Unfolding Partiality and Disjunctions in Stable Model Semantics

Tomi Janhunen, Ilkka Niemelä, Patrik Simons
Lab. for Theoretical Computer Science
Dept. of Computer Science and Eng.
Helsinki University of Technology
P.O.Box 5400, 02015 HUT, Finland
{Tomi.Janhunen,Ilkka.Niemela,Patrik.Simons}@hut.fi

Jia-Huai You
Dept. of Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2H1
you@cs.ualberta.ca

Abstract

The paper studies an implementation methodology for partial and disjunctive stable models where partiality and disjunctions are unfolded from a logic program so that an implementation of stable models for normal (disjunction-free) programs can be used as the core inference engine. The unfolding is done in two separate steps. Firstly, it is shown that partial stable models can be captured by total stable models using a simple linear and modular program transformation. Hence, reasoning tasks concerning partial models can be solved using an implementation of total models. Disjunctive partial stable models have been lacking implementations which now become available as the translation handles also the disjunctive case. Secondly, it is shown how total stable models of disjunctive programs can be determined by computing stable models for normal programs. Hence, an implementation of stable models of normal programs can be used as a core engine for implementing disjunctive programs. The feasibility of the approach is demonstrated by constructing a system for computing stable models of disjunctive programs using the *smodels* system as the core engine. The performance of the resulting system is compared to that of *dlv* which is a state-of-the-art special purpose system for disjunctive programs.

1 INTRODUCTION

Implementation techniques for declarative semantics of logic programs have advanced considerably during the last years. For example, the XSB system (Sagonas, Swift and Warren 1996) is a WAM-based full logic programming system supporting the well-founded semantics. In addition to this kind of a skeptical approach that is based on query evaluation also a credulous approach focusing on computing models of logic programs is gaining popularity. This work has been centered around the stable model semantics (Gelfond and Lifschitz 1988). There are reasonably efficient implementations available for computing stable models for disjunctive and normal (disjunction-free) programs, e.g., *dlv* (Leone et al. 1999), *DeReS* (Cholewiński, Marek and Truszczyński 1996), and *smodels* (Niemelä and Simons 1997, Simons 1999b). Furthermore, there are interesting applications in areas such as planning (Dimopoulos, Nebel and Koehler 1997), model checking (Liu, Ramakrishnan and Smolka 1998), reachability analysis (Heljanko 1999), and constraint satisfaction (Niemelä 1999). This approach is emerging as a new paradigm for logic programming (Marek and Truszczyński 1999, Niemelä 1999).

This paper aims to widen the scope of the stable model paradigm. It addresses two issues: partial models and disjunctions. Sometimes it is natural to use partial models to represent a domain. Even when working with total models, partial models could be useful, e.g., for debugging purposes to show what is wrong in a program without any total models. However, little has been done on implementing partial model computation and most of the work has focused on query evaluation w.r.t. the well-founded semantics. In the paper we show that total stable models can capture partial stable models using a simple linear program transformation. This transformation works also in the disjunctive case showing that implementations of total stable models, e.g. *dlv*, can be used for computing partial stable models. Using a suitable transformation of queries, a mechanism for query answering can be realized as well.

Our translation is interesting in many respects. First, it should be noted that the translation does not follow directly from the complexity results already available. It has been shown, e.g., that the problem of deciding whether a query is contained in some model (possibility inference) is Σ_2^P-complete for both partial stable models and total stable models (Eiter and Gottlob 1995, Eiter, Leone and Saccà 1998). This implies that there exists a polynomial time reduction from possibility inference w.r.t. partial models to possibility inference w.r.t. total models. However, this kind of a translation is guaranteed to preserve only the yes/no answer to the possibility inference problem. To the best of our knowledge, there is no program transformation given in the literature which captures partial stable models in terms of stable models. Second, not all translations are sufficient from a computational point of view. In practice, when a program is compiled into another form to be executed, certain computational properties of the translation play an important role:

- efficiency of the compilation (in which order of polynomial),
- modularity (are independent, separate compilations of parts of a program possible), and
- structural preservation (are the composition and intuition of the original program preserved so that debugging and understanding of runtime behavior are made possible).

