UNFOUNDED SETS AND PARTIAL STABLE MODELS FOR DISJUNCTIVE DEDUCTIVE DATABASES*

Thomas Eiter Nicola Leone
Information System Department
Technical University of Vienna
1040 Vienna, Austria
(eiter|leone)@dbai.tuwien.ac.at

Domenico Saccà
DEIS
Università della Calabria
87030 Rende, Italy
sacca@si.deis.unical.it

Abstract

Partial stable models for deductive databases, i.e., normal function-free logic programs (also called datalog programs), have two equivalent definitions: one based on 3-valued logics and another based on the notion of unfounded set. The notion of partial stable model has been extended to disjunctive deductive databases using 3-valued logics by Przymusinski. But, due to the lack of a suitable notion of unfounded set for disjunctive logic programs, it was so far impossible to characterize partial stable models in terms of unfounded sets in the disjunctive case.

In this paper, we present a new notion of unfounded set, which extends the classical definition of Van Gelder et al. from normal to disjunctive logic programs. Then, by using this notion, we provide a declarative characterization of partial stable models of disjunctive logic programs in terms of unfounded sets. In particular, we show that an interpretation is a partial stable model if and only if it is founded (i.e., all its true atoms are justifiable) and its false atoms form a maximal unfounded set.

1 Introduction

Deductive databases comprise explicit data and logical rules for computing queries on the data, which amount to a logic program without function symbols, i.e., datalog programs (extended with negation) [18, 3]. The need for representing disjunctive information led to disjunctive deductive databases [11], which fertilized also work on

*This work has been partially supported by *Christian Doppler Lab. for Expert Systems*, by *Istituto per la Sistemistica e l'Informatica (ISI-CNR)*, by the EC-US033 project "DEUS EX MACHINA: Non-Determinism in Deductive Databases", and by a MURST grant (40% share) under the project "Sistemi formali e strumenti per basi di dati evolute." .

disjunctive logic programming [10, 12]. In this paper, we address disjunctive deductive databases, which are identified with disjunctive function-free logic programs [8], and refer to them simply as programs.

To date, the *total stable* (or *2-valued stable*) model semantics [6, 14, 7] is a dominant semantics for both normal and disjunctive programs. It is widely acknowledged that the total stable semantics is the natural meaning to be assigned to a (disjunctive) program, provided that it admits some stable models. However, a drawback of this semantics is that several meaningful programs do not admit any total stable model, which results in a loss of information. To handle these cases, Przymusinski defined the notion of *partial stable* (or *3-valued stable*) model for both normal and disjunctive programs [14]. Partial stable models exist for a wider class of programs (actually, for normal programs they always exist). Moreover, on the class of (disjunctive) stratified programs, partial stable models coincide with total stable models. For normal programs, partial stable models have been also defined using the notion of unfounded set [16, 17]; but, so far this definition has not been extended to the disjunctive case, because of the lack of a suitable notion of unfounded set for disjunctive programs.

In this paper we give an equivalent definition of partial stable semantics for disjunctive programs. To this end, we present a new notion of unfounded set for disjunctive programs, which generalizes the analogous notion given for normal (i.e., nondisjunctive) programs in [20] (moreover, on total interpretations, it coincides with the definition given in [9] for disjunctive programs). Intuitively, as for normal programs, the notion of unfounded set can be seen as a form of CWA rule: if a set X of undefined atoms is an unfounded set w.r.t. an interpretation I, then the truth value of the atoms in X can be switched to false without violating any rule of the program.

Once defined unfounded sets, we provide an appealing characterization of 3-valued stable models in terms of unfounded sets. In particular, we show that an interpretation is a 3-valued stable model if and only if its true atoms are "justifiable" (see below) and the set of its false atoms is a maximal unfounded set.

To our knowledge, this is the first characterization of 3-valued stable models in terms of unfounded sets. It extends similar studies conducted in [20, 16, 17, 9], which investigated the relationships between unfounded sets and stable models in more restricted domains.

The sequel of the paper is organized as follows. After some preliminary on disjunctive deductive databases, Section 2 presents the notion of unfounded set and provides our definition of *P-stable* (*Partial stable*) model in terms of unfounded sets. Section 3 demonstrates the equivalence between P-stable models and the 3-valued stable models of [14]. Finally, Section 4 addresses some further issues and concludes the paper. Additional results on P-stable models and restricted notions thereof are given in the extended paper [4].

2 Partial Stable Models

Disjunctive Deductive Databases

Deductive databases are described in a logical language using variables from a set V, predicates of nonnegative arity from a countable set Π, and constants from a finite set Λ.

An *atom* is of the form $a(t_1, \ldots, t_n)$, where a is a *predicate* of arity n in Π, each $t_1, \ldots, t_n$ is either a constant in Λ or a variable in V. A *literal* is either a *positive literal* p or a *negative literal* $\neg p$, where p is an atom. We often 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, and given a set A of literals, $\neg.A$ denotes the set $\{\neg.L \mid L \in A\}$. A *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 $b_1, \ldots, b_k, \neg b_{k+1}, \ldots, \neg b_m$ is the *body* of r. We denote by $H(r)$ the set $\{a_1, \ldots, a_n\}$ of the head atoms, and by $B(r)$ the set $\{b_1, \ldots, b_k, \neg b_{k+1}, \ldots, \neg b_m\}$ of the body literals. Moreover, $B^+(r)$ and $B^-(r)$ denote the set of positive and negative literals occurring in $B(r)$, respectively. A *disjunctive deductive database* (simply *program* hereafter) is a finite set of rules. Thus, a disjunctive deductive database is a disjunctive datalog program [5]. A distinction between extensional (database) predicates and intensional (derived) predicates can be made as usual [18, 8]; for simplicity, we do not expand on this in detail.

A program is *normal* if it contains only *normal* clauses, i.e., the head contains exactly one atom ($n = 1$). A $\neg$-free (resp. $\vee$-free) program is called *positive* (resp. *normal*). An atom, a literal, a rule or program is *ground* if no variable appears in it. As common, we assume that the sets Π and Λ are implicitly given by the predicates and constants occurring in the considered program.

Interpretations and Unfounded Sets

We denote by $\mathcal{B}_{\mathcal{LP}}$ the *Herbrand Base* of $\mathcal{LP}$, i.e. the set of all ground atoms constructible from the predicates appearing in the rules of $\mathcal{LP}$ and the constants occurring in $\mathcal{LP}$.

We denote by *ground*($\mathcal{LP}$) the set of all ground instances of the rules occurring in $\mathcal{LP}$, where a *ground instance* of a rule r is obtained by any substitution σ that maps each variable X in r to a constant occurring in $\mathcal{LP}$. By $Lit_{\mathcal{LP}}$ we denote the set of all literals involving atoms from $\mathcal{B}_{\mathcal{LP}}$, i.e. $Lit_{\mathcal{LP}} = \mathcal{B}_{\mathcal{LP}} \cup \neg.\mathcal{B}_{\mathcal{LP}}$.

An *(Herbrand) interpretation* I of a program $\mathcal{LP}$ is a consistent set of ground literals, i.e. $I \subseteq Lit_{\mathcal{LP}}$ 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*. A (possibly empty) conjunction of ground literals C is *true* in I if every literal in C is in I, *false* in I if there exists some literal A in

C for which $\neg A \in I$ and *undefined* in I otherwise. We denote by I^+ and I^- the set of positive and negative literals occurring in I, respectively. $\overline{I}$ denotes the set of undefined atoms w.r.t. I (that is, $\overline{I} = \mathcal{B}_{\mathcal{LP}} - (I^+ \cup \neg.I^-)$). The interpretation I is *total* if $\overline{I}$ is empty (that is, $I^+ \cup \neg.I^- = \mathcal{B}_{\mathcal{LP}}$); otherwise, I is *partial*. A *total model* of $\mathcal{LP}$ is a total interpretation M which satisfies each rule r in *ground*($\mathcal{LP}$), i.e., if $B(r) \subseteq M$, then $H(r) \cap M \neq \emptyset$.[1] Observe that we defined no notion of partial model here, as, following the approach adopted in [16, 17] for normal programs, we will define P-stable models in terms of a foundedness condition and unfounded sets. A notion of partial model is anyway presented in the next section.

Definition 1 (Unfounded Set) *Let I be an interpretation for $\mathcal{LP}$. A set $X \subseteq \mathcal{B}_{\mathcal{LP}}$ of ground atoms is an* unfounded set *for $\mathcal{LP}$ w.r.t. I if, for each $a \in X$, at least one of the following conditions holds for each rule $r \in$ ground($\mathcal{LP}$) such that $a \in H(r)$:*

1. $B(r) \cap (\neg.I \cup X) \neq \emptyset$, *or*
2. $H(r) \not\subseteq (\neg.I \cup X)$.

An unfounded set X is consistent *w.r.t. I if $X \cap I^+ = \emptyset$.*

For normal programs unfounded sets have been introduced to draw negative inferences for the constructive definition of the well-founded model [20] (recall that the well-founded model is indeed defined as the limit of the sequence $I_0 = \emptyset$, $I_n = T_P(I_{n-1}) \cup \neg.U_P(I_{n-1})$, where T_P is the immediate consequence operator and $U_P(I)$ denotes the greatest unfounded set w.r.t. I). Thus, they can be seen as a sort of CWA rule allowing to derive negative information (i.e., the falsity of some atoms).

