every program $\mathcal{P}$, we associate a directed graph $DG_{\mathcal{P}} = (\mathcal{N}, E)$, called the *dependency graph* of $\mathcal{P}$, in which (i) each predicate of $\mathcal{P}$ is a node in $\mathcal{N}$ and (ii) there is an arc in E directed from a node a to a node b iff there is a rule r in $\mathcal{P}$ such that b and a are the predicates of a positive literal appearing in $H(r)$ and $B(r)$, respectively.

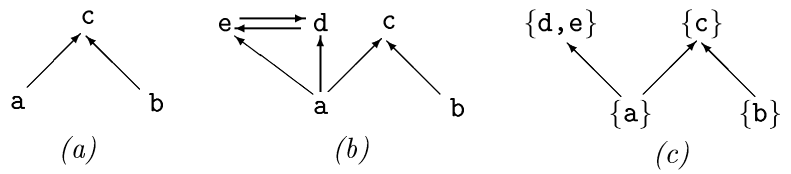

Figure 2: *Graphs (a) $DG_{\mathcal{P}_1}$, (b) $DG_{\mathcal{P}_2}$, and (c) $\hat{DG}_{\mathcal{P}_2}$*

The graph $DG_{\mathcal{P}}$ singles out the dependencies of the head predicates of a rule r from the positive predicates in its body.[7]

Example 6.1 Consider the program $\mathcal{P}_1$ consisting of the following rules:[8]

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

The dependency graph $DG_{\mathcal{P}_1}$ of $\mathcal{P}_1$ is depicted in Figure 2a. (Note that, since the sample programs are propositional, the nodes of the sample graphs in Figure 2 are atoms, as atoms coincide with predicates in this case.)

Consider now program $\mathcal{P}_2$, obtained by adding to $\mathcal{P}_1$ the rules

$$d \vee e \leftarrow a \qquad\qquad d \leftarrow e \qquad\qquad e \leftarrow d, \neg b$$

The dependency graph $DG_{\mathcal{P}_2}$ is shown in Figure 2b. □

The dependency graphs allow us to single out HCF programs [7, 8].

A program $\mathcal{P}$ is *HCF* iff there is no clause r in $\mathcal{P}$ such that two predicates occurring in the head of r are in the same cycle of $DG_{\mathcal{P}}$.

Example 6.2 The dependency graphs given in Figure 2 reveal that program $\mathcal{P}_1$ of Example 6.1 is HCF and that program $\mathcal{P}_2$ is not HCF, as rule $d \vee e \leftarrow a$ contains in its head two predicates belonging to the same cycle of $DG_{\mathcal{P}_2}$. □

Our method for testing the unfounded-free property on HCF programs is based on a transformation $\mathcal{R}_{\mathcal{P},I}$ that, given a set X of ground atoms, derives the atoms in X which satisfy at least one of the unfoundedness conditions of Definition 3.1.

Definition 6.3 Let $\mathcal{P}$ be a program. Define the $\mathcal{R}_{\mathcal{P},I}$ operator as follows:

$$\begin{array}{llll} \mathcal{R}_{\mathcal{P},I}: & 2^{B_{\mathcal{P}}} & \rightarrow & 2^{B_{\mathcal{P}}} \\ & X & \mapsto & \{a \in X \mid \; \forall r \in ground(\mathcal{P}) \text{ with } a \in H(r), \\ & & & \qquad B(r) \cap (\neg.I \cup X) \neq \emptyset \;\text{ or }\; (H(r) - \{a\}) \cap I \neq \emptyset\} \end{array}$$

□

[7]Note that negative literals cause no arc in $DG_{\mathcal{P}}$.

[8]We point out again that we use propositional examples for simplicity, but the algorithm is defined for the general case of (function-free) programs with variables.

It is easy to see that $\mathcal{R}_{\mathcal{P},I}$ is a monotonic operator. Moreover, given a set X of ground atoms, it is obvious that the sequence $R_0 = X$, $R_n = \mathcal{R}_{\mathcal{P},I}(R_{n-1})$ decreases monotonically and converges finitely to a limit that we denote by $\mathcal{R}^{\omega}_{\mathcal{P},I}(X)$.

Intuitively, $\mathcal{R}_{\mathcal{P},I}(X)$ discards from X only elements for which there exists some rule violating all unfoundedness conditions for $\mathcal{P}$ w.r.t. I. Thus, $\mathcal{R}^{\omega}_{\mathcal{P},I}(X)$ contains the union of all unfounded sets for $\mathcal{P}$ w.r.t. I included in X. As a consequence, if $\mathcal{R}^{\omega}_{\mathcal{P},I}(I^+)$ is the empty set for a total interpretation I, then I is unfounded-free. This intuition is formalized by the following lemma and proposition.

Lemma 6.4 *Let $\mathcal{P}$ be a program, I be a total interpretation for $\mathcal{P}$, and $J \subseteq B_{\mathcal{P}}$. Every unfounded set for $\mathcal{P}$ w.r.t. I which is contained in J is also contained in $\mathcal{R}^{\omega}_{\mathcal{P},I}(J)$.*

Proof. Let $X \subseteq J$ be an unfounded set for $\mathcal{P}$ w.r.t. I. For each $a \in X$ and for each rule r such that $a \in H(r)$, at least one of the following conditions holds: (i) $B(r) \cap (\neg.I \cup X) \neq \emptyset$ or (ii) $(H(r) - X) \cap I \neq \emptyset$. Hence, as $\{a\} \subseteq X$, $(H(r) - \{a\}) \cap I \neq \emptyset$ as well. Then, from the definition of $\mathcal{R}_{\mathcal{P},I}$, $\mathcal{R}_{\mathcal{P},I}(X) = X$ holds and, since $\mathcal{R}_{\mathcal{P},I}$ is monotonic and $X \subseteq J$, $\mathcal{R}^{\omega}_{\mathcal{P},I}(J)$ must contain X. □

Proposition 6.5 *Let $\mathcal{P}$ be a program and I be a total interpretation for $\mathcal{P}$. If $\mathcal{R}^{\omega}_{\mathcal{P},I}(I^+) = \emptyset$, then I is unfounded-free.*

Proof. Follows immediately from Lemma 6.4. □

Therefore, $\mathcal{R}^{\omega}_{\mathcal{P},I}(I^+) = \emptyset$ is sufficient to guarantee that I is unfounded-free.

Example 6.6 Consider again the (HCF) program $\mathcal{P}_1$ of Example 6.1. $\mathcal{P}_1$ consists of the rules

$$r_1 : a \vee b \qquad\qquad r_2 : c \leftarrow a \qquad\qquad r_3 : c \leftarrow b$$

Given the total interpretation $I_1 = \{a, c, \neg b\}$, we have

$$\begin{aligned} R_0 &= I_1^+ = \{a, c\} \\ R_1 &= \mathcal{R}_{\mathcal{P}_1,I_1}(R_0) = \{c\} \\ R_2 &= \mathcal{R}_{\mathcal{P}_1,I_1}(R_1) = \emptyset = \mathcal{R}^{\omega}_{\mathcal{P}_1,I_1}(I_1^+) \end{aligned}$$

Indeed, a is not in $\mathcal{R}_{\mathcal{P}_1,I_1}(R_0)$ because for rule r_1 both $B(r_1) \cap (\neg.I_1 \cup R_0) = \emptyset$ (as $B(r_1)$ is empty) and $(H(r_1) - \{a\}) \cap I_1 = \emptyset$ (as $H(r_1) - \{a\} = \{b\}$) hold. Atom c is in $\mathcal{R}_{\mathcal{P}_1,I_1}(R_0)$, because both rules with c in their heads (namely, r_2 and rule r_3) verify the conditions required by the $\mathcal{R}_{\mathcal{P}_1,I_1}$ operator. In particular, $B(r_2) \cap R_0 = \{a\} \neq \emptyset$ and $B(r_3) \cap \neg.I_1 = \{b\} \neq \emptyset$. At the next step, c is not in $\mathcal{R}_{\mathcal{P}_1,I_1}(R_1)$, as $B(r_2) \cap (\neg.I_1 \cup R_1) \neq \emptyset$ does not hold.

I_1 thus verifies the hypothesis of Proposition 6.5 and it is indeed easy to see that I_1 is unfounded-free.

For another example consider, on the same program $\mathcal{P}_1$, total interpretation $I_2 = \{a, b, c\}$, which is not unfounded-free, as $\{a\}$ and $\{b\}$ are unfounded sets for $\mathcal{P}_1$ w.r.t. I_2. We have that

$$\begin{aligned} R_0 &= I_2^+ = \{a, b, c\} \\ R_1 &= \mathcal{R}_{\mathcal{P}_1,I_2}(R_0) = \{a, b, c\} = \mathcal{R}^{\omega}_{\mathcal{P}_1,I_2}(I_2^+) \end{aligned}$$

In this case, $\mathcal{R}^{\omega}_{\mathcal{P}_1,I_2}(I_2^+) \neq \emptyset$ and we cannot apply Proposition 6.5, as it states a sufficient condition for the unfounded-free property. However, we will see (Theorem 6.9) that, since $\mathcal{P}_1$ is HCF, condition $\mathcal{R}^{\omega}_{\mathcal{P}_1,I_2}(I_2^+) \neq \emptyset$ allows us to derive that I_2 is not unfounded-free. □

Condition $\mathcal{R}^{\omega}_{\mathcal{P},I}(I^+) = \emptyset$ is thus sufficient for ensuring that I is unfounded-free (actually, for HCF programs it is also necessary). Unfortunately, this condition is not necessary in the general case.

Example 6.7 Consider the (non-HCF) program $\mathcal{P}$ consisting of the following rules:

$$a \vee b \qquad\qquad a \leftarrow b \qquad\qquad b \leftarrow a$$

Given the total interpretation $I = \{a, b\}$, we have that $\mathcal{R}^{\omega}_{\mathcal{P},I}(I^+) = \{a, b\} \neq \emptyset$, while I is unfounded-free. □

Thus, condition $\mathcal{R}^{\omega}_{\mathcal{P},I}(I^+) = \emptyset$ is not in general necessary for an interpretation to be unfounded-free. Nevertheless, we next prove that for HCF programs this condition is necessary (observe that the program $\mathcal{P}$ of Example 6.7, for which condition $\mathcal{R}^{\omega}_{\mathcal{P},I}(I^+) = \emptyset$ is not necessary for I to be unfounded-free, is not HCF).