All this points to the importance of finding good translation methods to enable the use of an existing inference engine to solve other interesting problems.

The efficiency of procedures for computing stable models of normal programs has increased substantially in recent years. This raises the question whether such an implementation can be used as a *core engine* for implementing other reasoning systems. We study the feasibility of this approach by developing a method for computing stable models of disjunctive programs using such a core engine. This is non-trivial as deciding whether a disjunctive program has a stable model is Σ_2^p-complete whereas the problem is NP-complete in the non-disjunctive case. The method has been implemented using *smodels* as the core engine. The performance of the implementation is compared to that of *dlv* that is a state-of-the-art special purpose system for disjunctive programs.

There are a number of novelties in the work. Maximal partial stable models for normal programs are known as *regular models*, *M-stable models*, and *preferred extensions* (Dung 1995, Saccà and Zaniolo 1990, You and Yuan 1994). Although this semantics has a sound and complete top-down query answering procedure (Dung 1995, Eshghi and Kowalski 1989), so far very little effort has been given to a serious implementation. For disjunctive programs, to our knowledge, no implementation has ever been attempted. As a result, we obtain (perhaps) the first scalable implementation of the regular model/preferred extension semantics, and the first implementation ever for partial stable model semantics for disjunctive programs. Our technical work on the relationship between stable and partial stable models via a translational approach provides a compelling argument for the naturalness of partial stable models: stable models and partial stable models share the same notion of unfoundedness, carefully studied earlier in (Eiter, Leone and Saccà 1997, Leone, Rullo and Scarcello 1997). In fact, our work extends the previous understanding in the following way: stable and partial stable models capture the same semantics by different representing programs.

The rest of the paper is structured as follows. We first review the basic definitions and concepts and then show that partial models can be captured using a simple program transformation. In Section 4 we describe the method for computing disjunctive stable models using an implementation of non-disjunctive programs as a core engine. Then we present some experimental results and finish with some concluding remarks.

2 DEFINITIONS AND NOTATIONS

A *disjunctive logic program P* (or, just *disjunctive program P*) is a set of rules of the form

$$a_1 \vee \cdots \vee a_k \leftarrow b_1, \ldots, b_m, \sim c_1, \ldots, \sim c_n \qquad (1)$$

where $k \geq 1$, $m, n \geq 0$ and a_i's, b_i's and c_i's are atoms from the Herbrand base $\mathrm{Hb}(P)$[1] of P. Let us also distinguish subclasses of disjunctive programs. If $k = 1$ for each rule of P, then P is a *disjunction-free* or *normal program*. If $n = 0$ for each rule of P, then P is called *positive*.

Literals are either atoms from $\mathrm{Hb}(P)$ or expressions of the form $\sim a$ where $a \in \mathrm{Hb}(P)$. For a set of atoms $A \subseteq \mathrm{Hb}(P)$, we define $\sim A$ as $\{\sim a \mid a \in A\}$. Let us introduce a shorthand $A \leftarrow B, \sim C$ for rules where $A \neq \emptyset$, B and C are subsets of $\mathrm{Hb}(P)$. In harmony with (1), the set of atoms A in the *head* of the rule is interpreted disjunctively while the set of literals $B \cup \sim C$ in the *body* of the rule is interpreted conjunctively. We wish

[1]For the sake of convenience, we assume that a given program P is already instantiated by the underlying Herbrand universe, and is thus ground.

to further simplify the notation $A \leftarrow B, \sim C$ in some particular cases. When A, B or C is a singleton $\{a\}$, we write a instead of $\{a\}$. If $B = \emptyset$ or $C = \emptyset$ we omit B and $\sim C$ (respectively) as well as the separating comma in the body of the rule.