Unfounded sets for disjunctive programs play essentially the same role. Indeed, if X is a consistent unfounded set (i.e., $I \cup \neg.X$ is an interpretation), for every rule r having an element of X in the head, either the body of r is false w.r.t. $I \cup \neg.X$ (Condition 1 of Definition 1), or the head of r is not false w.r.t. $I \cup \neg.X$ (Condition 2 of Definition 1). Thus, if we think in terms of classical (2-valued) interpretations, it is possible to extend I by assuming that all atoms in X are false,[2] without explicitly violating r; in fact, rule r is still satisfiable: if Condition 1 applies, then r is definitely satisfied (as the body is false), and if Condition 2 applies, r can be satisfied by assuming that the atoms in $H(r) - X$ are true. Since the atoms in X are "defined" only by such rules r (under the intended assignment-based reading of rules), intuitively every atom in X should be false, since the assumption that it is true is not much founded, according to a principle of closed world assumption. (For a more elaborate discussion on this, see Section 5.)

Example 1 Consider the program $\mathcal{LP} = \{a \vee b \leftarrow\}$ and the interpretation $I = \emptyset$. The set $X = \{a\}$ is a (consistent) unfounded set of $\mathcal{LP}$ w.r.t. I. Indeed, the unique rule of *ground*($\mathcal{LP}$) (with a in the head) satisfies Condition 2 of Definition 1, as $H(r) - (X \cup \neg.I) = \{b\}$. Likewise, we can easily verify that $Y = \{b\}$ is a (consistent) unfounded set for $\mathcal{LP}$ w.r.t. I (as $H(r) - (Y \cup \neg.I) = \{a\}$).
For the program $\mathcal{LP} = \{ a \vee b \leftarrow ;\ a \leftarrow b ;\ b \leftarrow a \}$ and the interpretation $I = \{a, b\}$ the only unfounded set is $\emptyset$.

We remark that a first attempt for extending the notion of unfounded set (from normal) to disjunctive programs may lead to adopt for Condition 2 the stronger requirement $(H(r) - X) \cap I^+ \neq \emptyset$, i.e., an atom from the head of r apart from X is true in I. This definition has been adopted in [9], where unfounded sets are used to characterize 2-valued stable models. However, this (stronger) condition turns out to be too strong to capture unfoundedness appropriately with respect to 3-valued semantics of rules that we consider later. Namely, if the head of a rule as well as its body is undefined in I, i.e., neither definitely true nor false, we nevertheless can conclude that –in the extremal case– all atoms in the head except one can be false, without definitely violating the rule (which happens if the body is true and the head is false). Arriving at such a conclusion is essential, however, in a form of the closed world assumption which is inherent to 3-valued stable semantics.

Other notions of unfounded sets are somehow implicit in various attempts to generalize the well-founded semantics from normal to disjunctive programs like, e.g., [1, 13, 15, 10]. Comparing these (implicit) notions to our unfounded sets is quite difficult, as unfounded sets there do not result into flat set of atoms like in Definition 1 (which is consistent to the standard definition of unfounded set for normal programs [20]). Note that [15] does not construct well-founded semantics bottom up as for normal programs [20], due to a lacking proper notion of unfounded set for the construction, but top-down.

For instance, given the program $\mathcal{LP} = \{a \vee b \leftarrow\}$, the stationary semantics [13] adds to $\mathcal{LP}$ the axiom $\neg(a \wedge b)$ (the conjunction $a \wedge b$ can be thus considered unfounded); while, according to Definition 1, the (nontrivial) unfounded sets are $\{a\}$ and $\{b\}$. Intuitively, our unfounded sets suggest that either a or b must be switched to false (see below), while stationary semantics (skeptically) states that they can not be true at the same time. Following the iterated construction of the well-founded semantics for normal logic programs, [1] proposes an iterated fixpoint characterization of disjunctive well-founded semantics. Interpretations are replaced by general states, which are consistent sets of disjunctions of literals (rather than literals) cf. [10]; iteratively defined general states M_n correspond to the interpretations I_n (see above) and refer to the maximum set F_S of negative disjunctions that can be assumed true (i.e., the maximum set of conjunctions that can be assumed false) from the program on state S, which corresponds to the greatest unfounded set $U_P(I)$. E.g., for $\mathcal{LP} = \{a \vee b \leftarrow\}$ and $S = \emptyset$, $F_S = \{\neg a \vee \neg b\}$; like in stationary semantics, it is (skeptically) stated that $a \wedge b$ can not be true.

It is worth pointing out that Definition 1 generalizes the standard definition of unfounded set given for normal programs in [20]: X is a *(standard) unfounded set* for a normal program $\mathcal{LP}$ w.r.t. I if, for each $a \in X$, for each rule with head a, $B(r) \cap (\neg.I \cup X) \neq \emptyset$ holds.

[1] In the context of classical (2-valued) logic, where interpretations are total, (total) models are identified with their positive parts. Within the context of partial (3-valued) interpretations, however, their negative parts are mentioned as well, to stay consistent with the framework.

[2] Note that assuming false a literal which is in I^+ would cause inconsistency.

Proposition 1 *Let $\mathcal{LP}$ be a normal program and I an interpretation for $\mathcal{LP}$. $X \subseteq \mathcal{B}_{\mathcal{LP}}$ is an unfounded set for $\mathcal{LP}$ w.r.t. I according with Definition 1 if and only if it is a standard unfounded set for $\mathcal{LP}$ w.r.t. I.*

Proof. For a normal program, Condition 2 of Definition 1 is never satisfied since $H(r) \subseteq X$, as $H(r)$ is made of a single element which belongs to X. Thus, for a normal program, Definition 1 reduces to the standard definition of unfounded set.

It is known that for normal programs, the union of two unfounded sets is an unfounded set as well. This property does not extend to disjunctive programs.

Example 2 Consider again the program $\mathcal{LP} = \{a \vee b \leftarrow\}$ and the interpretation $I = \emptyset$. In Example 1 we have seen that both $\{a\}$ and $\{b\}$ are unfounded sets for $\mathcal{LP}$ w.r.t. I, but we can easily verify that the union $X = \{a, b\}$ is not. Indeed, take any element in X, say a; although a is not false w.r.t. I, b is not unfounded as a is also in X (so that Condition 2 of Definition 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 not false (otherwise the rule $a \vee b \leftarrow$ becomes unsatisfiable).

Since the union of all unfounded sets is not necessarily an unfounded set, we cannot define the greatest unfounded set of an interpretation; instead, we next provide a notion of maximal unfounded set.

A consistent unfounded set X w.r.t. I is a *maximal unfounded set* for $\mathcal{LP}$ w.r.t. I if no proper superset Y of X is a consistent unfounded set for $\mathcal{LP}$ w.r.t. I. Next example shows a number of maximal unfounded sets.

Example 3 Let $\mathcal{LP} = \{a \vee b \leftarrow\}$. Both $X_1 = \{a\}$ and $X_2 = \{b\}$ are maximal unfounded sets for $\mathcal{LP}$ w.r.t. $I = \emptyset$.

X_1 is the only maximal unfounded set for $\mathcal{LP}$ w.r.t. the interpretation $\{b\}$ (note that X_2 is not a consistent unfounded set w.r.t. $\{b\}$).

Let $\mathcal{LP}_1$ be the following program:

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

$M_1 = \{a, \neg b, \neg c\}$ has the single maximal unfounded set $X_1 = \{b, c\}$; $M_2 = \{b, c, \neg a, \neg d\}$ has the single maximal unfounded $X_2 = \{a, d\}$.

Intuitively, a maximal unfounded set X is a maximal set of atoms that can be assumed false without violating the satisfiability of the rules of the program that contain some atom from X. Consistency of the unfounded set X ensures that the assumption of the falsity of the literals in X is consistent with I (that is, $I \cup \neg.X$ is an interpretation).

For instance, we have seen in the above examples that for the program $\mathcal{LP} = \{a \vee b \leftarrow\}$ both $\{a\}$ and $\{b\}$ are (consistent) unfounded sets w.r.t. $I = \emptyset$ while $\{a, b\}$ is not an unfounded set. Indeed, we can assume the falsity of either a or b without violating the satisfiability of $a \vee b \leftarrow$ (which can be satisfied by making true b or a, respectively); but we cannot assume the falsity of both a and b at the same time, as in the latter case we would have no way to satisfy rule $a \vee b \leftarrow$.

P-Stable Models

Once we provided a notion of unfounded set for disjunctive programs, we define P-stable models for such programs by extending the respective definition given in [16, 17] for normal programs.

To proceed, we need the concept of foundedness of an interpretation. *Foundedness*, called *justifiability* in [21], basically prescribes that every positive literal in an interpretation must be be derived from the rules possibly using negative literals as additional axioms.

Let for any positive (i.e., $\neg$-free) program P denote $\mathrm{MM}(P)$ the set of the minimal total models of P [11], where a total model M is *minimal* iff there does not exist a total model N of P such that $N^+ \subset M^+$. Moreover, let for any program $\mathcal{LP}$ and interpretation I be $\mathcal{LP}(I)$ the program obtained from $ground(\mathcal{LP})$ as follows: (i) remove all rules r having a negative literal $\neg p \in B(r)$ such that $\neg p \notin I^-$, and (ii) remove all negative literals from the remaining rules.

Definition 2 *An interpretation I of a program $\mathcal{LP}$ is* founded *iff $I^+ = M^+$ for some $M \in \mathrm{MM}(\mathcal{LP}(I))$.*

Notice that $\mathcal{LP}(I)$ is essentially the 2-valued Gelfond-Lifschitz transform [6], carried out in the context of partial interpretations.

Example 4 Consider the program $\mathcal{LP}_1$ of Example 3. Interpretation $M_1 = \{a, \neg b, \neg c\}$ is founded. Indeed, $\mathcal{LP}_1(M_1) = \{a \vee b \leftarrow\}$, and there exists a minimal total model N of $\mathcal{LP}_1(M_1)$ such that $N^+ = M_1^+$, namely $N = \{a, \neg b\}$. For $M_2 = \{b, c, \neg a, \neg d\}$, $\mathcal{LP}_1(M_2) = \{a \vee b \leftarrow, a \vee c \leftarrow\}$. Also M_2 is founded, since $N = \{b, c, \neg a\}$ is a minimal total model of $\mathcal{LP}_1(M_2)$ and $N^+ = M_2^+$.