The proof of the theorem refers to the notion of a collapsed dependency graph, which is the graph $\hat{DG}_{\mathcal{P}}$ obtained from $DG_{\mathcal{P}}$ by collapsing each (maximal) strongly connected component into a single node. Thus, every node of $\hat{DG}_{\mathcal{P}}$ is a set Q of predicates. Moreover, given a set Q of predicates and a set J of atoms, we denote by $\frac{J}{Q}$ the subset of J whose predicates are from Q.

Example 6.8 The collapsed dependency graph $\hat{DG}_{\mathcal{P}_2}$ of $DG_{\mathcal{P}_2}$ is depicted in Figure 2c. □

Theorem 6.9 *Let $\mathcal{P}$ be an HCF program and I a total interpretation for it. Then I is unfounded-free iff $\mathcal{R}^{\omega}_{\mathcal{P},I}(I^+) = \emptyset$.*

Proof. From Proposition 6.5, only $\Longrightarrow$ remains to be proven.

Suppose that $X = \mathcal{R}^{\omega}_{\mathcal{P},I}(I^+)$ is not empty. Take a node Q of $\hat{DG}_{\mathcal{P}}$ such that: (a) some atom in X has a predicate in Q, and (b) there exists no other node Q' of $\hat{DG}_{\mathcal{P}}$ containing a predicate of some atom in X such that Q is (transitively) reachable from Q' (i.e., with a directed path from Q' to Q in $\hat{DG}_{\mathcal{P}}$).[9] Consider now set $\frac{X}{Q}$. We prove that $\frac{X}{Q}$ is an unfounded set for $\mathcal{P}$ w.r.t. I. Let r be a rule in $ground(\mathcal{P})$ with an atom from $\frac{X}{Q}$ in its head. Since $\mathcal{R}_{\mathcal{P},I}(X) = X$, r satisfies at least one of the following conditions: (i) $B(r) \cap \neg.I \neq \emptyset$ or (ii) $B^+(r) \cap X \neq \emptyset$ or (iii) $(H(r) - \{a\}) \cap I \neq \emptyset$. Because of our choice of the node Q, no atom in the body of r belongs to $X - \frac{X}{Q}$ and condition (ii) yields $B^+(r) \cap \frac{X}{Q} \neq \emptyset$. Furthermore, condition (iii) entails $(H(r) - \frac{X}{Q}) \cap I \neq \emptyset$, because if $H(r)$ contained another element from $\frac{X}{Q}$, distinct from a, then $\mathcal{P}$ would not be HCF, as the predicates of the elements in $\frac{X}{Q}$ belong to the same cycle of the dependency graph. Thus, $\frac{X}{Q}$ is an unfounded set for $\mathcal{P}$ w.r.t. I. □

Corollary 6.10 *Let $\mathcal{P}$ be a propositional HCF program and M be a model for $\mathcal{P}$. Recognizing whether M is a stable model is polynomial.*

Proof. It is easy to see that $\mathcal{R}^{\omega}_{\mathcal{P},I}(I^+)$ is efficiently computable. Thus, the statement follows from Theorems 4.6 and 6.9. □

[9]Note that the existence of such a node is guaranteed, because the collapsed graph has no cycle.

It is worth noting that, by virtue of Proposition 4.4, condition $\mathcal{R}^{\omega}_{\mathcal{P},I}(I^+) = \emptyset$ can also be employed to check that a model of a positive HCF program is minimal. Thus, Theorem 6.9 suggests a way of doing this check which is a simpler alternative to the algorithm proposed in [8].

From the above results, if the program is HCF we can efficiently check the unfounded-free property by testing whether $\mathcal{R}^{\omega}_{\mathcal{P},I}(I^+) = \emptyset$. Unfortunately, as illustrated by Example 6.7, we cannot use this test for recognizing the unfounded-free property in the general case, and we would have to resort to an inefficient algorithm that blindly controls every element in the power set of I^+ to see whether it is an unfounded set.

The complexity result of Proposition 3.8 indicates that we have no chance of finding an efficient algorithm for checking the unfounded-free property in the general case (unless P = NP). Nevertheless, we will introduce some optimizations that, also in non-HCF programs, strongly improve the efficiency of the naive algorithm for the test of the unfounded-free property.

The first optimization is a direct consequence of Lemma 6.4. Indeed, because of that result, we can limit our search of unfounded subsets of I^+ to the power set of $\mathcal{R}^{\omega}_{\mathcal{P},I}(I^+)$ (which may be substantially smaller than the power set of I^+).

The second important optimization is supported by Proposition 6.11, which shows that the test of the unfounded-free property can be performed one component at a time. In this way, (a) the number of sets to be checked to verify the property is drastically pruned, and (b) the efficient technique suggested by Theorem 6.9 can be employed on the HCF components of $\mathcal{P}$, limiting the inefficient part of the computation to only the components that are not HCF.

Given a set Q of predicates, we denote by $subp(Q)$ the *subprogram of* Q, which is the set of the rules in $ground(\mathcal{P})$ with a head predicate from Q. (Note that the same rule may occur in the subprograms of two different (collapsed) nodes Q_1 and Q_2 of $\hat{DG}_{\mathcal{P}}$.)

Proposition 6.11 *Let $\mathcal{P}$ be a program, and I a total interpretation for $\mathcal{P}$. I is not unfounded-free iff there exists a node Q of $\hat{DG}_{\mathcal{P}}$ such that $\frac{I^+}{Q}$ contains a nonempty unfounded set for $subp(Q)$ w.r.t. I.*

Proof. ($\Longrightarrow$) Let X be a nonempty unfounded set for $\mathcal{P}$ w.r.t. I, which is contained in I^+. As in the proof of Theorem 6.9, take a node Q of $\hat{DG}_{\mathcal{P}}$ such that: (a) some atom in X has a predicate in Q, and (b) there exists no other node Q' of $\hat{DG}_{\mathcal{P}}$ containing a predicate of some atom in X such that Q is (transitively) reachable from Q' (i.e., with a directed path from Q' to Q in $\hat{DG}_{\mathcal{P}}$). Note that the existence of such a node is guaranteed, because the collapsed graph has no cycle. Consider now set $\frac{X}{Q}$. We demonstrate that $\frac{X}{Q}$ is an unfounded set for $subp(Q)$ w.r.t. I. Let r be a rule in $subp(Q)$ with an atom from $\frac{X}{Q}$ in the head. Since $\frac{X}{Q}$ is a subset of X, $subp(Q) \subseteq ground(\mathcal{P})$, and X is an unfounded set for $\mathcal{P}$ w.r.t. I, r satisfies at least one of the following conditions: (1) $B(r) \cap \neg.I \neq \emptyset$, (2) $B^+(r) \cap X \neq \emptyset$, (3) $(H(r) - X) \cap I \neq \emptyset$. Now, Condition 1 is the same for set $\frac{X}{Q}$ as well. If Condition 2 is satisfied, then $B^+(r) \cap \frac{X}{Q} \neq \emptyset$, as Q is the lowest node of $\hat{DG}_{\mathcal{P}}$ with predicates appearing in X (Condition b above). Finally, if Condition 3 holds, then $(H(r) - \frac{X}{Q}) \cap I \neq \emptyset$, since $\frac{X}{Q} \subseteq X$. Therefore, $\frac{X}{Q}$ is a nonempty unfounded set for $subp(Q)$ w.r.t. I contained in $\frac{I^+}{Q}$.

($\Longleftarrow$) Let Q be a node of $\hat{DG}_{\mathcal{P}}$ such that $\frac{I^+}{Q}$ contains a nonempty unfounded set X for $subp(Q)$ w.r.t. I. X is an unfounded set for $\mathcal{P}$ w.r.t. I as well, since all rules in $ground(\mathcal{P})$ which have an atom from X in the head occur also in $subp(Q)$ (by the definition of $subp(Q)$) and,

```
Function unfounded-free(P:Program; I: SetOfLiterals): Boolean;
var X, Y, J: SetOfLiterals;
    Q: SetOfPredicates;
begin
  for each node Q of DĜ_P
      X := I+/Q;
      repeat      (* Computation of R^ω_{subp(Q),I}(I+/Q) *)
            J := X;
            X := R_{subp(Q),I}(J);
      until J = X;
      if X ≠ ∅
        then if subp(Q) is HCF
                then return False;
                else   (* Computation of non-HCF components *)
                     for each Y ⊆ X do
                          if Y is an unfounded set for subp(Q) w.r.t. I
                            then return False;
                     end_for;
  end_for;
  return True;
end;
```

Figure 3: *Check of the Unfounded-Free Property*

therefore, also satisfy the unfoundedness conditions for $\mathcal{P}$. □

The results of Theorem 6.9 and Propositions 6.5 and 6.11 are all employed by the algorithm for the efficient test of the unfounded-free property which appears in Figure 3.

Informally, the algorithm processes one subprogram $subp(Q)$ at a time, as suggested by Proposition 6.11 (external **for** statement). The fixpoint $X = \mathcal{R}^{\omega}_{subp(Q),I}(\frac{I^+}{Q})$ of $\mathcal{R}_{subp(Q),I}$ is first computed (in the **repeat-until** loop). If it is empty, then, by virtue of Proposition 6.5, $\frac{I^+}{Q}$ contains no unfounded set; thus, the computation of $subp(Q)$ terminates, and the next subprogram is processed. Otherwise ($X \neq \emptyset$), we check whether $subp(Q)$ is HCF, and, if so, the algorithm terminates, returning $False$, as for HCF programs $\mathcal{R}^{\omega}_{subp(Q),I}(\frac{I^+}{Q}) \neq \emptyset$ is sufficient to guarantee that the interpretation is not unfounded-free (from Theorem 6.9). Finally, in the case that both $\mathcal{R}^{\omega}_{subp(Q),I}(\frac{I^+}{Q}) \neq \emptyset$ and $subp(Q)$ is not HCF, all subsets of $\mathcal{R}^{\omega}_{subp(Q),I}(\frac{I^+}{Q})$ are controlled (internal **for** statement) so that we can discover whether one of them is an unfounded set.

Note that the algorithm performs a nonpolynomial computation *only if* the program has a component that is not HCF and for which, further, $\mathcal{R}^{\omega}_{subp(Q),I}(\frac{I^+}{Q}) \neq \emptyset$.