2.1 PARTIAL AND TOTAL MODELS

Let P be any disjunctive program. A *partial interpretation* I for P is a pair $\langle T, F\rangle$ of subsets of $\mathrm{Hb}(P)$ such that $T \cap F = \emptyset$. The atoms in the sets $I^{\mathbf{t}} = T$, $I^{\mathbf{f}} = F$ and $I^{\mathbf{u}} = \mathrm{Hb}(P) - (T \cup F)$ are considered to be *true*, *false*, and *undefined*, respectively. We introduce constants $\mathbf{t}$, $\mathbf{f}$, and $\mathbf{u}$, to denote the respective three truth values. A partial interpretation I for P is a *total* interpretation for P whenever $I^{\mathbf{u}} = \emptyset$, i.e., if every atom of $\mathrm{Hb}(P)$ is either true or false. When no confusion arises, we use $I^{\mathbf{t}}$ alone to specify a total interpretation I for P (then $I^{\mathbf{f}} = \mathrm{Hb}(P) - I^{\mathbf{t}}$ and $I^{\mathbf{u}} = \emptyset$ hold).

Given a partial interpretation for P, the truth values of atoms are determined by $I^{\mathbf{t}}$, $I^{\mathbf{f}}$ and $I^{\mathbf{u}}$ as explained above while $\mathbf{t}$, $\mathbf{f}$ and $\mathbf{u}$ have their fixed truth values. For more complex logical expressions E, we use $I(E)$ to denote the truth value of E in I. The value $I(\sim a)$ is defined to be $\mathbf{t}$, $\mathbf{f}$, or $\mathbf{u}$ whenever $I(a)$ is $\mathbf{f}$, $\mathbf{t}$, or $\mathbf{u}$, respectively. To handle conjunctions and disjunctions, we introduce an ordering on the three truth values by setting $\mathbf{f} < \mathbf{u} < \mathbf{t}$. By default, a set of literals $L = \{l_1, \ldots, l_n\}$ denotes the conjunction $l_1 \wedge \cdots \wedge l_n$ while $\bigvee L$ denotes the corresponding disjunction $l_1 \vee \cdots \vee l_n$. The truth values $I(L)$ and $I(\bigvee L)$ are defined as the respective minimum and maximum among the truth values $I(l_1), \ldots, I(l_n)$. A rule $A \leftarrow B, \sim C$ is satisfied in I if and only if $I(\bigvee A) \geq I(B \cup \sim C)$. A partial interpretation M for P is a *partial model* of P if all rules of P are satisfied in M, and for a *total model*, also $M^{\mathbf{u}} = \emptyset$ holds. Finally, we introduce an ordering among partial models of a disjunctive program: $M_1 \leq M_2$ iff ${M_1}^{\mathbf{t}} \subseteq {M_2}^{\mathbf{t}}$ and ${M_1}^{\mathbf{f}} \supseteq {M_2}^{\mathbf{f}}$. A partial model M of P is a *minimal* one if there is no partial model M' of P such that $M' < M$ (i.e., $M' \leq M$ and $M' \neq M$).

2.2 STABLE MODELS

Given a partial interpretation I for a disjunctive program P, we define a reduction of P as follows:

$$P^I = \{A \leftarrow B \mid A \leftarrow B, \sim C \in P \text{ and } C \subseteq I^{\mathbf{f}}\}.$$

Note that this transformation coincides with the *Gelfond-Lifschitz reduction* of P (the GL-reduction of P) when I is a total interpretation.

Definition 2.1 (Total stable model) *A total interpretation N for a disjunctive program P is a stable model iff N is a minimal total model of P^N.*

The original definition of partial stable models (Przymusinski 1990) is based on a weaker reduction. Given a disjunctive program P and an interpretation I, the reduction P_I is the set of rules obtained from P by replacing any $\sim c$ in the body of a rule by $I(\sim c)$. As noted in (Przymusinski 1990), the only practical difference between P^I and P_I is that P_I has rules that correspond to rules of $A \leftarrow B, \sim C \in P$ satisfying $I(\sim C) = \mathbf{u}$. Note that if $I(\sim C) = \mathbf{t}$, then $A \leftarrow B \in P^I$, and if $I(\sim C) = \mathbf{f}$, then the partial models of P_I are not constrained by the rule included in P_I.