For total interpretations foundedness alone is sufficient to characterize (2-valued) stable models, which are indeed defined as the total founded interpretations of the program [6, 14, 7]. However, as for normal programs [16], foundedness alone is not sufficient to single out 3-valued stable models, as some conditions on the false literals are needed. Intuitively, what must be imposed on the false literals for a founded interpretation I to be a (partial) stable model is the consistency with the closed world assumption. In other words, every false literal in I must be definitely not derivable from the rules of the program (assuming I), and no further literal must be declarable false without violating some rule of the program. As we informally observed above, this consistency with the CWA is precisely formalized by the notion of maximal unfounded set.

Definition 3 (P-stable Model) *Let $\mathcal{LP}$ be a program and M an interpretation of it. Then M is a* P-stable (partial stable) *model of $\mathcal{LP}$ if the following conditions hold:*

(a) M is founded,

(b) $\neg.M^-$ is a maximal unfounded set for $\mathcal{LP}$ w.r.t. M.

Note that Condition (a) characterizes the true atoms of M, imposing that they have to be "justified" [21], while condition (b) characterizes the false atoms of M, requiring that they must be all and only the atoms that can be assumed false consistently with M and with the rules of the program (further negative assumptions would make some rule unsatisfiable).

Example 5 Consider the program $\mathcal{LP}_1$ of Example 3. Both $M_1 = \{a, \neg b, \neg c\}$ and $M_2 = \{b, c, \neg a, \neg d\}$ are P-stable models of $\mathcal{LP}_1$. Indeed, we have seen in Example 4 that M_1 and M_2 are founded, and Example 3 showed that $\neg.M_1^-$ and $\neg.M_2^-$ are maximal unfounded sets w.r.t. M_1 and M_2, respectively. On the other hand, the interpretation $M_3 = \{a, \neg d\}$ is not a P-stable model of $\mathcal{LP}_1$. While M_3 is founded (note that $\mathcal{LP}_1(M_3) = \{a \vee b \leftarrow \ , \ a \vee c \leftarrow \ \}$), $\neg.M_3^-$ is not a maximal unfounded set w.r.t. M_3: indeed, M_3 has the single maximal unfounded set $X = \{b, c\}$.

3 P-stable and 3-Valued Stable Models

In this section we show that the definition of P-stable model (Definition 3) is equivalent to the definition of 3-valued stable model as given in [14]. To this end, following [14], we define a 3-valued logic with T *(True)*, F *(False)*, and U *(Undefined)*, ordered as $F < U < T$. Given a program $\mathcal{LP}$, an interpretation I of $\mathcal{LP}$, and a ground literal L,

$$value_I(L) = \begin{cases} T, & \text{if } L \in I; \\ F, & \text{if } L \in \neg.I; \\ U, & \text{otherwise (i.e., } L \in \overline{I} \cup \neg.\overline{I}). \end{cases}$$

The value of a conjunction $C = L_1, \ldots, L_n$ of ground literals is the minimal value of the literals in the conjunction (i.e., $value_I(C) = min_{1 \leq i \leq n} value_I(L_i)$); if C is empty, then we assume $value_I(C) = T$. The value of a disjunction $D = L_1 \vee \cdots \vee L_n$ of ground literals is the maximum value of the literals in the disjunction (i.e., $value_I(D) = max_{1 \leq i \leq n} value_I(L_i)$). A ground rule r is *satisfied* by I if $value_I(H(r)) \geq value_I(B(r))$.

Definition 4 (3-valued Model [14]) *Let $\mathcal{LP}$ be a program and M be an interpretation for $\mathcal{LP}$. Then, M is a 3-valued model for $\mathcal{LP}$ if every rule r in ground($\mathcal{LP}$) is satisfied by M.*

The definition of 3-valued stable model involves a 3-valued version of the Gelfond-Lifschitz (GL) transformation [14] and resorts to the notion of 3-valued program that we report next. *A 3-valued program* is a program $\mathcal{LP}$ where the constants T, U and F may occur as body literals in the rules of $\mathcal{LP}$. We assume that in every interpretation I the literal T is true, the literal F is false, and the literal U is undefined. Thus, such literals can be thought of as already interpreted ground literals.

Given an interpretation I for a program $\mathcal{LP}$, the *GL-transformation* $\frac{\mathcal{LP}}{I}$ of $\mathcal{LP}$ w.r.t. I is the positive 3-valued program obtained from *ground*($\mathcal{LP}$) by replacing in the body of every rule all negative literals which are true (resp. undefined, false) by T (resp. U, F). An interpretation M for $\mathcal{LP}$ is a *3-valued model for* $\frac{\mathcal{LP}}{I}$ if it satisfies every rule in $\frac{\mathcal{LP}}{I}$. A 3-valued model M for $\frac{\mathcal{LP}}{I}$ (resp. $\mathcal{LP}$) is *minimal* if there does not exist a 3-valued model $M_1 \neq M$ for $\frac{\mathcal{LP}}{I}$ (resp. $\mathcal{LP}$) such that $M_1^+ \subseteq M^+$ and $M_1^- \supseteq M^-$. In [14] it has been proven that $\frac{\mathcal{LP}}{I}$ has at least one minimal 3-valued model.

Definition 5 (3-valued Stable Model [14]) *Let $\mathcal{LP}$ be a program and M be a 3-valued model for $\mathcal{LP}$. Then, M is a* 3-valued stable model *for $\mathcal{LP}$ iff M is a minimal 3-valued model of $\frac{\mathcal{LP}}{M}$.*

Notice that each 3-valued stable model of $\mathcal{LP}$ is a 3-valued model of $\mathcal{LP}$, and in fact a minimal 3-valued model [14].

Even if Definitions 5 and 3 are apparently quite different (the former relies on three valued logic, while the latter is based on unfounded sets), they yield exactly the same notion of stable model.

Example 6 Consider the program $\mathcal{LP}_1$ of Example 3:

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

We have seen in Example 5 that $M_1 = \{a, \neg b, \neg c\}$ is a P-stable model for $\mathcal{LP}_1$. It is easy to see that M_1 is also a 3-valued stable model for $\mathcal{LP}_1$. Indeed, $\frac{\mathcal{LP}_1}{M_1}$ is the following 3-valued program:

$$a \vee b \leftarrow \qquad d \leftarrow U, \ T \qquad a \vee c \leftarrow U,$$

and M_1 is a minimal 3-valued model of $\frac{\mathcal{LP}}{M_1}$. Similarly, $M_2 = \{b, c, \neg a, \neg d\}$, which has been proven to be a P-stable model of $\mathcal{LP}_1$ in Example 5, is a 3-valued stable model for $\mathcal{LP}_1$, as it is a minimal 3-valued model of $\frac{\mathcal{LP}_1}{M_2}$, which is as follows:

$$a \vee b \leftarrow \qquad d \leftarrow T, \ F \qquad a \vee c \leftarrow T.$$

However, the interpretation $M_3 = \{a, \neg d\}$ is not a 3-valued stable model of $\mathcal{LP}_1$; in fact, M_3 is not a 3-valued model of $\mathcal{LP}_1$ (it violates the rule $d \leftarrow \neg d, \neg b$), and hence it can not be a 3-valued stable model of $\mathcal{LP}_1$.

The sequel of the section is devoted to formally demonstrate the equivalence between Definitions 5 and 3. The proof of this equivalence requires some preliminary result.

Lemma 1 *Let M be a P-stable model for a program $\mathcal{LP}$. Then, M is a 3-valued model for both $\mathcal{LP}$ and $\frac{\mathcal{LP}}{M}$.*

Proof. It is immediately recognized that M is a 3-valued model for $\mathcal{LP}$ if and only if it is a 3-valued model for $\frac{\mathcal{LP}}{M}$. It is thus sufficient to prove that M is a 3-valued model for $\mathcal{LP}$. Let r be a rule in *ground*($\mathcal{LP}$). We have to prove that: (a) $value_M(H(r)) = F \Rightarrow value_M(B(r)) = F$, and (b) $value_M(H(r)) = U \Rightarrow value_M(B(r)) \neq T$.
(a). If $value_M(H(r)) = F$ then all atoms in the head of r are false. Since M is a

P-stable model, from Definition 3, $\neg.M^-$ is an unfounded set w.r.t. M. Therefore, r (whose head contains elements in $\neg.M^-$) must satisfy at least one of the unfoundedness conditions of Definition 1 for $X = \neg.M^-$ w.r.t. M. Now, Condition 2 cannot be verified, as $H(r) \subseteq \neg.M^-$. Therefore, $B(r) \cap \neg.M \neq \emptyset$, and, as a consequence, $value_M(B(r)) = F$ (note that, since $X = \neg.M^-$, $\neg.M \cup X = \neg.M$).
(b). We prove the contrapositive: $value_M(B(r)) = T \Rightarrow value_M(H(r)) \neq U$. Let $value_M(B(r)) = T$ hold. Then, the rule r' obtained from r by deleting all negative body literals is in $\mathcal{LP}(M)$, as all negative body literals of r are true w.r.t. M. The body of r' is true w.r.t. M (as $B(r') \subseteq B(r) \subseteq M$). Since M is founded, by Definition 3 $M^+ = N^+$ for some $N \in \mathrm{MM}(\mathcal{LP}(M))$; consequently, $B(r') \subseteq M$ implies that $H(r') \cap M \neq \emptyset$. Therefore, $value_M(H(r)) = T$ (hence, $value_M(H(r)) \neq U$), as $H(r') = H(r)$.