Theorem 6.12 *Given a program $\mathcal{P}$ and a total interpretation I for $\mathcal{P}$, Function* unfounded-free *of Figure 3 terminates in a finite amount of time, returning the correct answer.*

Proof. The two **for** statements are performed on a finite number of elements and thus terminate in a finite amount of time. Moreover, the **repeat-until** statement also terminates finitely, since it computes a monotonically decreasing sequence (lower bounded by the empty set). The function thus ends in a finite amount of time.

The correctness of the function is a consequence of the results of Proposition 6.5, Theorem 6.9, and Proposition 6.11, since the function implements the results of these statements. □

We close this subsection with some remarks on the efficiency of Function *unfounded-free*.

The function solves the decisional problem of checking whether an interpretation is unfounded-free. According to Proposition 3.8, this problem is co-NP-hard. Thus, unless $\mathcal{P}$ = NP, exponential time is needed. Actually, the algorithm runs in single exponential time, as, in the worst case, all subsets of I^+ have to be analyzed to see whether any of them is an unfounded set.

Nevertheless, there is a meaningful class of programs on which the function terminates in polynomial time. In particular, HCF programs belong to this class, which clearly contains all normal logic programs as well (as an $\vee$-free program is HCF). Moreover, the algorithm also runs polynomially on the non-HCF programs for which $\mathcal{R}^{\omega}_{subp(Q),I}(\frac{I^+}{Q}) = \emptyset$ on every non-HCF component. We point out that on programs not in this polynomial class, the inefficient part of the computation is limited to the subprograms that are not HCF. Thus, HCF components are solved in polynomial time, and, furthermore, the test for unfoundedness is done on power sets that in general are of a substantially smaller size than the power set of I^+.

Concerning space bounds, it is easy to see that the function runs in polynomial space. Indeed, even if (in the worst case) the last **for** statement is executed on the power set of I^+, the elements of the power set do not need to be stored, as they can be generated when needed (following a straightforward enumeration policy).

6.2 An Algorithm for Computing Stable Models

We have already seen that $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is contained in every stable model of $\mathcal{P}$ (see Proposition 5.6) and that it is computable in polynomial time (see Proposition 5.12). Thus, in order to compute stable models, we start from the evaluation of $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$. However, once we have computed $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$, two questions come up. How do we check whether $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is a stable model (recall that, by Corollary 5.7, if $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is a stable model, it is the unique stable model)? If $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is not a stable model, how do we (efficiently) move beyond $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ toward the stable models of $\mathcal{P}$?

To this end, we next define the notion of a *possibly-true literal*, which, as it will be clear later in this section, plays a crucial role in our computational model and will allow us to answer both of our questions. A similar notion has previously been used in [35] for the efficient computation of the well-founded model of Datalog programs and in [11] for the evaluation of the stable model semantics of ordered logic programs.

Definition 6.13 Let I be an interpretation for a program $\mathcal{P}$. A *possibly-true conjunction* of $\mathcal{P}$ w.r.t. I is a set of literals of the form $\{a, \neg b_1, ..., \neg b_n\}$, $n \geq 0$, such that there exists a rule $r \in ground(\mathcal{P})$ for which all of the following conditions hold.

1. $a \in H(r)$, and $B^-(r) = \{\neg b_1, ..., \neg b_n\}$ (i.e., a is an atom in the head, and $\neg b_1, ..., \neg b_n$ is the negative part of the body);

2. $H(r) \cap I = \emptyset$ (i.e., the head is not true w.r.t. I);

3. $B^+(r) \subseteq I$ (i.e., every positive literal of the body is true w.r.t. I); and

4. $B^-(r) \cap \neg.I = \emptyset$ (i.e., no negative literal of the body is false w.r.t. I).

We call each literal in a possibly-true conjunction of $\mathcal{P}$ w.r.t. I a *possibly-true literal* of $\mathcal{P}$ w.r.t. I. The set of all possibly-true conjunctions of $\mathcal{P}$ w.r.t. I is denoted by $PT_{\mathcal{P}}(I)$. □

Example 6.14 Consider a program $\mathcal{P}$ consisting of the rule $a \vee b \leftarrow c, \neg d$, and let $I = \{c\}$ be an interpretation for $\mathcal{P}$. There are two possibly-true conjunctions of $\mathcal{P}$ w.r.t. I, namely, $\{a, \neg d\}$ and $\{b, \neg d\}$. □

The next lemma states the relationship between possibly-true literals and models.

Lemma 6.15 *Let I be an interpretation for program $\mathcal{P}$. $I \cup \neg.(B_{\mathcal{P}} - I)$ is a model for $\mathcal{P}$ iff $PT_{\mathcal{P}}(I) = \emptyset$.*

Proof. ($\Longleftarrow$) By Definition 6.13, $PT_{\mathcal{P}}(I) = \emptyset$ implies that, for each $r \in ground(\mathcal{P})$, either (1) $B^+(r) \not\subseteq I$ or (2) $B^-(r) \cap \neg.I \neq \emptyset$ (i.e., some negative literal in the body of r is false w.r.t. I) or (3) $H(r) \cap I \neq \emptyset$. Note that all of these conditions depend only on the positive part I^+ of I. Now, consider the total interpretation $M = I \cup \neg.(B_{\mathcal{P}} - I)$. Since $I^+ = M^+$, we have that, for each $r \in ground(\mathcal{P})$, the three conditions above still hold if we replace I with M. Thus, for each $r \in ground(\mathcal{P})$, either the body of r is false w.r.t. M or its head is true w.r.t. M. Hence, M is a model for $\mathcal{P}$.
($\Longrightarrow$) Let $N = I \cup \neg.(B_{\mathcal{P}} - I)$. Thus, by the definition of the model, for each $r \in ground(\mathcal{P})$, either $B(r)$ is not true w.r.t. N, that is, $B(r) \not\subseteq N$ (in particular, $B(r)$ is false, as N is a total interpretation) or $H(r)$ is true w.r.t. N, that is, $H(r) \cap N \neq \emptyset$. Since $I \subseteq N$, $B(r) \not\subseteq N$ implies $B(r) \not\subseteq I$. Further, since head literals are positive and $N^+ = I^+$, we have that $H(r) \cap N \neq \emptyset$ iff $H(r) \cap I \neq \emptyset$. Thus, each rule $r \in ground(\mathcal{P})$ is such that either its body is not true w.r.t. I or its head is true w.r.t. I. Consider now a rule $r \in ground(\mathcal{P})$ whose head is not true w.r.t. I, and thus $B(r) \not\subseteq I$ holds. Since the head of r is also not true w.r.t. N, $B(r) \not\subseteq N$ is verified. Assume now that $B^+(r) \subseteq I$ and $B^-(r) \not\subseteq I$; hence, $B^-(r) \not\subseteq N$ holds. Now, since N is a total interpretation, $B^-(r) \not\subseteq N$ iff $B^-(r) \cap \neg.N \neq \emptyset$, that is, there exists a (negative) literal in $B^-(r)$, say $\neg a$, such that $a \in N^+$. And since $N^+ = I^+$, it turns out that $B^-(r) \cap \neg.I \neq \emptyset$ holds. We can therefore conclude that, for each rule $r \in ground(\mathcal{P})$, either $B^+(r) \not\subseteq I$ or $B^-(r) \cap \neg.I \neq \emptyset$ or $H(r) \cap I \neq \emptyset$. Hence, $PT_{\mathcal{P}}(I) = \emptyset$ follows immediately from Definition 6.13. □

We are now ready for the following proposition.

Proposition 6.16 *$\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is a stable model for a program $\mathcal{P}$ iff $PT_{\mathcal{P}}(\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset))$ is empty.*

Proof. ($\Longleftarrow$) We show that $\neg.(B_{\mathcal{P}} - \mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)) \subseteq \mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$, that is, $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is a total interpretation for $\mathcal{P}$. Thus, by Theorem 5.4, $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is a stable model for $\mathcal{P}$.
By Definition 6.13, $PT_{\mathcal{P}}(\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)) = \emptyset$ implies that for each rule $r \in ground(\mathcal{P})$, at least one of the following conditions holds: (1) $B^+(r) \not\subseteq \mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ or (2) $B^-(r) \cap \neg.\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset) \neq \emptyset$ or (3) $H(r) \cap \mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset) \neq \emptyset$. Condition 1 implies that $B(r)$ contains some positive literal that is either

false w.r.t. $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ or in $(B_{\mathcal{P}} - \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset))$. Condition 2, in turn, states that some negative literal in the body of r is false w.r.t. $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$. Finally, Condition 3 says that the head of r contains some (positive) literal in $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$. Now, let a be an atom in $(B_{\mathcal{P}} - \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset))$. Thus, for each rule $r \in ground(\mathcal{P})$ with $a \in H(r)$, either $B(r)$ is false w.r.t. $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ or $B(r)$ contains some positive literal in $(B_{\mathcal{P}} - \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset))$ or, finally, there exists a head literal, say b, belonging to $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$. And since a was taken to be an arbitrary element of $(B_{\mathcal{P}} - \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset))$, we may conclude that $(B_{\mathcal{P}} - \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset))$ is an unfounded set of $\mathcal{P}$ w.r.t. $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ (see Definition 3.1). Hence, $(B_{\mathcal{P}} - \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)) \subseteq GUS_{\mathcal{P}}(\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset))$. Now, since $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is a fixpoint of $\mathcal{W}_{\mathcal{P}}$ by hypothesis, we have that $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)^{-} = GUS_{\mathcal{P}}(\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset))$, from which $\neg.(B_{\mathcal{P}} - \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)) \subseteq \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ follows.
($\Longrightarrow$) Since stable models are total interpretations, $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ coincides with $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset) \cup (B_{\mathcal{P}} - \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset))$. Thus, $PT_{\mathcal{P}}(\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)) = \emptyset$ follows immediately from Lemma 6.15. □

Corollary 6.17 *Let $\mathcal{P}$ be a program. If $PT_{\mathcal{P}}(\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)) = \emptyset$, then $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is the unique stable model of $\mathcal{P}$.*

Proof. Follows from Proposition 6.16 and Corollary 5.7. □

Thus, if $PT_{\mathcal{P}}(\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)) = \emptyset$, we are done, as $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is the unique stable model of $\mathcal{P}$. Otherwise, we have to move forward, as $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset) \subseteq M$ for each stable model M of $\mathcal{P}$. To see how, we provide the following proposition, which is fundamental for our computational strategy.