Definition 2.2 (Partial stable model) *A partial interpretation M for a disjunctive program P is a partial stable model of P iff M is a minimal partial model of P_M.*

(Partial) stable models are intimately related to *unfounded sets* (Eiter et al. 1997, Leone et al. 1997).

Definition 2.3 (Unfounded sets) *Let I be a partial interpretation for a disjunctive program P. A set $U \subseteq \mathrm{Hb}(P)$ of ground atoms is an* unfounded set *for P w.r.t. I, if at least one of the following conditions holds for each rule $A \leftarrow B, \sim C \in P$ such that $A \cap U \neq \emptyset$:*

UF1: *$B \cap I^{\mathbf{t}} \neq \emptyset$ or $C \cap I^{\mathbf{t}} \neq \emptyset$,*

UF2: *$B \cap U \neq \emptyset$, or*

UF3: *$(A - U) \cap (I^{\mathbf{t}} \cup I^{\mathbf{u}}) \neq \emptyset$.*

An unfounded set U for P w.r.t. I is I-consistent iff $U \cap I^{\mathbf{t}} = \emptyset$.

The first and the last condition coincide with $I(B \cup \sim C) = \mathbf{f}$ and $I(\bigvee (A - U)) \neq \mathbf{f}$, respectively. The intuition is that the atoms of an unfounded set U can be assumed to be false without violating the satisfiability of the rules of the program that contain some atoms from U. In particular, a partial interpretation I for P becomes interesting, if the union of all unfounded sets U for P w.r.t. I is also an unfounded set for P w.r.t. I. In this case, the program P possesses *the greatest unfounded* set U for P w.r.t. I.[2] A total interpretation I is considered to be *unfounded free* for a program P iff there is no unfounded set U for P w.r.t. I such that $U \cap I^{\mathbf{t}} \neq \emptyset$. These notions lead to the following characterization of stable models.

[2]For normal programs P, this property holds for all partial interpretations I; however, for disjunctive programs this is no longer the case.

Theorem 2.4 (Leone et al. 1997) *If M is a total interpretation for a disjunctive program P, then M is a stable model of P iff $M^{\mathbf{f}}$ is the greatest unfounded set for P w.r.t. M iff M is unfounded free for P.*

On the other hand, (Eiter et al. 1997) show that partial stable models can be defined essentially without reference to three-valued logic.

Theorem 2.5 (Eiter et al. 1997) *If M is a partial interpretation for a disjunctive program P, then M is a partial stable model of P iff*

- *$M^{\mathbf{t}}$ is a minimal total model of P^M and*
- *$M^{\mathbf{f}}$ is a maximal M-consistent unfounded set for P w.r.t. M.*

3 UNFOLDING PARTIALITY

Let P be a disjunctive program. In the following, we describe a translation of P into another disjunctive program $\mathrm{Tr}(P)$ such that the stable models of $\mathrm{Tr}(P)$ correspond to the partial stable models of P.

Let us introduce a new atom $a^{\bullet}$ for each $a \in \mathrm{Hb}(P)$. The intuitive reading of $a^{\bullet}$ is that a is *potentially* true. For a set of literals $L \subseteq \mathrm{Hb}(P) \cup {\sim}\mathrm{Hb}(P)$, we define $L^{\bullet} = \{a^{\bullet} \mid a \in L\} \cup \{{\sim}a^{\bullet} \mid {\sim}a \in L\}$. The translation $\mathrm{Tr}(P)$ of a disjunctive program P is as follows:

$$\begin{aligned}&\mathrm{Tr}(P) = \\ &\quad \{A \leftarrow B, {\sim}C^{\bullet}; A^{\bullet} \leftarrow B^{\bullet}, {\sim}C \mid A \leftarrow B, {\sim}C \in P\} \cup \\ &\quad \{a^{\bullet} \leftarrow a \mid a \in \mathrm{Hb}(P)\}\end{aligned}$$

where we use semicolons to separate program rules. Note that the Herbrand base of $\mathrm{Hb}(\mathrm{Tr}(P))$ is $\mathrm{Hb}(P) \cup \mathrm{Hb}(P)^{\bullet}$. The rules $a^{\bullet} \leftarrow a$ introduced for each $a \in \mathrm{Hb}(P)$ enforce consistency in the sense that if a is true, then a must also be potentially true. The intended representation of a partial stable model M of P is given by the following equations.