We are now in the position to prove that every P-stable model is a 3-valued model.

Theorem 1 *Let M be a P-stable model for a program $\mathcal{LP}$. Then, M is a 3-valued stable model for $\mathcal{LP}$.*

Proof. From Definition 5, we have to demonstrate that M is a minimal 3-valued model for $\frac{\mathcal{LP}}{M}$. By Lemma 1, it remains to prove only the minimality of M. We proceed by contradiction. Assume that $M_1 \neq M$ is a 3-valued model for $\frac{\mathcal{LP}}{M}$ such that $M_1^+ \subseteq M^+$ and $M_1^- \supseteq M^-$. We proceed as follows: we first prove that (a) M_1^+ cannot be a proper subset of M^+ and after that, using (a), we show that (b) M_1^- cannot be a proper superset of M^-. The details of the proof can be found in [4].

The proof of the converse of Theorem 1 is divided in the following two lemmata.

Lemma 2 *Every 3-valued stable model is founded.*

Proof. Let M be a 3-valued stable model for $\mathcal{LP}$. By contradiction, assume that the total interpretation N of $\mathcal{LP}(M)$ such that $N^+ = M^+$ is not a minimal total model of $\mathcal{LP}(M)$. Then, there exists a total model $N1$ for $\mathcal{LP}(M)$ such that $N1^+$ is a proper subset of M^+. We show that the interpretation $J = N1^+ \cup M^-$ is a 3-valued model for $\frac{\mathcal{LP}}{M}$; this implies that M is not a minimal 3-valued model of $\frac{\mathcal{LP}}{M}$, and hence contradicts that M is a 3-valued stable model of $\mathcal{LP}$. The details of the proof can be found in [4].

Lemma 3 *If M is a 3-valued stable model, then $\neg.M^-$ is a maximal unfounded set w.r.t. M.*

Proof. Let M be a 3-valued stable model for $\mathcal{LP}$. It is easy to see that $\neg.M^-$ is an unfounded set for $\mathcal{LP}$ w.r.t. M. Indeed, since M is a 3-valued model for $\mathcal{LP}$, every rule with a literal from $\neg.M^-$ in the head either has a false body (if all head literals are false) or its head contains some literal which is not in $\neg.M^-$. To see that $\neg.M^-$ is a maximal unfounded set, i.e., there does not exist an X with $\emptyset \neq X \subseteq \overline{M}$ such that $X \cup \neg.M^-$ is an unfounded set w.r.t. M, we demonstrate that the existence of such an X entails $J = \neg.X \cup M$ is a 3-valued model for $\frac{\mathcal{LP}}{M}$, which contradicts the minimality of M for $\frac{\mathcal{LP}}{M}$ and hence that fact that M is a 3-valued stable model of $\mathcal{LP}$. To this end, consider any rule r in $\frac{\mathcal{LP}}{M}$, which is obtained from a rule r' in $ground(\mathcal{LP})$. We it is easy to see that r is satisfied in J. As a consequence, J is a 3-valued of $\frac{\mathcal{LP}}{M}$, which raises the desired contradiction.

We can finally state the main theorem of this section.

Theorem 2 *M is a P-stable model for a program $\mathcal{LP}$ if and only if M is a 3-valued stable model for $\mathcal{LP}$.*

Proof. Follows from Theorem 1 and Lemmata 2 and 3.

Thus, Definition 3 can be seen as a characterization of 3-valued stable models in terms of unfounded sets.

4 Conclusion

In this paper, we have extended the notion of unfounded set from normal programs to disjunctive programs. By using unfounded set, we presented the notion of P-stable model for acceptable partial models for a disjunctive program. These models satisfy two principles: First, the condition of foundedness (also called justifiability [21]), and second, the principle of CWA, extended in a natural way from normal programs to disjunctive programs.

We showed that P-stable models coincide with the 3-valued stable models of Przymusinski [14]. Hence, the definition of P-stable model, can be seen as a characterization of 3-valued stable models in terms of classical interpretations and unfounded sets.

A couple of problems are left open here. In particular, the design of algorithms for efficient query evaluation under P-stable semantics is an interesting issue. Moreover, aspects on the side of knowledge representation would merit investigation, as well as fixpoint operators for computing P-stable models (like the alternating fixpoint operator for the well-founded model [2, 19]).

References

[1] Ch. Baral, 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, Springer, 1992, pp. 309-319.

[2] Ch. Baral, V.S. Subrahmanian, Dualities between alternative semantics for logic programming and non-monotonic reasoning, Journal of Automated Reasoning, 10 (1993) 399–420.

[3] S. Ceri, G. Gottlob, L. Tanca, *Logic Programming and Databases*, Springer, Berlin, 1990.

[4] T. Eiter, N. Leone, D. Saccà, On the partial semantics for disjunctive deductive databases, CD-TR 95/82, Christian Doppler Lab for Expert Systems, TU Vienna, Vienna, 1995.

[5] T. Eiter, G. Gottlob, H. Mannila, Adding disjunction to datalog, in *Proc. Thirteenth ACM Symposium on Principles of Database Systems (PODS '94)*, Minneapolis, Minnesota (USA), ACM, 1994, pp. 267–278. Full paper CD-TR 96/90, Christian Doppler Lab for Expert Systems, TU Vienna, Vienna, 1996.

[6] M. Gelfond, V. Lifschitz, The stable model semantics for logic programming, in *Logic Programming: Proceedings Fifth Intl Conference and Symposium*, Washington, Seattle (USA), MIT Press, 1988, pp. 1070–1080.

[7] M. Gelfond, V. Lifschitz, Classical negation in logic programs and disjunctive databases, New Generation Computing, 9 (1991) 365-385.

[8] J.A. Fernández, J. Minker, Semantics of disjunctive deductive databases, in *Proc. 4th Intl Conference on Database Theory (ICDT '92)*, J. Biskup *et al.* eds., LNCS 646, Springer, 1992, pp. 21–50.

[9] N. Leone, P. Rullo, F. Scarcello, Declarative and fixpoint characterizations of disjunctive stable models, in *Logic Programming: Proc. Twelth Intl Symposium (ILPS '95)*, Portland, Oregon (USA), J. Lloyd ed., MIT Press, 1995, pp. 399–413.

[10] J. Lobo, J. Minker, A. Rajasekar, *Foundations of disjunctive logic programming*, MIT Press, 1992.

[11] J. Minker, On indefinite data bases and the closed world assumption, in *Proc. 6th Conference on Automated Deduction (CADE '82)*, New York, D.W. Loveland ed., LNCS 138, Springer, 1982, pp. 292–308.

[12] J. Minker, Overview of disjunctive logic programming, Annals of Mathematics and Artificial Intelligence, 12 (1994) 1–24.

[13] T. Przymusinski, Stationary semantics for disjunctive logic programs and deductive databases, in *Logic Programming: Proc. 1990 North American Conference*, S. Debray ed., MIT Press, 1990, pp. 40–62.

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

[15] K. Ross, The well-founded semantics for disjunctive logic programs, in *Deductive and Object-Oriented Databases: Proc. First Intl Conference (DOOD '89)*, Kyoto Research Park, Kyoto, Japan, W. Kim, J.-M. Nicolas, and S. Nishio eds., Elsevier, 1990, pp. 385–402.

[16] D. Saccà, C. Zaniolo, Stable models and nondeterminism in logic programs with negation, in *Proc. Ninth ACM Symposium on Principles of Database Systems (PODS '90)*, Nashville, Tennessee (USA), ACM, 1990, pp. 205–218.

[17] D. Saccà, C. Zaniolo, Partial models and three-valued models in logic programs with negation, in *Proc. 1st Intl Workshop on Logic Programming and Nonmonotonic Reasoning*, Washington DC, A. Nerode, W. Marek, and V.S. Subrahmanian eds., MIT Press, 1991, pp. 87–104.

[18] J.D. Ullman, *Principles of Database and Knowledge-Base Systems*, Computer Science Press, Rockville, Maryland (USA), 1989.

[19] A. Van Gelder, The alternating fixpoint of logic programs with negation, in *Proc. Eighth ACM Symposium on Principles of Database Systems (PODS '89)*, Philadelphia, Pennsylvania (USA), ACM, 1989, pp. 1–10.

[20] A. Van Gelder, K. Ross, J. Schlipf, The well-founded semantics for general logic programs, Journal of the ACM, 38:3 (1991) 620–650.

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

Computing Disjunctive Ordered Logic*

F. Buccafurri
ISI – CNR
c/o DEIS Univ. della Calabria
87030 Rende, Italy
bucca@si.deis.unical.it

N. Leone
Information Systems Dep.
Technical University of Vienna
A – 1040 Vienna, Austria
leone@dbai.tuwien.ac.at

P. Rullo
DIMET
Univ. di Reggio Calabria
89100 ReggioCal., Italy
rullo@si.deis.unical.it

Abstract

Disjunctive Ordered Logic ($\mathcal{DOL}$, for short) is an extension of disjunctive logic programming with inheritance and true negation. The semantics of $\mathcal{DOL}$, recently proposed in the literature, is model theoretic and based on the notion of stable model.

In this paper, the definition of unfounded set is first extended to $\mathcal{DOL}$ programs. Then, an elegant characterization of stable models in terms of unfounded sets is provided. Finally, the results are exploited to provide an effective strategy for the computation of the stable model semantics of $\mathcal{DOL}$.

1 Introduction

Ordered logic ($\mathcal{OL}$) [5, 6, 7, 2] is a formalism extending Datalog with inheritance and true negation. An $\mathcal{OL}$ program is, indeed, a partially ordered set of components, where components lying lower in the hierarchy denote more trustable beliefs than higher ones. Moreover, explicit negation is allowed in the head of the rules (elsewhere called *true negation*), hereby obtaining the management of exceptions and defeasible reasoning.