Proposition 6.18 *Let I be an interpretation for a program $\mathcal{P}$ and M a stable model for $\mathcal{P}$. If $I \subset M$ and $I^{+} \subset M^{+}$, then there exists a possibly-true conjunction $X \in PT_{\mathcal{P}}(I)$ such that $I \subset I \cup X \subseteq M$.*

Proof. Since $I \subset M$ by hypothesis, and since every possibly-true conjunction $X = \{a, \neg b_1, ..., \neg b_n\}$ is not contained in I (as a is not in I, from Condition 2 of Definition 6.13), it is sufficient to prove that there exists a possibly-true conjunction $X \in PT_{\mathcal{P}}(I)$ such that $X \subseteq M$.

Let $Z = M^{+} - I^{+}$. Since M is a stable model and thus, by Theorem 4.6, unfounded-free, Z is not an unfounded set of $\mathcal{P}$ w.r.t. M. Thus, by Definition 3.1, there exists a (positive) literal $a \in Z$ for which there is a rule $r \in ground(\mathcal{P})$ with $a \in H(r)$, such that (1) $B(r)$ is not false w.r.t. M, (2) $B^{+}(r) \cap Z = \emptyset$, and (3) $(H(r) - Z) \cap M = \emptyset$. Now, we have the following:

- Since M is a total interpretation, $B(r)$ not false w.r.t. M (Condition 1 above) implies that $B(r)$ is true w.r.t. M, that is, $B(r) \subseteq M$.
- From $B(r) \subseteq M$ and $B^{+}(r) \cap Z = \emptyset$ (Condition 2 above), it can easily be recognized that $B^{+}(r) \subseteq I$ follows.
- $B(r) \subseteq M$ and the hypothesis $I \subset M$ imply that $B(r)$ is not false in I. In particular, no negative literal in the body of r is false w.r.t. I, that is, $B^{-}(r) \cap \neg.I = \emptyset$.
- Finally, $(H(r) - Z) \cap M = \emptyset$ (condition 3 above) iff $H(r) \cap (M - Z) = \emptyset$ (by set theory). Since $H(r)$ consists of only positive literals, it turns out that $H(r) \cap (M^{+} - Z) = \emptyset$, from which $H(r) \cap I = \emptyset$ immediately follows.

In other words, Conditions 2 through 4 of Definition 6.13, namely, $B^{+}(r) \subseteq I$, $B^{-}(r) \cap \neg.I = \emptyset$, and $H(r) \cap I = \emptyset$, are all satisfied. Therefore, we may conclude that $X = \{a, \neg b_1, ..., \neg b_n\}$, where $\neg b_1, ..., \neg b_n$ are the negative literals in $B(r)$, is a possibly-true conjunction of $\mathcal{P}$ w.r.t. I. Since $a \in M$, $\{\neg b_1, ..., \neg b_n\} \subseteq B(r)$, and $B(r) \subseteq M$, it turns out that $X \subseteq M$, and hence we have proved the proposition. □

Corollary 6.19 *If $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is not a stable model, then, for each stable model M of $\mathcal{P}$, there exists a possibly-true conjunction $X \in PT_{\mathcal{P}}(\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset))$ such that $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset) \subset \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset) \cup X \subseteq M$.*

Proof. Since $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ is not a stable model, by virtue of Proposition 5.6 for each stable model M of $\mathcal{P}$, it holds that $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset) \subset M$. The statement thus follows from Proposition 6.18. □

The above corollary suggests a simple way of moving in the direction of stable models starting from $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$. Indeed, for each stable model M, there exists a possibly-true conjunction $X \in PT_{\mathcal{P}}(\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset))$ such that $X \subseteq M$. Now, let M be a fixed stable model. Obviously, $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset) \cup X$ is also contained in M. Thus, by iteratively applying the inflationary $\mathcal{T}_{\mathcal{P}}$ operator $\overline{\mathcal{T}}_{\mathcal{P}}$ (see below), starting from $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset) \cup X$, a fixpoint, say $\hat{I}$ of $\overline{\mathcal{T}}_{\mathcal{P}}$, contained in M is finitely reached (note that $\mathcal{T}_{\mathcal{P}}$ is monotonic and $B_{\mathcal{P}}$ is finite). Indeed, for a generic interpretation J, $J \subseteq M$ implies $\mathcal{T}_{\mathcal{P}}(J) \subseteq M$ as an immediate corollary of Proposition 5.5. Now, if $PT_{\mathcal{P}}(\hat{I}) = \emptyset$ then $\hat{I} \cup \neg.(B_{\mathcal{P}} - \hat{I})$ is a model by Lemma 6.15 and, since it is contained in M, $\hat{I} \cup \neg.(B_{\mathcal{P}} - \hat{I}) = M$ follows by minimality of stable models. Otherwise, a possibly-true conjunction $X' \in PT_{\mathcal{P}}(\hat{I})$ contained in M is chosen and added to $\hat{I}$, and the computation restarts from $\hat{I} \cup X' \subseteq M$ by using $\overline{\mathcal{T}}_{\mathcal{P}}$. The process is reiterated until a fixpoint of $\overline{\mathcal{T}}_{\mathcal{P}}$, say J, such that $PT_{\mathcal{P}}(J) = \emptyset$, is found. Again, by Lemma 6.15, $J \cup \neg.(B_{\mathcal{P}} - J)$ is a model and, in particular, $J \cup \neg.(B_{\mathcal{P}} - J) = M$.

We note that this computation is nondeterministic, as we have assumed the ability to guess each time a "right" possibly-true conjunction contained in the stable model M, thus generating a sequence of interpretations contained in M. Unfortunately, this model of computation is unrealistic; an exhaustive search, in which all possibly-true conjunctions are tried, is actually required. As a consequence, in a real computation, whenever an interpretation J such that $J = \overline{\mathcal{T}}_{\mathcal{P}}(J)$ and $PT_{\mathcal{P}}(J) = \emptyset$ is reached, a check on whether $J \cup \neg.(B_{\mathcal{P}} - J)$ is unfounded-free is needed to recognize whether it is a stable model (recall that by Lemma 6.15 it is a model); this check somehow verifies that "right" possibly-true conjunctions have been chosen.

We begin formalizing these observations by defining the notion of a *computation of* $\mathcal{P}$.

Definition 6.20 Let $\mathcal{P}$ be a program. A sequence of sets of ground literals $\{\mathcal{V}_n\}_{n \in \mathcal{N}}$ is a *computation of* $\mathcal{P}$ if the following hold:

- $V_0 = \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$
- If $\overline{\mathcal{T}}_{\mathcal{P}}(V_n) \neq V_n$ then $V_{n+1} = \overline{\mathcal{T}}_{\mathcal{P}}(V_n)$, otherwise
  - If $PT_{\mathcal{P}}(V_n) \neq \emptyset$ then $V_{n+1} = X \cup V_n$, for some $X \in PT_{\mathcal{P}}(V_n)$
  - else $V_{n+1} = V_n$;

  where $\overline{\mathcal{T}}_{\mathcal{P}}(V_n) = \mathcal{T}_{\mathcal{P}}(V_n) \cup V_n$

□

Clearly, there are finitely many computations of $\mathcal{P}$ ($2^{|B_{\mathcal{P}}|}$ is an upper bound, where $|B_{\mathcal{P}}|$ is the (finite) size of the Herbrand base of $\mathcal{P}$). We denote by $\mathcal{F}_{\mathcal{P}}$ the family of all the computations of $\mathcal{P}$. Let $\{\mathcal{V}_n\}_{n \in \mathcal{N}}$ be an element of $\mathcal{F}_{\mathcal{P}}$. It is easily recognized that $\{\mathcal{V}_n\}_{n \in \mathcal{N}}$ is a monotonically increasing sequence. Since the base of $\mathcal{P}$ is finite, $\{\mathcal{V}_n\}_{n \in \mathcal{N}}$ is upper bounded, so that there exists a natural k such that $V_j = V_k$, for each $j \geq k$. We denote the limit V_k of $\{\mathcal{V}_n\}_{n \in \mathcal{N}}$ by V^{ω}.

Informally, the family $\mathcal{F}_{\mathcal{P}}$ can be represented by a rooted tree, where the root represents $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ and each path P from the root to a leaf represents a computation $\{\mathcal{V}_n\}_{n\in\mathcal{N}}$ of $\mathcal{P}$, the leaf being the limit V^{ω}. Each intermediate node of P represents a term $V_n \in \{\mathcal{V}_n\}_{n\in\mathcal{N}}$. We call such a tree the *choice tree of* $\mathcal{P}$, denoted $\mathcal{CT}_{\mathcal{P}}$, and we call each node of P representing a term $V_n \in \{\mathcal{V}_n\}_{n\in\mathcal{N}}$ such that $V_n = \overline{\mathcal{T}}_{\mathcal{P}}(V_n)$ a *choice node* of $\mathcal{CT}_{\mathcal{P}}$. Apart from the leaves, the choice nodes are the points where the possibly-true conjunctions have to be chosen. Clearly, the root of $\mathcal{CT}_{\mathcal{P}}$ is a choice node. Note that each choice node representing a term V_n has as many children as the size of the set $PT_{\mathcal{P}}(V_n)$. Any other node of $\mathcal{CT}_{\mathcal{P}}$, except the leaves, has exactly one child.

Next we show the relationships between stable models of $\mathcal{P}$ and the leaves of the choice tree of $\mathcal{P}$ (i.e., the limits V^{ω} of the computations of $\mathcal{P}$). We provide a one-to-one correspondence between the set of stable models of $\mathcal{P}$ and a subset of the leaves of the choice tree. Thus, the choice tree of $\mathcal{P}$ both reduces the number of interpretations that are "candidates" to be stable models, and provides us with a constructive way of computing the stable models.

In more detail, we show that the stable models exactly coincide with the interpretations $V^{\omega} \cup \neg.(B_{\mathcal{P}} - V^{\omega})$ that are unfounded-free (where, again, V^{ω} denotes the limit of a computation). To this end, we first give the following result.