Definition 3.1 *Let M be a partial interpretation of a program P and N a total interpretation of* $\mathrm{Tr}(P)$. *M and N are said to satisfy the* correspondence equations *iff the following equations hold.*

$$M^{\mathbf{t}} = \{a \in \mathrm{Hb}(P) \mid a \in N^{\mathbf{t}} \text{ and } a^{\bullet} \in N^{\mathbf{t}}\} \quad \text{(CE1)}$$
$$M^{\mathbf{f}} = \{a \in \mathrm{Hb}(P) \mid a \in N^{\mathbf{f}} \text{ and } a^{\bullet} \in N^{\mathbf{f}}\} \quad \text{(CE2)}$$
$$M^{\mathbf{u}} = \{a \in \mathrm{Hb}(P) \mid a \in N^{\mathbf{f}} \text{ and } a^{\bullet} \in N^{\mathbf{t}}\} \quad \text{(CE3)}$$
$$\emptyset = \{a \in \mathrm{Hb}(P) \mid a \in N^{\mathbf{t}} \text{ and } a^{\bullet} \in N^{\mathbf{f}}\} \quad \text{(CE4)}$$

For instance, an atom $a \in \mathrm{Hb}(P)$ is undefined in a partial interpretation M for P exactly whenever a is false and $a^{\bullet}$ is true in the corresponding total interpretation N for $\mathrm{Tr}(P)$. Note that total interpretations that are models of $\mathrm{Tr}(P)$ satisfy CE4 immediately, since the set of rules $\{a^{\bullet} \leftarrow a \mid a \in \mathrm{Hb}(P)\}$ is included in $\mathrm{Tr}(P)$. Consequently, the "fourth truth value" is ruled out. The following example demonstrates how the representation given in Definition 3.1 allows us to capture the partial stable models of a disjunctive program P with the total stable models of $\mathrm{Tr}(P)$.

Example 3.2 *Consider a disjunctive program*

$$P = \{a \vee b \leftarrow {\sim}c;\;\; b \leftarrow {\sim}b;\;\; c \leftarrow {\sim}c\}.$$

Now a becomes false by the minimization of partial models, since the falsity of a does not affect the satisfiability of any rule. Thus the unique partial stable model of P is $M = \langle \emptyset, \{a\} \rangle$. Note that the reduction $P_M = \{a \vee b \leftarrow \mathbf{u};\; b \leftarrow \mathbf{u};\; c \leftarrow \mathbf{u}\}$. Then consider the translation

$$\begin{aligned}\mathrm{Tr}(P) = \{\, & a \vee b \leftarrow {\sim}c^{\bullet}; && b \leftarrow {\sim}b^{\bullet}; && c \leftarrow {\sim}c^{\bullet}; \\ & a^{\bullet} \vee b^{\bullet} \leftarrow {\sim}c; && b^{\bullet} \leftarrow {\sim}b; && c^{\bullet} \leftarrow {\sim}c; \\ & a^{\bullet} \leftarrow a; && b^{\bullet} \leftarrow b; && c^{\bullet} \leftarrow c \quad \}\end{aligned}$$

The unique stable model of $\mathrm{Tr}(P)$ *is $N = \{b^{\bullet}, c^{\bullet}\}$ which represents (by CE2 and CE3) the setting that b and c are undefined and a is false in M.*

It is well-known that a disjunctive program P may not have any partial stable models. In such cases, the translation $\mathrm{Tr}(P)$ should not have stable models either, if the translation $\mathrm{Tr}(P)$ is to be faithful.