Disjunctive Ordered Logic ($\mathcal{DOL}$, for short), first presented in [3], is an extension of $\mathcal{OL}$ with disjuntion. A main difficulty with designing $\mathcal{DOL}$ is the definition of its semantics. Indeed, the interactions between components (in the form of belief defeating) is to be restated with respect to the context of the more simple $\mathcal{OL}$, to take into account the presence of disjuncts in rule heads. In [3] is defined a stable models semantics for $\mathcal{DOL}$, and shown that it naturally extends stable model semantics to the context of disjunctive ordered logic.

This paper concerns characterizations and computation of $\mathcal{DOL}$. In particular, the contribution of the paper is two-fold.

*This work has been partially supported by *Cristian Doppler Lab. for Expert Systems, Istituto per la Sistemistica e l'Informatica ISI-CNR*, the EC-US033 project "DEUS EX MACHINA: Non-Determinism in Deductive Databases" and by a MURST grant (40% share) under the project "Sistemi formali e strumenti per basi di dati evolute."

- We first introduce a notion of *unfounded set* which generalizes both the definition given for traditional logic programming in [12, 11] and that presented for disjunctive logic programming in [8]. Using this notion, we give an elegant characterization of stable models in terms of unfounded sets; in particular, we demonstrate that a model is a stable model if and only if it is unfounded-free.

- Then, we exploit the presented results to provide an effective strategy for the computation of the stable model semantics of $\mathcal{DOL}$ programs.

For lack of space, we have suppressed the proofs of the results. The complete proofs as well as additional details on the language can be found in the technical report [4] which is available on email request to the authors.

2 The $\mathcal{DOL}$ Language

2.1 $\mathcal{DOL}$ by Examples

In this section we present some examples aimed at showing the main peculiarities of $\mathcal{DOL}$ for knowledge representation and reasoning.

Example 1 We want to describe the following situation (this example is a modified version of one appearing in [1]): (1) normally, a person can use both his arms and is right-handed; moreover, a person can write if he is right-handed (resp., left-handed) and his right (resp., left) arm is ok, and (2) you have seen *max* and you remember that he has one of his arms broken, but you do not remember which one; furthemore, you know that *max* is left-handed.

The above described knowledge is encoded in the $\mathcal{DOL}$ program $\mathcal{P}_{max}$ of Figure 1. Here, default knowledge (point 1. above) is represented in the higher component c_1, whose rules express conditions that normally hold. The exceptions to the default knowledge (that concern *max*) are stated in the lowest component c_2. Clearly, knowledge described in the higher component in the hierarchy flows down into the lower one. Now the question is: "Can *max* write or not?". Since *max* is left-handed (rule r_7) he can write provided that his left arm is OK (see rule r_5). Unfortunately, because of the uncertainty deriving from the incomplete knowledge about the state of *max*'s arms (rule r_6), we cannot definitely provide a positive or negative answer for the above question. Rather, two *possible* answers can be constructed: (a) "If max has his left arm broken, then he cannot write", and (b) "If max has his right arm broken, then he can write". It will become apparent later in this paper that the stable model semantics correctly represents the above situation. Indeed, according to the $\mathcal{DOL}$ semantics as defined in Section 2.3 below, $\mathcal{P}_{max}$ has two stable models, namely, $M_1 = \{\neg right\_handed(max), \neg right\_arm\_ok(max), left\_arm\_ok(max), can\_write(max)\}$, and $M_2 = \{\neg right\_handed(max), right\_arm\_ok(max), \neg left\_arm\_ok(max)\}$. Thus, each stable model of $\mathcal{P}_{max}$ represents one of the possible scenarios.

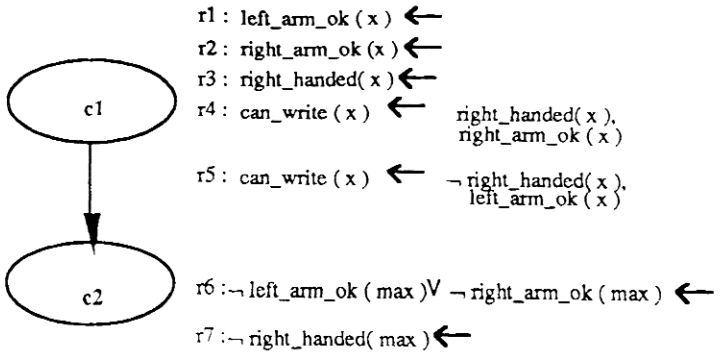

Figure 1: the $\mathcal{P}_{max}$

Example 2 We next use $\mathcal{DOL}$ to model the reasoning process of a doctor that has to formulate a diagnosis in presence of a symptom called *acute exordium cephalgia*. The $\mathcal{DOL}$ program that we are going to present is incremental to simulate the stepwise refinement of the diagnostic process. In particular, every new information acquired by the doctor (e.g., by a test) is represented by the addition of a new component (describing the new information) to the program.

At first, the doctor considers all possible diagnostic hypothesis which are compatible with the given symptom. For the sake of simplicity, we confine ourselves to consider only three of such hypotesis, namely, 1) *chephalgia due to a trauma* (denoted by T), 2) *chephalgia due to an infectious disease* (denoted by ID), 3) *chephalgia due to a tumor* (denoted by Tu). Under this simplification, the $\mathcal{DOL}$ program modeling this first reasoning step is the program $\mathcal{P}_0$ of figure 2 whose stable models are $M_1 = \{chephalgia, T\}$, $M_2 = \{chephalgia, ID\}$ and $M_3 = \{chephalgia, Tu\}$. Each stable model clearly represents a possible diagnosis.

Now, suppose that, on the basis of the case-history of the patient, the doctor can exclude that a trauma (hypothesis T) has occurred, i.e., $\neg T$ is true. As a consequence, only hypotesis 2 and 3 above (i.e., infectious disease and tumor, respectively) still hold. The $\mathcal{DOL}$ program representing this new situation is program $\mathcal{P}_1$ depicted in figure 2.b (here, a lower component, consisting of rule $\neg T \leftarrow$, has been added to $\mathcal{P}_0$). Indeed, the stable models of $\mathcal{P}_1$ are $M_4 = \{chephalgia, ID, \neg T\}$ and $M_5 = \{chephalgia, Tu, \neg T\}$, which capture the intended semantics of the program.

As a further step, the doctor performs the *neurological examination*. We suppose that a paralisys of the oculomotor nerve is detected. Then, the doctor assumes the existence of a tumor, as paralysis could be caused by compression. The program modeling this reasoning level is $\mathcal{P}_2$ (see 2).b It is obtained from $\mathcal{P}_1$ by adding a lower component whose definition consists of the fact ($Paralisys \leftarrow$), representing the result of the neurological examination, and the rule $Tu \leftarrow Paralisys$ representing the conclusion to which the doctor is induced. The stable model semantics for $\mathcal{P}_2$ is given by the unique stable model $M_6 = \{chephalgia, Tu, \neg T, Paralisys\}$, which correctly captures the knowledge of the doctor corresponding to the current reasoning step.

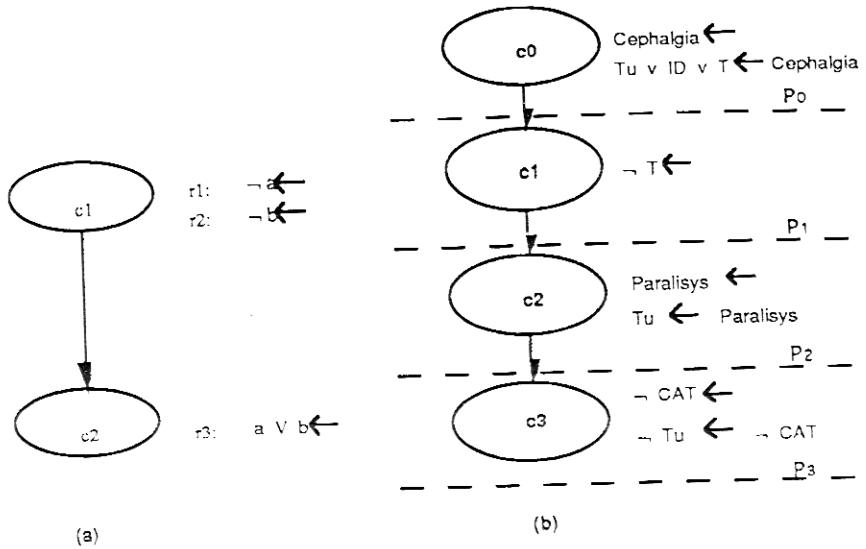

Figure 2: (a) the program $\mathcal{P}_1$; (b) the programs of example 2

Indeed, this knowledge can be summarized as follows: (a) there is no trauma (and $\neg T$ is true in M_6); (b) there is a tumor whose existence has been infered by the presence of a paralisys (and Tu and *paralisys* are both in M_6) and (c) no information on infectious disease ID is available (and neither ID nor $\neg ID$ are in M_6).

However, the lack of complete information induces the doctor to find a further confirmation to the above conclusion. To this end, he executes the Computerized Axial Tomography (CAT). Surprisingly, the result of this test shows the absence of tumor. As a consequence, infectious disease becomes the only possible diagnosys. Program $\mathcal{P}_3$ in figure 2.b correctly models this situation as its unique stable model (according with the definition provided in the next subsection) is $M_7 = \{cephalgia, ID, \neg Tu, \neg T, paralisys, \neg CAT\}$ which represents the final diagnosys. This conclusion can be actually explained as a particular infectious disease (encephalitis) that, as sometimes happens, determined the paralisys of the oculomotor nerve.