Lemma 6.21 *Let $\{\mathcal{V}_n\}_{n\in\mathcal{N}}$ be a computation of $\mathcal{P}$ and V_n an element of $\{\mathcal{V}_n\}_{n\in\mathcal{N}}$. $V_n = V^{\omega}$ iff $PT_{\mathcal{P}}(V_n) = \emptyset$.*

Proof. By Definition 6.20, we have that $V_{n+1} = V_n$ iff $V_n = \mathcal{T}_{\mathcal{P}}(V_n) \cup V_n$ and $PT_{\mathcal{P}}(V_n) = \emptyset$. Thus, $V_{n+1} = V_n$ implies $PT_{\mathcal{P}}(V_n) = \emptyset$, which proves the ($\Longrightarrow$) part of the lemma. Concerning the ($\Longleftarrow$) part, it suffices to show that $PT_{\mathcal{P}}(V_n) = \emptyset$ implies $V_n = \mathcal{T}_{\mathcal{P}}(V_n) \cup V_n$. Let a be an atom occurring in $\mathcal{T}_{\mathcal{P}}(V_n)$. By definition, $a \in \mathcal{T}_{\mathcal{P}}(V_n)$ iff there exists a ground rule $r \in ground(\mathcal{P})$ such that $a \in H(r)$, $H(r) - \{a\} \subseteq \neg.V_n$ (i.e., all other head literals are false) and $B(r) \subseteq V_n$. Now, if $a \notin V_n$, it immediately follows from Definition 6.13 that $X = \{a, \neg b_1, ..., \neg b_n\}$, where $\neg b_1, ..., \neg b_n$ are the negative literals in the body of r, is a possibly-true conjunction of $\mathcal{P}$ w.r.t. V_n. That is, $a \in \mathcal{T}_{\mathcal{P}}(V_n)$ and $a \notin V_n$ imply the existence of X. Hence, $PT_{\mathcal{P}}(V_n) = \emptyset$ implies that any atom $a \in \mathcal{T}_{\mathcal{P}}(V_n)$ belongs to V_n, that is, $\mathcal{T}_{\mathcal{P}}(V_n) \subseteq V_n$. From this, $V_n = \mathcal{T}_{\mathcal{P}}(V_n) \cup V_n$ follows. □

Theorem 6.22 *Let $\{\mathcal{V}_n\}_{n\in\mathcal{N}}$ be a computation of $\mathcal{P}$. If $V^{\omega} \cup \neg.(B_{\mathcal{P}} - V^{\omega})$ is an unfounded-free interpretation then it is a stable model of $\mathcal{P}$.*

Proof. By Lemma 6.21, we have that $PT_{\mathcal{P}}(V^{\omega}) = \emptyset$ and thus, by Lemma 6.15, $V^{\omega} \cup \neg.(B_{\mathcal{P}} - V^{\omega})$ is a model for $\mathcal{P}$. Since it is unfounded-free, by Theorem 4.6, $V^{\omega} \cup \neg.(B_{\mathcal{P}} - V^{\omega})$ is a stable model for $\mathcal{P}$. □

Next we prove the converse of Theorem 6.22, which will prove the completeness of our computational strategy.

Theorem 6.23 *Let M be a stable model for a program $\mathcal{P}$. $M = V^{\omega} \cup \neg.(B_{\mathcal{P}} - V^{\omega})$, for some computation $\{\mathcal{V}_n\}_{n\in\mathcal{N}}$ of $\mathcal{P}$.*

Proof. We give a constructive proof of the existence of a computation $\{\mathcal{V}_n\}_{n\in\mathcal{N}}$ whose elements are contained in M. Let $\{\mathcal{V}_n\}_{n\in\mathcal{N}}$ be the sequence of interpretations such that $V_0 = \mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ and V_{n+1} is as follows (see Definition 6.20):

1. If $\overline{\mathcal{T}}_{\mathcal{P}}(V_n) \neq V_n$, then $V_{n+1} = \overline{\mathcal{T}}_{\mathcal{P}}(V_n)$; otherwise,

   2 If $PT_{\mathcal{P}}(V_n) \neq \emptyset$, then $V_{n+1} = X \cup V_n$, where $X \in PT_{\mathcal{P}}(V_n) \cap M$,

   3 Else $V_{n+1} = V_n$.

We next show that, for each $n \in \mathcal{N}$, V_n is well defined and $V_n \subseteq M$. We proceed by induction.
(Base of the induction.) V_0 is clearly well defined, and its inclusion in M stems from Proposition 5.6, as $V_0 = \mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$.
Assuming that $V_n \subseteq M$ (*inductive hypothesis*), it follows that V_{n+1} is well defined. Indeed, be ill defined V_{n+1} could only through Point 2 above, and Proposition 6.18 ensures the existence of X belonging to $PT_{\mathcal{P}}(V_n) \cap M$. [10] Moreover, it is easy to see that $V_{n+1} \subseteq M$. Indeed,

- If $V_{n+1} = \mathcal{T}_{\mathcal{P}}(V_n) \cup V_n$, then $V_{n+1} \subseteq M$, since $V_n \subseteq M$ by inductive hypothesis, and $\mathcal{T}_{\mathcal{P}}(V_n) \subseteq M$ holds as an immediate corollary of Proposition 5.5.
- If $V_{n+1} = X \cup V_n$, where $X \in PT_{\mathcal{P}}(V_n) \cap M$, it is then immediate that $V_{n+1} \subseteq M$, as both X and V_n are subsets of M.
- If $V_{n+1} = V_n$, then $V_{n+1} \subseteq M$ for inductive hypothesis.

Now, it is easy to see that $\{\mathcal{V}_n\}_{n \in \mathcal{N}}$ is a computation. Let V^{ω} be the limit of $\{\mathcal{V}_n\}_{n \in \mathcal{N}}$ (clearly $V^{\omega} \subseteq M$). By Lemma 6.21, $PT_{\mathcal{P}}(V^{\omega}) = \emptyset$, and hence by Lemma 6.15, $V^{\omega} \cup \neg(B_{\mathcal{P}} - V^{\omega})$ is a model. From the minimality of M (see Corollary 4.7), $V^{\omega} \cup \neg.(B_{\mathcal{P}} - V^{\omega}) = M$ follows. □

Figure 4 shows an algorithm for the computation of all stable models of a disjunctive Datalog program $\mathcal{P}$. First, the least fixpoint $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is computed by the **repeat-until** statement of the *main* program. Then, according to Corollary 6.17, the condition $PT_{\mathcal{P}}(\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)) = \emptyset$ is tested to verify whether $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is the unique stable model of $\mathcal{P}$. If so, $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is given as the output of the algorithm. Otherwise, the procedure *Compute_Stable* is invoked to generate the family $\mathcal{F}_{\mathcal{P}}$ of the computations of $\mathcal{P}$. This procedure, which has been written in recursive form for clarity, is based on a backtracking technique, and its structure is that of a preorder visit of the choice tree $\mathcal{CT}_{\mathcal{P}}$ of $\mathcal{P}$. Initially, *Compute_Stable* is run with actual parameter $V_n = \mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$, which corresponds to a visit to the root of $\mathcal{CT}_{\mathcal{P}}$. In general, the procedure is invoked with the actual parameter corresponding to a choice node of $\mathcal{CT}_{\mathcal{P}}$ (recall that the root of $\mathcal{CT}_{\mathcal{P}}$ is a choice node). Then, for each possibly-true conjunction $X \in PT_{\mathcal{P}}(V_n)$ (possibly none), X is added to V_n and the set of ground literals V'_{n+1}, corresponding to the next choice node of $\mathcal{CT}_{\mathcal{P}}$, is computed by the **repeat-until** statement. The procedure then recursively invokes itself with actual parameter V'_{n+1}. Whenever the condition $PT_{\mathcal{P}}(V_n) = \emptyset$ is verified, the function of Figure 3 is invoked to check whether $V_n \cup \neg.(B_{\mathcal{P}} - V_n)$ is unfounded free. Note that a possible inconsistency in V'_{n+1} is detected by the algorithm (see the second **until** condition and the subsequent **if** statement), hence pruning the choice tree of $\mathcal{P}$ by discarding computations whose elements are not interpretations. It is easy to see that the latter optimization is correct: if V'_{n+1} is inconsistent, then V'_{n+m} is inconsistent as well, for any $m \geq 1$ (since the computation is monotonic).

Example 6.24 Consider the program $\mathcal{P}$ consisting of the following rules:

$d \leftarrow$	$e \leftarrow d, \neg f$	$a \vee c \leftarrow \neg b$
$b \leftarrow \neg a$	$b \leftarrow c$	$g \leftarrow a$

[10] Actually, more than one X of this form may exist, but the choice is immaterial.

Input: A disjunctive logic program $\mathcal{P}$.
Output: The stable models of $\mathcal{P}$.

Function *unfounded-free(P: Program; I: SetOfLiterals): Boolean; (* see Figure 3 *)*

Procedure $Compute\_Stable(V_n : SetOfLiterals)$;
var X, V'_n, V'_{n+1}: SetOfLiterals;
  if $PT_{\mathcal{P}}(V_n) = \emptyset$ *(* V_n is the limit of a computation *)*
    then if *unfounded-free*($\mathcal{P}$; $V_n \cup \neg.(B_{\mathcal{P}} - V_n)$)
        then output "$V_n \cup \neg.(B_{\mathcal{P}} - V_n)$ *is a stable model of* $\mathcal{P}$";
      end_if
    else for each $X \in PT_{\mathcal{P}}(V_n)$ **do**
        $V'_{n+1} := V_n \cup X$; *(* Assume the truth of a possibly-true conjunction *)*
        repeat
          $V'_n := V'_{n+1}$;
          $V'_{n+1} := \overline{\mathcal{T}}_{\mathcal{P}}(V'_n)$;
        until $V'_{n+1} = V'_n$ or $V'_{n+1} \cap \neg.V'_{n+1} \neq \emptyset$;
        if $V'_{n+1} \cap \neg.V'_{n+1} = \emptyset$ *(* V'_{n+1} is consistent *)*
          then $Compute\_Stable(V'_{n+1})$
      end_for
  end_if
end_procedure

var I, J: SetOfLiterals;
begin *(* Main *)*
  $I := \emptyset$;
  repeat *(* Computation of $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ *)*
      $J := I$;
      $I := \mathcal{W}_{\mathcal{P}}(I)$;
  until $I = J$;
  if $PT_{\mathcal{P}}(I) = \emptyset$
    then output "*I is the unique stable model of* $\mathcal{P}$";
    else $Compute\_Stable(I)$;
end

Figure 4: Algorithm for the Computation of Stable Models

Clearly, $\mathcal{P}$ has two stable models, namely, $\{a, \neg b, \neg c, d, e, \neg f, g\}$ and $\{\neg a, b, \neg c, d, e, \neg f, \neg g\}$.