Example 3.3 *Consider a program*

$$P = \{a \vee b \vee c \leftarrow;\;\; a \leftarrow {\sim}b;\;\; b \leftarrow {\sim}c;\;\; c \leftarrow {\sim}a\}$$

and $\mathrm{Tr}(P) =$

$$\begin{aligned}\{\, & a \vee b \vee c \leftarrow; && a \leftarrow {\sim}b^{\bullet}; && b \leftarrow {\sim}c^{\bullet}; && c \leftarrow {\sim}a^{\bullet}; \\ & a^{\bullet} \vee b^{\bullet} \vee c^{\bullet} \leftarrow; && a^{\bullet} \leftarrow {\sim}b; && b^{\bullet} \leftarrow {\sim}c; && c^{\bullet} \leftarrow {\sim}a \;\} \\ & \cup\, C\end{aligned}$$

where $C = \{a^{\bullet} \leftarrow a;\;\; b^{\bullet} \leftarrow b;\;\; c^{\bullet} \leftarrow c\}$.

Consider a partial model $M = \langle \{a, b\}, \emptyset \rangle$ of P and a total model $N = \{a, a^{\bullet}, b, b^{\bullet}, c^{\bullet}\}$ of $\mathrm{Tr}(P)$ *that satisfy the equations CE1–CE4 in Definition 3.1. Now the reduced program P_M is*

$$\{a \vee b \vee c \leftarrow;\;\; a \leftarrow \mathbf{f};\;\; b \leftarrow \mathbf{u};\;\; c \leftarrow \mathbf{f}\}$$

and since $M' = \langle \{a, b\}, \{c\} \rangle < M$ is a partial model of P_M, M is not a partial stable model of P. On the other hand, the reduct

$$\mathrm{Tr}(P)^N = \{a \vee b \vee c \leftarrow;\;\; a^{\bullet} \vee b^{\bullet} \vee c^{\bullet} \leftarrow;\;\; b^{\bullet} \leftarrow\} \cup C.$$

But $N' = \{a, a^\bullet, b, b^\bullet\} \subset N$ is a model of $\mathrm{Tr}(P)^N$ so that N is not a stable model of $\mathrm{Tr}(P)$. The reader may analyze the other candidates in a similar fashion. It turns out that P does not have partial stable models. Nor does $\mathrm{Tr}(P)$ have stable models.

The main result of this section is established using the characterizations of disjunctive stable models in Theorems 2.4 and 2.5 (see the appendix for a proof).

Theorem 3.4 *Let M be a partial interpretation of a disjunctive program P and N a total interpretation of the translation $\mathrm{Tr}(P)$ such that CE1–CE4 are satisfied. Then M is a partial stable model of P iff N is a stable model of $\mathrm{Tr}(P)$.*

Let us yet address the possibility of using an inference engine for computing total stable models to answer queries concerning partial stable models. This is highly interesting, because there are already systems available for computing total stable models (Cholewiński et al. 1996, Leone et al. 1999, Simons 1999b) while partial stable models lack implementations. Here we must remind the reader that partial stable models can be used in different ways in order to evaluate queries. Typically two modes of reasoning are used: *certainty inference* and *possibility inference*. In the former approach, a query Q should be true in all (intended) models of P while Q should be true in some (intended) model of P in the latter approach. Moreover, maximal partial stable models (under set inclusion) are sometimes distinguished; this is how *regular models* and *preferred extensions* are obtained for normal programs (Dung 1995, Saccà and Zaniolo 1990, You and Yuan 1994). We are particularly interested in possibility inference where the maximality condition makes no difference: $M(Q) = \mathbf{t}$ for some partial stable model M of P iff $M'(Q) = \mathbf{t}$ for some maximal partial stable model M' of P.

We consider queries Q that are sets of literals over $\mathrm{Hb}(P)$ and queries are translated as follows: $\mathrm{Tr}(Q) = Q \cup Q^\bullet$. As a direct consequence of Theorem 3.4 and CE1, we obtain the following.