2.2 $\mathcal{DOL}$ Programs

A *term* is either a constant or a variable (note that function terms are not considered). 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. 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 *rule* r is a clause of the form $H \leftarrow B$, where H (*head* of the rule) is a disjunction $H_1 \vee ... \vee H_n$ of literals and B (*body* of the rule) is a conjunction $B_1 \wedge ... \wedge B_m$ of literals (note that negative literals may occur in the head of a rule). Given a rule r, we shall denote by $H(r)$ and $B(r)$ the set of literals in the head and in the body of r, respectively.

A term, an atom, a literal or a rule is *ground* if no variable appears in it.

Let $(C, \leq)$ be a finite partially ordered set of symbols, called *identifiers*. A *component* is a pair $(c, D(c))$, where $c \in C$ and $D(c)$, the *definition of* c, is a finite set of rules.

A $\mathcal{DOL}$ *program* (on C) is a set of components, one for each element of C. In the following we shall denote by $<$ the reflexive-reduction of $\leq$ (i.e., $a < b$ iff $a \leq b$ and $a \neq b$).

2.3 Stable Model Semantics

In this section we define the stable models semantics for disjunctive ordered logic.

The *Universe* $U_{\mathcal{P}}$ of $\mathcal{P}$ is the set of all constants appearing in the rules of the components of $\mathcal{P}$. The *Base* $B_{\mathcal{P}}$ of $\mathcal{P}$ is the set of all possible ground literals constructible from the predicates appearing in the rules of $\mathcal{P}$ and the constants occurring in $U_{\mathcal{P}}$ (clearly, both $U_{\mathcal{P}}$ and $B_{\mathcal{P}}$ are finite). Notice that, unlike disjunctive Datalog, the Base of a $\mathcal{DOL}$ program contains also negative literals (as both negative and positive literals are treated in a uniform way).

Given a rule r occurring in a component of $\mathcal{P}$, 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 constants in $U_{\mathcal{P}}$. We denote by $ground(\mathcal{P})$ the (finite) multiset of the ground instances of the rules occurring in the components of $\mathcal{P}$. We can define a function $compOf$ from $ground(\mathcal{P})$ onto the set C of the identifiers, associating with a ground instance $\overline{r}$ of r the (unique) component of r.

Two ground rules r_1 and r_2 are *conflicting on* L if $L \in H(r_1)$ and $\neg.L \in H(r_2)$.

An *interpretation* I for a $\mathcal{DOL}$ progam $\mathcal{P}$ is a subset of $B_{\mathcal{P}}$ such that $I \cap \neg.I = \emptyset$.

Given an interpretation I, a ground literal L is *true* (resp., *false*) in I if $L \in I$ (resp., $L \in \neg.I$). A literal $L \in B_{\mathcal{P}}$ which is neither true nor false w.r.t. I is said to be *undefined* in I.

The head of a rule r is *true* in the interpretation I is $H(r) \cap I \neq \emptyset$; the body of r is *true* in I if $B(r) \subseteq I$. A rule $r \in ground(\mathcal{P})$ is *satisfied* in I if either the head is true in I or the body of r is not true in I.

A rule r such that $B(r)$ is true in I and, further, some head literal, say, L, is in I is said to be *applied in* I *w.r.t.* L.

Next we introduce the concept of *model*. The notion of satisfiability of a rule is not sufficient, as it does not take into account the possible presence of explicit contradictions. Hence, we have to introduce the following preliminary definitions.

Definition 1 Let I be an interpretation. Let $r1$ and r_2 be rules in $ground(\mathcal{P})$. We say that r_1 *defeates* r_2 *on the literal* L *in* I if: 1) $r1$ and r_2 are conflicting rules on L and $\neg.L \in H(r_1)$, 2) $compOf(r_2) < compOf(r_1)$ does not hold, 3) $\neg.L \in I$ and $B(r_1) \subseteq I$ (i.e., r_1 is applied in I).

A rule $r \in ground(\mathcal{P})$ is *defeated in* I, if for each $L \in H(r)$, there exists $r_1 \in ground(\mathcal{P})$ such that r_1 defeates r on L in I. □

Intuitively, defeating defines the possible ways to solve conflicts deriving from the presence in the program of contradictory pieces of information. An example follows.

Example 3 Consider the program $\mathcal{P}_1$, of Figure 2.a, consisting of the components c_1 and c_2, with $c_1 < c_2$. Here, it should be noticed that rule r_3 is conflicting with r_1 on a and with r_2 on b, and that r_3 is more specific (in the inheritance hierarchy) than both r_1 and r_2. Consider now for $\mathcal{P}_1$ the interpretation $I = \{a, b\}$. Note that: (i) r_3 is both applied w.r.t. a and w.r.t. b in I, (ii) r_3 defeats at the same time r_1 and r_2 (respectively, on $\neg a$ and on $\neg b$). □

Definition 2 Let I be an interpretation for $\mathcal{P}$. A rule $r \in ground(\mathcal{P})$ is *effective* w.r.t. I if it is not defeated in I and, further, $B(r) \subseteq I$. □

Example 4 Let $I = \{a, b\}$ be an interpretation for $\mathcal{P}_1$ of figure 2.a. Rule r_3 is effective w.r.t. I (as it is not defeated in I and its body is true in I). □

We are now ready to provide the definition of model.

Definition 3 Let I be an interpretation for $\mathcal{P}$. We say that I is a *model* for $\mathcal{P}$ if, for each effective rule $r \in ground(\mathcal{P})$ w.r.t. I, r is satisfied in I. □

Example 5 The interpretation $I = \{\neg a, b\}$ is a model for the program $\mathcal{P}_1$ of figure 2.a. Indeed, the effective rules w.r.t. I are r_1 and r_3, both of which are satisfied in I. The interpretation $I = \{chephalgia, Tu, ID, \neg T\}$ is a model for the program $\mathcal{P}_1$ of figure 2.b. Indeed, all the rules of the program are both effective w.r.t. I and satisfied in I. □

Definition 4 Let $\mathcal{P}$ be a $\mathcal{DOL}$ program. A *knowledge-minimal model* (or, simply, "minimal model") M for $\mathcal{P}$ is a model for which there exists no other model N such that $N \subset M$. □

Now, we introduce the notion of stable model for a $\mathcal{DOL}$ program. A characterization of stable models in terms of unfounded sets is also given.

Definition 5 Let I be an interpretation for $\mathcal{P}$. The *reduction of* $\mathcal{P}$ *w.r.t.* I is $P^I = \{r \in ground(\mathcal{P}) \mid r \text{ is effective w.r.t. } I\}$. □

We note that the reduction of a program is simply a set of ground rules. Given a set S of ground rules, we denote by $H(S)$ the positive disjunctive program (called the *positive version* of S), obtained from S by considering each negative literal $\neg q(\overline{X})$ as a positive one with predicate symbol $\neg q$. Moreover, we denote by $\mathcal{MM}(H(S))$ the set of minimal models [10] of the program $H(S)$ (note that the minimal model semantics is applicable since $H(S)$ is a positive program).

Definition 6 Let M be a model for a program $\mathcal{P}$. We say that M is a $\mathcal{DOL}$ *stable model* (or simply "stable model") of $\mathcal{P}$ if $M \in \mathcal{MM}(H(P^M))$ (i.e., M is a minimal model of $H(P^M)$) □

Example 6 Consider the model $M = \{\neg a, b\}$ of the program $\mathcal{P}_1$ in figure 2.a. Now, the reduction of $\mathcal{P}_1$ w.r.t. I is $\mathcal{P}_1^M = \{r_1 : \neg a \leftarrow,\ r_3 : a \vee b \leftarrow\}$ as r_1 and r_3 are the only effective rules of $\mathcal{P}_1$ w.r.t. M. The positive version of $\mathcal{P}_1^M$ has the minimal model the following minimal models [10]: $\{a, \neg a\}$ and $\{\neg a, b\}$. Thus, M is a stable model. Similarly, it can be shown that also $\{a, \neg b\}$ is a stable model for $\mathcal{P}_1$.

Consider now the model $M = \{chephalgia, Tu, \neg T\}$ for the program $\mathcal{P}_1$ of figure 2.b. It easy to see that M is a minimal model of the positive version of the reduction of $\mathcal{P}_1$ w.r.t. M. (Note that all the rules of $\mathcal{P}_1$ are effective w.r.t. M). Hence, M is a stable model for the program $\mathcal{P}_1$. Similarly, it easy to show that $M_1 = \{chephalgia, ID, \neg T\}$ is a stable model for $\mathcal{P}_1$. □

2.4 A Declarative Characterization of Stable Models

In this section we first introduce the notion of *unfounded set* for a $\mathcal{DOL}$ program and give some properties of unfounded sets; then, we provide a declarative characterization of stable models in terms of unfounded sets.

The notion of unfounded set, presented next, generalizes both the analog notions given in [12] for classical (i.e. disjunction-free) logic programming, and in [8] for disjunctive logic programming.

Definition 7 Let I be an interpretation for $\mathcal{P}$. A set $X \subseteq B_{\mathcal{P}}$ of ground atoms is an *unfounded set* for $\mathcal{P}$ w.r.t. I if, for each $L \in X$, for each rule $r \in ground(\mathcal{P})$ such that $L \in H(r)$, at least one of the following conditions holds: 1) $B(r) \cap \neg.I \neq \emptyset$, i.e. $B(r)$ is false w.r.t. I, or 2) $B(r) \cap X \neq \emptyset$, i.e. some literal in $B(r)$ is in X, or 3) $(H(r) - X) \cap I \neq \emptyset$, i.e., a literal in the head of r, distinct from L and other elements of X, is in I, or 4) r is defeated in I. □

Unlike traditional (disjunction-free) logic programming, the union of two unfounded sets is not necessarily an unfounded set.