The algorithm of Figure 4 starts by computing $V_0 = \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset) = \{d, e, \neg f\}$. Now, the possibly-true conjunctions of $\mathcal{P}$ w.r.t. $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$ are $X_1 = \{a, \neg b\}$, $X_2 = \{c, \neg b\}$ and $X_3 = \{b, \neg a\}$. One of them, say X_1, is chosen, and the sequence

$$\begin{array}{l} V_1 = \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset) \cup X_1 = \{a, \neg b, d, e, \neg f\} \\ V_2 = \mathcal{T}_{\mathcal{P}}(V_1) \cup V_1 = \{a, \neg b, d, e, \neg f, g\} \\ V_3 = \mathcal{T}_{\mathcal{P}}(V_2) \cup V_2 = V_2 \end{array}$$

computed. Since $PT_{\mathcal{P}}(V_3) = \emptyset$ and $V_3 \cup \neg.(B_{\mathcal{P}} - V_3)$ is an interpretation (i.e., it is consistent) and unfounded-free (as no subset of its positive part V_3^+ is an unfounded set), the algorithm outputs the stable model $V_3 \cup \neg.(B_{\mathcal{P}} - V_3) = \{a, \neg b, \neg c, d, e, \neg f, g\}$.

Next, another possibly-true conjunction, say $X_2 = \{c, \neg b\}$, is chosen and the sequence

$$\begin{array}{l} V_1 = \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset) \cup X_2 = \{\neg b, c, d, e, \neg f\} \\ V_2 = \mathcal{T}_{\mathcal{P}}(V_1) \cup V_1 = \{\neg b, b, c, d, e, \neg f\} \end{array}$$

computed. Since V_2 is not an interpretation (as both b and $\neg b$ are in V_2), this computation is stopped here (no output).

Finally, the possibly-true conjunction $X_3 = \{b, \neg a\}$ is chosen and the sequence

$$\begin{array}{l} V_1 = \mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset) \cup X_3 = \{\neg a, b, d, e, \neg f\} \\ V_2 = \mathcal{T}_{\mathcal{P}}(V_1) \cup V_1 = V_1 \end{array}$$

computed. Since $PT_{\mathcal{P}}(V_2) = \emptyset$ and, further, $V_2 \cup \neg.(B_{\mathcal{P}} - V_2)$ is an unfounded-free interpretation, the algorithm outputs the stable model $V_2 \cup \neg.(B_{\mathcal{P}} - V_2) = \{\neg a, b, \neg c, d, e, \neg f, \neg g\}$. $\square$

It is worth noting that all operators utilized in the algorithm of Figure 4 can be implemented easily (and efficiently). Indeed, the implementations of $\overline{\mathcal{T}}_{\mathcal{P}}$ and $PT_{\mathcal{P}}$ are immediate. Moreover, since $\mathcal{W}_{\mathcal{P}}$ is applied only on unfounded-free interpretations, its negative part $\neg.GUS_{\mathcal{P}}(I)$ can be computed efficiently as it coincides with $B_{\mathcal{P}} - \phi_{\lambda}$ (see the proof of Point b of Proposition 3.7), where ϕ_{λ} is the limit of a monotonic computation. Furthermore, observe that all the operators are definable in relational algebra and that most operations performed in the algorithm are set-oriented (the only exception is the **for** statement that picks up the possibly-true conjunctions); the method is thus amenable for deductive databases.

Theorem 6.25 *Given a program $\mathcal{P}$, the algorithm of Figure 4 terminates in a finite amount of time and returns the stable models of $\mathcal{P}$.*

Proof. The **repeat-until** statement of the *main* section in Figure 4 computes the least fixpoint $\mathcal{W}_{\mathcal{P}}^{\omega}(\emptyset)$. Since $\mathcal{W}_{\mathcal{P}}$ is monotonic and $B_{\mathcal{P}}$ is finite, this computation halts after a finite number of steps. Likewise, the **repeat-until** of the procedure *Compute_Stable* computes a sequence inductively defined as $I_0 = V_n \cup X$, $I_{n+1} = \overline{\mathcal{T}}_{\mathcal{P}}(I_n)$. Since $\overline{\mathcal{T}}_{\mathcal{P}}$ is monotonic (recall that $\overline{\mathcal{T}}_{\mathcal{P}}(I) = \mathcal{T}_{\mathcal{P}}(I) \cup I$) and $B_{\mathcal{P}}$ finite, this loop also halts finitely. Therefore, the algorithm of Figure 4 terminates in a finite amount of time.

Concerning correctness, we will just observe that the algorithm essentially generates the family $\mathcal{F}_{\mathcal{P}}$ of the computations of $\mathcal{P}$ (except those that contain terms that are not interpretations of

$\mathcal{P}$, for optimization reasons). Thus, Theorems 6.22 and 6.23 guarantee both the soundness and the completeness of the algorithm. $\Box$

Remarks

- We emphasize that our algorithm does not entail computation of all the negative literals of stable models. The models are "total" interpretations, and thus it is sufficient to compute explicitly only those negative literals that are necessary to derive the positive part of the model. In our approach, such literals are supplied by possibly-true conjunctions.
- We could have accomplished the task of computing stable models using $\mathcal{W}_{\mathcal{P}}$ instead of $\overline{\mathcal{T}}_{\mathcal{P}}$ to move beyond $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ (recall that stable models are fixpoints of $\mathcal{W}_{\mathcal{P}}$). However, using $\mathcal{W}_{\mathcal{P}}$ presents a drawback, as $\mathcal{W}_{\mathcal{P}}$ is defined on an interpretation for which the $GUS_{\mathcal{P}}$ exists and testing such a property is a computationally demanding task (see Section 3).
- As particular cases, the algorithm also computes the minimal models of positive disjunctive logic programs, the perfect models of stratified (disjunctive) logic programs, and the stable models of normal logic programs. $\Box$

Another point worth discussing is complexity. Essentially, the algorithm generates all computations of $\mathcal{P}$ (except those whose elements are not interpretations of $\mathcal{P}$), that is, it performs a controlled search in the choice tree of $\mathcal{P}$. The limit of a computation (i.e., a leaf of the choice tree) is computed in polynomial time, as the number of elements preceding the limit (i.e., the length of a path of the tree) is linear in $|B_{\mathcal{P}}|$, and the computation of each element requires evaluation of $\overline{\mathcal{T}}_{\mathcal{P}}$ and $PT_{\mathcal{P}}$, which are computable in polynomial time. The check of whether a leaf of the choice tree is unfounded-free is performed by a call to Function *unfounded-free*. This function executes in at most single exponential time and polynomial space (see Section 6.1). Therefore, each computation is carried out in single exponential time and polynomial space.

Since the number of computations (i.e., the number of leaves of the choice tree) is single exponential, the whole execution of the algorithm is done in single exponential time in the worst case. Moreover, as the algorithm deals with one computation at a time, it runs in polynomial space.

For some restricted classes of programs, including disjunction-free programs that are *locally stratified* [49], *weakly stratified* [48], or *modularly stratified* [55], the problem of finding the set of stable models is known to be solvable in polynomial time. Our algorithm runs in polynomial time for each program $\mathcal{P}$ such that $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is a total interpretation. Indeed, $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is computed in polynomial time and, when $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is total, the algorithm successfully terminates, returning the unique stable model of the program, as $PT_{\mathcal{P}}(\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)) = \emptyset$. Since for disjunction-free programs $\mathcal{W}^{\omega}_{\mathcal{P}}(\emptyset)$ is the well-founded model, the algorithm terminates in polynomial time on every normal logic program having a total well-founded model. It follows that the algorithm also evaluates in polynomial time locally stratified, weakly stratified, and modularly stratified normal logic programs.

7 Related Work

The notion of unfounded sets presented in this paper generalizes from normal to disjunctive programs the corresponding notion defined by Van Gelder, Ross and Schlipf in [64] (as proven in Proposition 3.3). In [21], to characterize the 3-valued stable models of Przymusinski [50] in terms of unfounded sets, Eiter et al. provide a definition of unfounded sets for disjunctive logic programs. In that definition, Condition 3 of Definition 3.1 is replaced by the weaker requirement $H(r) \not\subseteq (\neg.I \cup X)$, because $(H(r) - X) \cap I^+ \neq \emptyset$ turns out to be too strong to capture unfoundedness appropriately with respect to the 3-valued stable model semantics of [50]. Other notions of unfounded sets are somehow implicit in various attempts to generalize the well-founded semantics from normal to disjunctive logic programs (see, e.g., [2, 37, 52, 51, 56]). However, comparing these (implicit) notions of unfounded sets to our notion is quite difficult because in these other notions unfounded sets do not result in flat sets of atoms as does Definition 3.1 (and the standard definition of unfounded sets for normal programs [64]).

To our knowledge, this paper is the first to present a precise characterization of disjunctive stable models in terms of unfounded sets. Our basic result, stating that disjunctive stable models precisely coincide with the unfounded-free models (Theorem 4.6), generalizes to the disjunctive case the analogous result given for normal logic programs in [57]. The second characterization, stating that a model is stable iff the set of its false atoms coincides with its greatest unfounded set (Theorem 4.8), is related to the assumption-based framework of Bondarenko, Toni, and Kowalski [9]. In their framework, our greatest unfounded set can be seen as a notion of "acceptability" for a given set of assumptions.

The fixpoint semantics presented in Section 5 has much in common with the fixpoint results of [64]. Indeed, as shown in Proposition 5.3, on the domain of normal logic programs, our $\mathcal{W}_{\mathcal{P}}$ operator coincides with Van Gelder et al.'s well-founded operator. Therefore, the results in Section 5 generalize the analogous results presented by Van Gelder et al. for traditional programs. In particular, our Theorem 5.4, Corollary 5.7, and Proposition 5.6 generalize, respectively, Theorem 5.4, Corollary 5.6, and Corollary 5.7 of [64].