Corollary 3.5 *A query Q is true in a (maximal) partial stable model of P iff $\mathrm{Tr}(Q)$ is true in a stable model of $\mathrm{Tr}(P)$.*

What about using an inference engine for computing partial stable models to answer queries concerning stable models? A slight extension of the translation $\mathrm{Tr}(P)$ is needed for this purpose: let $\mathrm{Tr}_2(P)$ be $\mathrm{Tr}(P)$ augmented with a set of rules $\{f \leftarrow a^\bullet, {\sim}a \mid a \in \mathrm{Hb}(P)\}$ where $f \notin \mathrm{Hb}(P)$ is a new atom. The purpose of these additional rules is to detect partial stable models with remaining undefined atoms. A query Q is translated into $\mathrm{Tr}_2(Q) = Q \cup \{{\sim}f\}$.

Corollary 3.6 *A query Q is true in a stable model of P if and only if $\mathrm{Tr}_2(Q)$ is true in a partial stable model of $\mathrm{Tr}_2(P)$.*

4 UNFOLDING DISJUNCTIONS

In this section we develop a method for reducing the task of computing a (total) stable model of a disjunctive program to stable model computation for normal (disjunction-free) programs. Since the problem of deciding whether a disjunctive program has a stable model is Σ_2^p-complete (Eiter and Gottlob 1995) whereas the problem is NP-complete in the non-disjunctive case (Marek and Truszczyński 1991), the reduction cannot be computable in polynomial time unless the polynomial hierarchy collapses. Our idea is to use a straightforward generate and test approach.

The basic idea is that given a disjunctive program P we compute its stable models in two phases: (i) we *generate* model candidates and (ii) *test* candidates for stability until we find a suitable model. For generating model candidates we construct a normal program $\mathrm{Gen}(P)$ such that the stable models of $\mathrm{Gen}(P)$ give the candidate models. For testing a candidate model M we build another normal program $\mathrm{Test}(P, M)$ such that $\mathrm{Test}(P, M)$ has no stable models iff M is a stable model of the original disjunctive program P. Hence, given a procedure for computing stable models for normal programs all stable models of a disjunctive program P can be generated as follows: for each stable model M of $\mathrm{Gen}(P)$, decide whether $\mathrm{Test}(P, M)$ has a stable model and if this is not the case, output M as a stable model of P.

It is easy to construct a normal program $\mathrm{Gen}(P)$ for generating candidate models, e.g., by introducing for each atom $a \in \mathrm{Hb}(P)$, two rules $a \leftarrow {\sim}\hat{a}$; $\hat{a} \leftarrow {\sim}a$ where $\hat{a}$ is a new atom. These rules generate stable models corresponding to every subset of $\mathrm{Hb}(P)$. In order to prune this set of models to those with all rules in P satisfied, it is sufficient to include a rule

$$f \leftarrow {\sim}f, {\sim}a_1, \ldots, {\sim}a_k, b_1, \ldots, b_m, {\sim}c_1, \ldots, {\sim}c_n \quad (2)$$

for each rule of the form (1) in P where f is a new atom. As f cannot be in any stable model, the rule functions as an integrity constraint eliminating the models where each b_i is included, every c_j is excluded but no a_l is included.

In order to guarantee completeness, it is sufficient that for each stable model M of P there is a corresponding model candidate which agrees with M w.r.t. $\mathrm{Hb}(P)$. It is clear that our first generating program satisfies this condition. However, for efficiency it is important to devise a generating program that has as few as possible (candidate) stable models provided that completeness is not lost. For given a disjunctive program P, a generating program $\mathrm{Gen}(P)$ that typically has far fewer stable models but still preserves completeness can be constructed as follows:

$$\begin{aligned}
&\mathrm{Gen}(P) = \\
&\quad \{a \leftarrow {\sim}\hat{a}, B, {\sim}C \mid A \leftarrow B, {\sim}C \in P, a \in A\} \cup \\
&\quad \{\hat{a} \leftarrow {\sim}a \mid a \in \mathrm{Hb}(P)\} \cup \\
&\quad \{f \leftarrow {\sim}f, {\sim}A, B, {\sim}C \mid A \leftarrow B, {\sim}C \in P\}
\end{aligned}$$

Proposition 4.1 *Let P be a disjunctive program. Then if M is a stable model of P, there is a stable model M' of* $\mathrm{Gen}(P)$ *with $M = M' \cap \mathrm{Hb}(P)$.*

Proof. Let M be a stable model of P and let $M' = M \cup \{\hat{a} \mid a \in \mathrm{Hb}(P) - M\}$. Now clearly $M = M' \cap \mathrm{Hb}(P)$. We show that M' coincides with the minimal model Min of $\mathrm{Gen}(P)^{M'}$, i.e., M' is a stable model of $\mathrm{Gen}(P)$. We prove this by showing that (i) M' is a model of $\mathrm{Gen}(P)^{M'}$ and that (ii) $M' \subseteq Min$ which together imply $M' = Min$.