Example 7 Consider the program $\mathcal{P}$ consisting of one only component containing the rule $a \vee b \leftarrow$ and let $I = \{a, b\}$ be an interpretation for it. Clearly, both $\{a\}$ and $\{b\}$ are unfounded sets for $\mathcal{P}$ w.r.t. J, whereas the union $\{a, b\}$ is not. □

Thus, in general, we have a number of maximal unfounded sets for a fixed program w.r.t. a given interpretation. Given a program $\mathcal{P}$ and an interpretation I for it, we denote by $MUS_{\mathcal{P}}(I)$ the set of all maximal unfounded sets for $\mathcal{P}$ w.r.t. I. If $MUS_{\mathcal{P}}(I)$ is a singleton (i.e., there exists a unique maximal unfounded set for $\mathcal{P}$ w.r.t. I) we call its unique element the *greatest unfounded set* of $\mathcal{P}$ w.r.t. I. We denote by $\mathbf{I}_{\mathcal{P}}$ the domain of all the interpretations such that $MUS_{\mathcal{P}}(I)$ is a singleton.

Definition 8 Let I be an interpretation for $\mathcal{P}$ such that $MUS_{\mathcal{P}}(I)$ is a singleton. The *greatest unfounded set* of $\mathcal{P}$ w.rt. I, denoted by $GUS_{\mathcal{P}}(I)$, is the (unique) element of $MUS_{\mathcal{P}}(I)$. □

Next we individuate some cases where there exists the greatest unfounded set. Clearly each disjunction-free $\mathcal{DOL}$ program, i.e. each ordered logic program, has the greatest unfounded set w.r.t. any interpretation. Indeed, in this case, it is easy to recognize that definition 7 coincides with the definition of unfounded set for ordered logic programs given in [2]. Now, we show that, for a given $\mathcal{DOL}$ program there exists a class of interpretations, namely *unfounded-free interpretations*, w.r.t. which $\mathcal{P}$ has the greatest unfounded set (thus, this class of interpretations is contained in $\mathbf{I}_\mathcal{P}$).

Definition 9 Let I be an interpretation for $\mathcal{P}$. I is *unfounded-free* (for $\mathcal{P}$) if $I \cap X = \emptyset$, $\forall X \in MUS_\mathcal{P}(I)$. □

From the above definition it follows that all subsets of an unfounded-free interpretation are unfounded-free as well.

Proposition 1 *Let I be an unfounded-free interpretation for the program $\mathcal{P}$, and J be an interpretation such that $J \subseteq I$. Then, J is unfounded-free.*

Now we prove that, for an unfounded-free interpretation I, the program $\mathcal{P}$ has $GUS_\mathcal{P}(I)$.

Proposition 2 *Let I be an unfounded-free interpretation for $\mathcal{P}$. Then, $\mathcal{P}$ has the greatest unfounded set $GUS_\mathcal{P}(I)$.*

The next result shows the monotonicity of $GUS_\mathcal{P}$.

Proposition 3 *Let I and J be two unfounded-free interpretations such that $I \subseteq J$. Then, $GUS_\mathcal{P}(I) \subseteq GUS_\mathcal{P}(J)$.*

Now we provide a number of characterizations of models in terms of unfounded sets.

Proposition 4 *Let M be an interpretation for $\mathcal{P}$. Then M is a model for $\mathcal{P}$ if and only if $B_\mathcal{P} - M$ is an unfounded set for $\mathcal{P}$ w.r.t. M.*

We next show that the set of stable models coincides with that of unfounded-free models.

Theorem 1 *Let M be a model for a $\mathcal{DOL}$ program $\mathcal{P}$. M is a stable model for $\mathcal{P}$ if and only if is unfounded-free.*

3 The Computation of $\mathcal{DOL}$

Next we provide a constructive definition of stable models by a non-deterministic operator $V_\mathcal{P}$. Then, we give a concrete algorithm implementing $V_\mathcal{P}$ through a backtracking strategy. Informally, we first compute a set of ground literals which is contained in every stable model, by evaluating the least fixpoint $W_\mathcal{P}^\infty(\emptyset)$ of a suitable operator $W_\mathcal{P}$. This part of the computation is deterministic and efficient ($W_\mathcal{P}^\infty(\emptyset)$ is polynomial-time computable). Then, the main problem is "how" to move beyond $W_\mathcal{P}^\infty(\emptyset)$ toward the stable models of $\mathcal{P}$, as in principle all possible extension of $W_\mathcal{P}^\infty(\emptyset)$ are candidated to possibly be stable models. To restrict the search space, we observe that only the literals which satisfy some conditions can lead to the computation of stable models. We call these literals *possibly-true literals*. Thus, we continue the computation from $W_\mathcal{P}^\infty(\emptyset)$ by assuming true a possibly-true literal and deriving all the deterministic consequences of this assumption. This process is iterated until either a stable model or a contradiction is found. In the latter case, we backtrack to the last choice point and do another choice (that is, we remove the last assumption and assume the truth of another possibly-true literal). That is, backtracking is used to simulate non determinism.

We next provide the definition of the deterministic immediate consequence operator for $\mathcal{DOL}$ programs. This operator is an extension of that defined for three-valued interpretations of traditional logic programming.

Definition 10 Let $\mathcal{P}$ be a program and $\mathbf{J}$ be the set of interpretations for $\mathcal{P}$. The *immediate consequence operator* is defined as follows:

$$T_\mathcal{P} : \mathbf{J} \rightarrow 2^{B_\mathcal{P}}$$

$$T_\mathcal{P}(I) = \{L \in B_\mathcal{P} \; s.t. \; \exists r \in ground(\mathcal{P}), \; L \in H(r), H(r) - \{L\} \subseteq \neg.I, B(r) \subseteq I\}$$

□

Based on the above operator and on the notion of greatest unfounded set, we define the operator $W_\mathcal{P}$ as follows:

Definition 11 Given a program $\mathcal{P}$, we define the operator $W_\mathcal{P}$ as follows:

$$W_\mathcal{P} : \mathbf{I}_\mathcal{P} \rightarrow 2^{B_\mathcal{P}}$$

$$W_\mathcal{P}(I) = T_\mathcal{P}(I) \cup \neg GUS_\mathcal{P}(I)$$

□

Proposition 5 *Given a program $\mathcal{P}$, let $\{W_n\}_{n \in \mathcal{N}}$ be the sequence whose n-th term is the n-fold application of the $W_\mathcal{P}$ operator starting from the empty set (i.e., $W_0 = \emptyset$, $W_n = W_\mathcal{P}(W_{n-1})$). Then: a) $\{W_n\}_{n \in \mathcal{N}}$ converges to a limit $W_\mathcal{P}^\infty(\emptyset)$ and b) for each stable model M for $\mathcal{P}$, $W_\mathcal{P}^\infty(\emptyset) \subseteq M$ holds.*

Thus, by repeatedly applying $W_\mathcal{P}$ from the empty set, the least fixpoint $W_\mathcal{P}^\infty(\emptyset)$ is (finitely) reached and $W_\mathcal{P}^\infty(\emptyset)$ is contained in any stable model. Now the point is how to move beyond $W_\mathcal{P}^\infty(\emptyset)$, in the direction of stable models. To this end, we introduce the notion of *possibly-true fact* for $\mathcal{DOL}$ programs. A similar notion has previously been introduced in different contexts in [2, 8]

Definition 12 Given an interpretation I for $\mathcal{P}$, an undefined literal $L \in \overline{I}$ belongs to the set of *possibly true fact w.r.t. I*, denoted by $PT_\mathcal{P}(I)$, if there exists a rule $r \in ground(\mathcal{P})$ such that $L \in H(r)$, $B(r) \subset I$ and $H(r) \cap I = \emptyset$. Clearly, $PT_\mathcal{P}(I) \subseteq \overline{I}$
□

The usefulness of possibly-true literals in our computational model is pointed out by the following proposition, according to which, if an interpretation I is (strictly) contained in some stable model M, then there exists a possibly-true literal in $PT_{\mathcal{P}}(M)$ which belongs to M.

Proposition 6 *Let M be a stable model for a program $\mathcal{P}$. If $I \subset M$ then, $PT_{\mathcal{P}}(I) \cap M \neq \emptyset$.*

Thus, we can take from $PT_{\mathcal{P}}(W^{\infty}_{\mathcal{P}}(\emptyset))$ a possibly-true literal, say L, belonging to some stable model M. Clearly, $W^{\infty}_{\mathcal{P}}(\emptyset) \cup \{L\} \subseteq M$ holds. Now, if I is an interpretation contained in a stable model, say M, it easy to prove that $\overline{T_{\mathcal{P}}}(I) \subseteq M$ holds, where $\overline{T_{\mathcal{P}}}$ is the skeptical immediate consequence operator (see below). On the basis of this result, we can then iteratively apply $\overline{T_{\mathcal{P}}}$ until eventually a stable model is reached.

The formalization of the process described above requires some preliminary definitions.