Inoue and Sakama [34] and Fitting [27] have also developed fixpoint characterizations of stable models. Inoue and Sakama's work considers disjunction and negation but does not combine the two constructs, as our does. Fitting's characterization, an extension of his previous work on well-founded semantics [27], presents the results in the general setting of bilattices rather than confining things to the framework of conventional logic programming. Fitting's characterization has not been extended to the disjunctive case, and an investigation of whether our work could be used to do so would be interesting.

A fixpoint semantics for disjunctive logic programs appears in [24, 26]. However, because this work characterizes the perfect model semantics of Przymusinski [49], it therefore corresponds to ours only on the class of stratified disjunctive logic programs.

On the complexity side, our results complement the deep complexity analysis done by Eiter et al. [17, 18, 19], where the complexity of the main computational problems arising in the context of disjunctive logic programming is determined.

On the work on the computational aspects of disjunctive logic programs, the algorithms based on a bottom-up computational model [25, 26, 10, 45, 60, 62], seem closest to our work. The algorithm presented in [26] exploits the fixpoint characterizations of [24] to evaluate stratified programs. It uses the *model-tree* data structure to represent information and to compute query answers. Informally, a *model-tree* encodes a finite family of interpretations, where each branch of the tree represents an interpretation. Every operation required for the fixpoint computation can easily be performed on the model trees. A similar approach, called *state generation*, is described in [45] for the computation of positive logic programs. This method is based on *hyperresolution* and utilizes an operator applied to disjunctive Herbrand *states* (i.e., disjunctive facts) whose least fixpoint is the set of logical consequences (*minimal model state*) of the program. The relationships between these two methods (namely, [26] and [45]) is analyzed in [60], where the authors also point out that generating models for stable semantics can be achieved through the iterative version of the fixpoint operator from [26] by using the *evidential transformation* [23] and the *3-S transformation* [61]. Another algorithm for computing stable models which uses a bottom-up strategy is presented by Brass and Dix in [10]. Their algorithm first computes the "residual program" – a program where no positive literals appear in the rules' bodies – which is equivalent to the original program under stable model semantics. Stable models are then computed on (a simple extension of) Clark's completion of the residual program. A different approach, which is based on integer programming methods, is proposed in [6]. A disjunctive logic program is translated into an integer programming problem where constraints correspond to the program's clauses. Well-known techniques borrowed from the linear programming domain are then employed. The stable models of the disjunctive logic program are immediately obtained from the solutions of the linear programming problem. Also Dix and Müller have tried implementation of semantics of disjunctive logic programs based on abstract properties [16], but their procedure applies only to stratified logic programs.

A drawback of all the algorithms cited above [25, 26, 10, 45, 60, 16] is that they require exponential space in the worst case. Thus, when compared to these methods, our algorithm has the important advantage of requiring only polynomial space. (Note that polynomial space complexity is a fundamental requirement for nonmonotonic systems and deductive database systems [46].)

A bottom-up procedure running in polynomial space has been designed by Stüber [62]. In analogy to the Davis-Putnam procedure [14], this procedure computes disjunctive stable models using case analysis and simplification. A peculiarity of this method is that it avoids the repetition of the generated (stable) models. Compared to Stüber's method, our algorithm has two main advantages: our method is better suited than Stüber's method for deductive databases (as [62] requires the instantiation of the program, which is not feasible in the context of deductive databases working with large amount of data); and Stüber's algorithm may require exponential time for checking whether a model is stable even if the program is HCF, while our procedure for checking stability is always polynomial on these programs.

A further group of work on the computation of disjunctive logic programs concerns the implementation of extensions of Prolog to cope with disjunctive rules. In [53], Reed et al. provide a characterization of disjunctive logic programs which agrees with the declarative semantics of Minker [44] (on positive programs) and provides a fixpoint semantics for Inheritance near-Horn Prolog (InH-Prolog) [38] – a proof procedure extending Prolog with case-analysis. Another strategy, applicable to *range-restricted* positive programs, is incorporated in the prover SATCHMO developed by Manthey and Bry [40]. SATCHMO is a refutation system that uses Prolog to process the Horn clauses, while acts as a forward-chaining prover, simulating hyperresolution, on the non-Horn rules. [39] proposes an extension of SATCHMO, called SATCHMORE (standing for SATCHMO with RElevancy), where the set of clauses to be used for forward chaining is significantly reduced by marking relevant literals.

In contrast to our algorithm, the above methods [38, 53, 40, 39] are (completely or partially) based on a top-down computational model. Thus, our algorithm has the well-known advantages and drawbacks of bottom-up evaluation strategies, while these methods have the advantages and drawbacks of top-down strategies. However, it is worth noting that our method is more powerful than both SATCHMO and InH-Prolog in that it can be employed to perform the more general forms of reasoning on disjunctive logic programs. For instance, both brave and cautious reasoning, whose complexity (under minimal or stable model semantics) is, respectively, Σ_2^P and Π_2^P, can be performed by using our algorithm. SATCHMO and InH-Prolog cannot perform these forms of reasoning because they are based on "flat" backtracking procedures which cannot solve decisional problems located at the second level of the polynomial hierarchy [18, 19, 17]. These systems can perform cautious reasoning only for (disjunction of) *positive* literals (whose complexity is co-NP).

Since our algorithm can be employed for the computation of the stable model semantics of normal logic programs (as a particular case), work on this problem is related to our method. Among the available results in this area are the branch-and-bound method of Subrahmanian et al. [63], the Saccá-Zaniolo backtracking technique [57], the linear programming methods of Bell et al. [4, 5, 6], the strategy proposed by Cuadrado and Pimentel [13], the technique by Niemelä and Simons [46], and the algorithm by Fuentes [29].

Finally, the method employed by our algorithm to check the stability condition adds new insights to the work of Ben-Eliyahu and Dechter [7] and Ben-Eliyahu and Palopoli [8] on HCF disjunctive logic programs.

8 Conclusion

We have proposed a new notion of unfounded sets for disjunctive logic programs which allows us to provide both declarative and fixpoint characterizations of disjunctive stable models. Our characterizations point out some basic properties of stable models and shed light on their intrinsic nature.

The fixpoint semantics provides the ground for the design of an algorithm for the computation of the stable model semantics. Indeed, by exploiting the above theoretical results, we have written an algorithm for the computation of the stable models of disjunctive deductive databases.

Currently, we are exploring the implementation of the proposed algorithm. It is indeed very important to evaluate the advantages of our method against other techniques by a meaningful and thorough experimentation. We are pursing this work at the Technical University of Vienna within *FWF Project P11580-MAT: "A Query System for Disjunctive Deductive Databases,"* where a disjunctive deductive database system based on the techniques presented here is actually implemented and tested. The testing of the system will follow the methodology for experimenting with nonmonotonic reasoning systems developed by Cholewiński et al. [12].

Another question that we are investigating actively is whether, besides elegantly characterizing the stable model semantics, our new definition of unfounded sets permits definition of a suitable extension of the well-founded semantics to disjunctive logic programs.

Acknowledgments

The authors are grateful to Francesco Buccafurri for several useful discussions on the notion of unfounded sets and on computational issues. The paper benefited greatly from the comments of the referees for *Information and Computation*; they suggested that the results be extended to programs with function symbols. Theorem 5.11 was suggested by a referee for ILPS '95.

References

[1] Apt K.R. and Bol R.N. (1994), Logic Programming and Negation: A Survey, *Journal of Logic Programming*, **19/20**, 9–71.

[2] Baral, C. (1991), Generalized negation as failure and semantics of normal disjunctive logic programs, *in* "Proc. Intl Conf. on Logic Programming and Automated Reasoning (LPAR '92)," St. Petersburg, A. Voronkov, ed., LNCS 624, pp.309–319, Springer.

[3] Baral, C. and Gelfond, M. (1994), Logic Programming and Knowledge Representation *Journal of Logic Programming*, **19/20**, 73–148.

[4] Bell, C., Nerode, A., Ng, R. and Subrahmanian, V.S. (1992), Implementing Deductive Databases by Linear Programming, *in* "Proc. 1992 ACM SIGMOD/SIGACT/SIGART Symp. on Principles of Database Systems," San Diego, pp.283–291.

[5] Bell, C., Nerode, A., Ng, R. and Subrahmanian, V. S. (1993), Implementing Stable Semantics by Linear Programming, *in* "Proc. of the Second International Workshop on Logic Programming and Nonmonotonic Reasoning (LPNMR-93)," Lisbon, Portugal, pp.23–42, MIT Press.

[6] Bell, C., Nerode, A., Ng, R. and Subrahmanian, V.S. (1994), Mixed Integer Programming Methods for Computing Non-Monotonic Deductive Databases, *Journal of ACM*, **41**(6), 1178–1215.

[7] Ben-Eliyahu, R. and Dechter, R. (1994), Propositional Semantics for Disjunctive Logic Programs, *Annals of Mathematics and Artificial Intelligence*, Baltzer AG, Science Publishers, **12**, 53–87.

[8] Ben-Eliyahu, R. and Palopoli, L. (1994), Reasoning with Minimal Models: Efficient Algorithms and Applications, *in* "Proc. Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR-94)," pp. 39–50.

[9] Bondarenko, A., Kowalski, R. and Toni, F. (1993), An Assumption-Based Framework for Non-Monotonic Reasoning, *in* "Proceedings of the Second International Workshop on Logic Programming and Nonmonotonic Reasoning (LPNMR-93)," Lisbon, pp. 171–189, MIT Press.

[10] Brass, S. and Dix, J. (1995), Disjunctive Semantics Based upon Partial and Bottom-Up Evaluation, *in* "Proc. of the 12th Int. Conf. on Logic Programming," Tokyo, pp. 199–213, MIT Press.

[11] Buccafurri, F., Leone, N. and Rullo, P. (1996), Stable Models and their Computation for Logic Programming with Inheritance and True Negation, *Journal of Logic Programming*, **27**(1), 5–43.

[12] Cholewiński, P., Marek, V.W., Mikitiuk A. and Truszczyński, M. (1995), Experimenting with Nonmonotonic Reasoning, *in* "Proc. of the 12th International Conference on Logic Programming," pp.267–281, MIT Press.

[13] Cuadrado, J. and Pimentel, S. (1989), A Truth Maintenance System Based on Stable Models, *in* "Proc. 1989 North American Conf. on Logic Programming," pp.274–290.

[14] Davis, M. and Putnam, H. (1960), A Computing Procedure for Quantification Theory, *Journal of ACM*, **7**(3), 201–215.