Definition 13 Let I be an interpretation for a program $\mathcal{P}$. A rule $r \in ground(\mathcal{P})$ is *defeasible in the literal $L \in H(r)$ w.r.t. I* if there exists a rule $r' \in ground(\mathcal{P})$ such that $compOf(r') > compOf(r)$ does not hold, $\neg.L \in H(r')$, and $B(r') \cap \neg.I = \emptyset$ □

Definition 14 Let $\mathcal{P}$ be a program and $\mathbf{J}$ the set of interpretations for $\mathcal{P}$. The *skeptical immediate consequence operator*

$$\overline{T_{\mathcal{P}}} \ : \ \mathbf{J} \rightarrow 2^{B_{\mathcal{P}}}$$

is defined as follows: given an interpretation $I \in \mathbf{J}$, a literal $L \in B_{\mathcal{P}}$ belongs to $\overline{T_{\mathcal{P}}}(I)$ iff there exists $r \in ground(\mathcal{P})$ such tht $L \in H(r)$, $H(r) - \{L\} \subseteq \neg.I$, $B(r) \subseteq I$ and, further, r is not defeasible in L w.r.t. I. □

Proposition 7 *Let M be a stable model for a program $\mathcal{P}$ and I be an interpretation such that $I \subseteq M$. Then $\overline{T_{\mathcal{P}}}(I) \subseteq M$ holds.*

We are now in a position to define the nondeterministic transformation $V_{\mathcal{P}}$ which will allow us to compute the stable models.

Definition 15 Let $\mathcal{P}$ be a $\mathcal{DOL}$ program. Define the nondeterministic transformation $V_{\mathcal{P}}$ as follows:

$$V_{\mathcal{P}} : 2^{B_{\mathcal{P}}} \rightarrow 2^{2^{B_{\mathcal{P}}}}$$

$$V_{\mathcal{P}}(I) = \begin{cases} \{I \cup \overline{T_{\mathcal{P}}}(I)\} & \text{if } \overline{T_{\mathcal{P}}}(I) \cup I \neq I \\ \{I \cup \{A\} \mid A \in PT_{\mathcal{P}}(I)\} & \text{otherwise} \end{cases}$$ □

Thus, if $I \neq \overline{T_{\mathcal{P}}}(I) \cup I$, then $V_{\mathcal{P}}(I)$ is the singleton $\{\overline{T_{\mathcal{P}}}(I) \cup I\}$; otherwise, $V_{\mathcal{P}}(I)$ is the family of sets of ground literals $I \cup \{A\}$, one for each possibly-true fact $A \in PT_{\mathcal{P}}(I)$.

Definition 16 A *deterministic computation of $\mathcal{P}$*, denoted $C_{\mathcal{P}}$, is a sequence of sets of ground literals inductively defined as follows: $V^0_{C_{\mathcal{P}}} = W^{\infty}_{\mathcal{P}}(\emptyset), ..., V^i_{C_{\mathcal{P}}} = J$, $i > 0$, where $J \in V_{\mathcal{P}}(V^{i-1}_{C_{\mathcal{P}}})$. The set $\mathcal{F}_{\mathcal{P}}$ of all deterministic computations is the *nondeterministic computation* of $\mathcal{P}$. □

Let $\mathcal{F}_{\mathcal{P}}$ be the nondeterministic computation of $\mathcal{P}$ and $C_{\mathcal{P}}$ any element of it. It can be easily recognized that $C_{\mathcal{P}}$ is monotonically increasing. Since the Base of $\mathcal{P}$ is finite, $C_{\mathcal{P}}$ is upper bounded; so that there exists a natural k such that $V^j_{C_{\mathcal{P}}} = V^k_{C_{\mathcal{P}}}$, for each $j \geq k$. We denote $V^k_{C_{\mathcal{P}}}$ by $V^{\infty}_{C_{\mathcal{P}}}$. Note that $V^{\infty}_{C_{\mathcal{P}}}$ is such that $V_{\mathcal{P}}(V^{\infty}_{C_{\mathcal{P}}}) = \{V^{\infty}_{C_{\mathcal{P}}}\}$ (see definition 15). We further remark that there exists a natural $i \geq 0$ such that (1) $V^i_{C_{\mathcal{P}}} = W^{\infty}_{\mathcal{P}}(\emptyset)$, for each $C_{\mathcal{P}} \in \mathcal{F}_{\mathcal{P}}$ and (2) for any pair of deterministic computations $C_{\mathcal{P}}$ and $C'_{\mathcal{P}}$ in $\mathcal{F}_{\mathcal{P}}$, $V^j_{C_{\mathcal{P}}} = V^j_{C'_{\mathcal{P}}}$, for each $0 \leq j \leq i$.

Definition 17 Let $C_{\mathcal{P}}$ be a deterministic computation. If $V^{\infty}_{C_{\mathcal{P}}}$ is an unfounded-free interpretation and $\neg.V^{\infty}_{C_{\mathcal{P}}} \subseteq GUS_{\mathcal{P}}(V^{\infty}_{C_{\mathcal{P}}})$, then $C_{\mathcal{P}}$ is a *successful computation*, otherwise it is a *failed computation*. □

Next we show that, if M is a stable model for $\mathcal{P}$ then there exists a successful computation $C_{\mathcal{P}}$ such that $M = V^{\infty}_{C_{\mathcal{P}}}$, and vice versa. That is, the set of stable models coincides with the set of all $V^{\infty}_{C_{\mathcal{P}}}$ such that $C_{\mathcal{P}}$ is a successful computation.

We next provide a proposition stating that a succesful computation correctly computes a stable model.

Theorem 2 *Let $C_{\mathcal{P}}$ be a successful computation. Then $V^{\infty}_{C_{\mathcal{P}}}$ is a stable model for $\mathcal{P}$.*

We next show that if $V^{\infty}_{C_{\mathcal{P}}} = W^{\infty}_{\mathcal{P}}(\emptyset)$ then $V^{\infty}_{C_{\mathcal{P}}}$ is the unique stable model for $\mathcal{P}$.

Corollary 1 *Let $C_{\mathcal{P}}$ be a successful computation. If $V^{\infty}_{C_{\mathcal{P}}} = W^{\infty}_{\mathcal{P}}(\emptyset)$ then $V^{\infty}_{C_{\mathcal{P}}}$ is the unique stable model for $\mathcal{P}$.*

Next we prove the converse of theorem 2, that is, if M is a stable model then there exists a successful computation $C_{\mathcal{P}}$ such that $M = V^{\infty}_{C_{\mathcal{P}}}$. To this end, we use the result in proposition 6 stating that, if an interpretation I is (properly) contained in a stable model M, then there exists some possibly-true fact in $PT_{\mathcal{P}}(I)$ which belongs to M (note that, as a consequence, $I \subset M \Rightarrow PT_{\mathcal{P}}(I) \neq \emptyset$). Further, by proposition 7, we know that if I is an interpretation such that $I \subseteq M$, then $\overline{T_{\mathcal{P}}}(I) \subseteq M$. Our computation strategy is based on the above results. We informally describe how a successful computation determining M can be built starting from $W^{\infty}_{\mathcal{P}}(\emptyset)$ (recall that, by virtue of proposition 5, $W^{\infty}_{\mathcal{P}}(\emptyset) \subseteq M$). In a given step of the computation, either we apply the operator $\overline{T_{\mathcal{P}}}$, adding only literals of M (by virtue of proposition 7), or choose a possibly true facts belonging to M (this is always possible by virtue of proposition 6). More formally,

Theorem 3 *Let M be a stable model for a $\mathcal{DOL}$ program $\mathcal{P}$. Then $M = V^{\infty}_{C_{\mathcal{P}}}$, for some successful computation $C_{\mathcal{P}}$.*

Thus, theorems 2 and 3 provide the ground for the computation of the stable model semantics of disjunctive ordered logic programs. Indeed, they ensure the soundness and the completeness, respectively, for our computation strategy. An effective algorithm implementing the above described computational strategy is presented in [4].

References

[1] Baral, C., Gelfond, M. Logic Programming and Knowledge Representation *Journal of Logic Programming*, Vol. 19/20, May/July 1994, pp. 73–148.

[2] Buccafurri, F., Leone, N., Rullo, P., Stable Models and their Computation for Logic Programming with Inheritance and True Negation, to appear in *Journal of Logic Programming*, Vol 27(1) april 96, pp 5-43.

[3] Buccafurri, F., Leone, N., Rullo, P., Palopoli, L., Disjunctive Ordered Logic, *Proc. of the Int. Conf. DEXA '95*, pp. 303-312, London, 4-8 September 1995.

[4] Buccafurri, F., Leone, N., Rullo, P., Semantics and Computation of Disjunctive Ordered Logic, *Tec.Report ISI-CNR N. 95-07*.

[5] Laenens, E., Saccá, D., and Vermeir, D., Extending Logic Programming, *Proc. of ACM SIGMOD*, May 1990.

[6] Laenens, E., Vermeir, D., A Fixpoint Semantics for Ordered Logic, *Journal of Logic and Computation*, vol.1, n.2, December, 1990, pp. 159-185.

[7] Leone, N., and Rossi, G., Well-founded Semantics and Stratification for Ordered Logic Programs, *New Generation Computing*, Vol. 12, N. 1, Springer-Verlag, November 1993, pp. 91-121.

[8] Leone, N., Rullo,P., Scarcello, F., Declarative and fixpoint characterizations of disjunctive stable models, in *Logic Programming: Proc. Twelth Intl Symposium (ILPS '95)*, Portland, Oregon (USA), J. Lloyd ed., MIT Press, 1995, pp. 399–413.

[9] Leone, N., Rullo,P., Scarcello, F., Disjunctive Stable Models: Unfounded Sets, Fixpoint Semantics and Computation, Technical Report TR 95-14 ISI-CNR, 1995.

[10] J. Minker. On Indefinite Data Bases and the Closed World Assumption. *Proceedings of the 6th Conference on Automated Deduction (CADE-82)*, 1982, pp. 292–308.

[11] Saccá, D., Zaniolo, C., Stable Models and Nondeterminism in Logic Programs with Negation, *Proc. ACM PODS*, 1990.

[12] Van Gelder, A., Ross, K. A., Schlipf, J. S., The Well-Founded Semantics for General Logic Programs, *Journal of the ACM*, 1990.