[15] Dix, J. (1992), Semantics of Logic Programs: Their Intuitions and Formal Properties. An Overview. *in* "Logic, Action and Information. Proc. of the Konstanz Colloquium in Logic and Information (LogIn'92)," pp.241-329, DeGruyter.

[16] Dix, J. and Müller, M. (1993), Implementing Semantics of Disjunctive Logic Programs Using Fringes and Abstract Properties, *in* "Proc. of the Second International Workshop on Logic Programming and Nonmonotonic Reasoning (LPNMR-93)," Lisbon, pp. 43–59, MIT Press.

[17] Eiter, T. and Gottlob, G. (1995), On the Computational Cost of Disjunctive Logic Programming: Propositional Case, *Annals of Mathematics and Artificial Intelligence*, J. C. Baltzer AG, Science Publishers, **15**, 289–323.

[18] Eiter, T., Gottlob, G. and Mannila, H. (1994), Adding Disjunction to Datalog, *in* "Proc. ACM PODS-94," pp. 267–278.

[19] Eiter, T., Gottlob, G. and Mannila, H. (1997), Disjunctive Datalog, *ACM Transactions on Database Systems*, Forthcoming.

[20] Eiter, T., Leone, N. and Sacca', D. (1996), The Expressive Power of Partial Models for Disjunctive Deductive Databases, *in* "Proc. of International Workshop on Logic in Databases–LID'96," San Miniato, Pisa, Italy, pp. 261–280.

[21] Eiter, T., Leone, N. and Saccá, D. (1997), On the Partial Semantics for Disjunctive Deductive Databases, *Annals of Mathematics and Artificial Intelligence*, J. C. Baltzer AG, Science Publishers, Forthcoming.

[22] Elkan, C. (1990), A rational Reconstruction of Nonmonotonic Truth Maintenance Systems, *Artificial Intelligence*, **43**, 219–234.

[23] Fernández, J.A., Lobo, J., Minker, J. and Subrahmanian V.S. (1993), Disjunctive LP + Integrity Constraints = Stable Model Semantics, *Annals of Mathematics and Artificial Intelligence*, **8**(3-4), 449–474.

[24] Fernández, J.A. and Minker, J. (1991), Bottom-up Evaluation of Hierarchical Disjunctive Deductive Databases, *in* "Proc. Eighth Intl. Conference on Logic Programming," pp.660–675, MIT Press.

[25] Fernández, J.A. and Minker, J. (1992), Semantics of Disjunctive Deductive Databases, *in* "Proc. 4th Intl. Conference on Database Theory (ICDT-92)," Berlin, pp. 21–50.

[26] Fernández, J.A. and Minker, J. (1995), Bottom-Up Computation of Perfect Models for Disjunctive Theories, *Journal of Logic Programming*, **25**(1), 33–51.

[27] Fitting, M. (1993), The Family of Stable Models, *Journal of Logic Programming*, **17**(2/3&4), 197–225.

[28] Fitting, M. (1991), Well-Founded Semantics Generalized, *in* "Logic Programming, Proc. of the 1991 International Symposium," V. Saraswat and K. Ueda (eds.), pp. 71–84, MIT Press, Cambridge, MA.

[29] Fuentes, L.O. (1991), *Applying Uncertainty Formalisms to Well-Defined Problems*, manuscript.

[30] Gelfond, M. and Lifschitz, V. (1991), Classical Negation in Logic Programs and Disjunctive Databases, *New Generation Computing*, **9**, 365–385.

[31] Gelfond, M. and Lifschitz, V. (1988), The Stable Model Semantics for Logic Programming, *in* "Proc. of Fifth Logic Programming Symposium," pp. 1070–1080, MIT Press.

[32] Gottlob, G. (1994), Complexity and Expressive Power of Disjunctive Logic Programming, *in* "Proc. of the International Logic Programming Symposium (ILPS-'94)," M. Bruynooghe ed., Ithaca NY, pp. 23–42, MIT Press.

[33] IFIP-GI Workshop (1994), "Disjunctive Logic Programming and Disjunctive Databases," 13-th IFIP World Computer Congress.

[34] Inoue, K. and Sakama, C. (1996), A Fixpoint Characterization of Abductive Logic Programs, *Journal of Logic Programming*, **27**, No.2, 107–136.

[35] Leone, N. and Rullo, P. (1992), Safe Computation of the Well-Founded Semantics of DATALOG Queries, *Information Systems*, **17**(1), 17–31.

[36] Leone, N. , Rullo, P. and Scarcello, F. (1995), Declarative and Fixpoint Characterizations of Disjunctive Stable Models, *in* "Proc. of International Logic Programming Symposium–ILPS'95," Portland, Oregon, pp.399-413, MIT Press.

[37] Lobo, J., Minker, J. and Rajasekar, A. (1992), "Foundations of disjunctive logic programming," The MIT Press.

[38] Loveland, D. W. and Reed, D. W. (1989), A near-Horn Prolog for compilation, Technical Report CS-1989-14, Duke University and *in* "Computational Logic: Essays in Honor of Alan Robinson."

[39] Loveland, D. W., Reed, D. W. and Wilson D. S. (1995), SATCHMORE: SATCHMO with RElevancy, *Journal of Automated Reasoning*, **14**, 325–351.

[40] Manthey, R. and Bry, F. (1988), SATCHMO: A theorem prover implemented in Prolog, *in* "Proc. of the Ninth Int'l Conf. on Automated Deduction."

[41] Marek, W. and Subrahmanian, V.S. (1989), The Relationship between Logic Program Semantics and Non-Monotonic Reasoning, *in* "Proc. of the 6th International Conference on Logic Programming – ICLP'89," pp. 600–617, MIT Press.

[42] Marek, W. and Subrahmanian, V.S. (1992), The Relationship between Stable, Supported, Default and Autoepistemic Semantics for General Logic Programs, *Theoretical Computer Science*, **103**, 365–386.

[43] Marek, W. and Truszczyński, M. (1991), Autoepistemic Logic, *Journal of ACM*, **38**(3), 588–619.

[44] Minker, J. (1982), On Indefinite Data Bases and the Closed World Assumption, *in* "Proc. of the 6^{th} Conference on Automated Deduction (CADE-82)," pp. 292–308.

[45] Minker, J. and Rajasekar, A. (1990), A Fixpoint Semantics for Disjunctive Logic Programs, *Journal of Logic Programming*, **9**(1), 45–74.

[46] Niemelä, I. and Simons, P. (1996), Efficient Implementation of the Well-founded and Stable Model Semantics, *in* "Proc. of JICSLP '96," MIT Press.

[47] Poole, D. (1989), What the lottery paradox tells us about default reasoning, *in* "Proc. of the First Int'l Conf. on Principles of Knowledge Representation and Reasoning," R. Brachman, H. Levesque and R. Reiter editors, pp. 333–340.

[48] Przymusinska, H. and Przymusinski, T. (1988), Weakly Perfect Model Semantics for Logic Programs, *in* "Proc. Fifth Int. Conf. and Symp. on Logic Programming," pp.1106-1120.

[49] Przymusinski, T. (1988), On the Declarative Semantics of Deductive Databases and Logic Programming, *in* "Foundations of deductive databases and logic programming," Minker, J. ed., ch. 5, pp.193–216, Morgan Kaufman, Washington, D.C.

[50] Przymusinski, T. (1991), Stable Semantics for Disjunctive Programs, *New Generation Computing*, **9**, 401–424.

[51] Przymusinski, T. (1994), Static Semantics for Normal and Disjunctive Logic Programs, *Annals of Mathematics and Artificial Intelligence*, J. C. Baltzer AG, Science Publishers, **15**.

[52] Przymusinski, T. (1990), Stationary semantics for disjunctive logic programs and deductive databases, *in* "Proc. of North American Conference on Logic Programming," pp. 40–62.

[53] Reed, D. W., Loveland D. W. and Smith B. T. (1991), An Alternative Characterization of Disjunctive Logic Programs, *in* "Proc. of the International Logic Programming Symposium (ILPS-'91)," pp.55–68.

[54] Reiter, R. (1978), On Closed-World Data Bases, *in* "Logic and Data Bases," H. Gallaire, J. Minker (eds.), pp. 55–76, Plenum Press.

[55] Ross, K.A. (1990), Modular Stratification and Magic Sets for Datalog Programs with Negation, *in* "Proc. ACM Symposium on Principles of Database Systems," pp.161-171.

[56] Ross. K.A. (1990), The Well Founded Semantics for Disjunctive Logic Programs, *in* "Deductive and Object–Oriented Databases," W. Kim, J.–M. Nicolas and S. Nishio, ed., pp.385–402, Elsevier Science Publishers B. V.

[57] Saccá, D. and Zaniolo, C. (1990), Stable Models and Nondeterminism in Logic Programs with Negation, *in* "Proc. ACM Symposium on Principles of Database Systems," pp.205-217.

[58] Sakama, C. (1989), Possible model semantics for disjunctive databases, *in* "Proc. of the first international conference on deductive and object oriented databases," pp.1055–1060.

[59] Sakama, C., Inoue, K. (1995), Embedding Circumscriptive Theories in General Disjunctive Programs, *in* "Proc. LPNMR '95," pp.344–357.

[60] Seipel, D., Minker, J. and Ruiz, C. (1997), Model Generation and State Generation for Disjunctive Logic Programs, *Journal of Logic Programming*, Forthcoming.

[61] Ruiz, C. and Minker, J. (1995), Computing Stable and Partial Stable Models of Extended Disjunctive Logic Programs, *in* "Proc. of Nonmonotonic Extensions of Logic Programming," *LNCS* 927, pp.205–229, Springer-Verlag.

[62] Stuber, J. (1994), Computing Stable Models by Program Transformation, *in* "Proc. 11th International Conference on Logic Programming," pp. 58–73, MIT Press.

[63] Subrahmanian, V.S., Nau, D. and Vago, C. (1995), WFS + Branch and Bound = Stable Models, *IEEE Transactions on Knowledge and Data Engineering*, **7**, No. 3, 362-377.

[64] Van Gelder, A., Ross, K. A. and Schlipf, J. S. (1991), The Well-Founded Semantics for General Logic Programs, *Journal of ACM*, **38**(3), 620–650.

[65] Vardi, M. (1982), Complexity of relational query languages, *in* “Proceedings 14th ACM STOC,” pp. 137–146.