Property (i) holds because M is a model of P and because $\hat{a} \leftarrow \in \mathrm{Gen}(P)^{M'}$ iff $a \in \mathrm{Hb}(P) - M$ and $a \leftarrow B \in \mathrm{Gen}(P)^{M'}$ iff $a \in M$ and $C \cap M = \emptyset$. In order to establish (ii) we note that (iii) for all $a \in \mathrm{Hb}(P)$, $a \in Min$ implies $a \in M$ because for each rule in $\mathrm{Gen}(P)^{M'}$ with such an atom a in the head, $a \in M$. Consider $A \leftarrow B \in P^M$. If the body B is true in Min, then by (iii) B is true in M and, hence, at least one $a_i \in A \cap M$. Then $a_i \leftarrow B \in \mathrm{Gen}(P)^{M'}$ and $a_i \in Min$. Hence, Min is a model of P^M which implies $M \subseteq Min$ and $M' \subseteq Min$. □

A (total) model candidate $M \subseteq \mathrm{Hb}(P)$ is a stable model of a program P if it is a minimal model of the GL-transform P^M of the program. This test can be reduced to an unsatisfiability problem in propositional logic using techniques presented in (Niemelä 1996): M is a minimal model of P^M iff

$$\begin{aligned}
P^M &\cup \{\neg a \mid a \in \mathrm{Hb}(P) - M\} \\
&\cup \{\neg b_1 \vee \cdots \vee \neg b_m\}
\end{aligned} \tag{3}$$

is unsatisfiable where $M = \{b_1, \ldots, b_m\}$ and the rules in P^M are seen as clauses. This problem can be solved by testing non-existence of stable models for a normal program $\mathrm{Test}(P, M)$ which is constructed for a disjunctive program P and a total interpretation $M \subseteq \mathrm{Hb}(P)$ as follows:

$$\begin{aligned}
&\mathrm{Test}(P, M) = \\
&\quad \{a \leftarrow {\sim}\hat{a}, B \mid A \leftarrow B \in P^M, a \in A \cap M, B \subseteq M\} \cup \\
&\quad \{\hat{a} \leftarrow {\sim}a \mid a \in \mathrm{Hb}(P)\} \cup \\
&\quad \{f \leftarrow {\sim}f, {\sim}A, B \mid A \leftarrow B \in P^M, B \subseteq M\} \cup \\
&\quad \{f \leftarrow {\sim}f, M\}
\end{aligned}$$

Proposition 4.2 *Let P be a disjunctive program and M its (total) model. Then M is a minimal model of P^M iff* $\mathrm{Test}(P, M)$ *has no stable model.*

Proof. Let $M \subseteq \mathrm{Hb}(P)$ be a total model of P.

($\Rightarrow$) Let M' be a stable model of $\mathrm{Test}(P, M)$. If $a \in \mathrm{Hb}(P) - M$, then there is no rule with a in the head in $\mathrm{Test}(P, M)$ and $a \notin M'$. Hence, $M' \cap \mathrm{Hb}(P) \subseteq M$. As $f \notin M'$ and $f \leftarrow M \in \mathrm{Test}(P, M)^{M'}$, there is some $a \in M$ such that $a \notin M' \cap \mathrm{Hb}(P)$. Consider $A \leftarrow B \in P^M$. Let $B \subseteq M' \cap \mathrm{Hb}(P) \subseteq M$ but suppose $A \cap M' \cap \mathrm{Hb}(P) = \emptyset$. Then $f \leftarrow B \in \mathrm{Test}(P, M)^{M'}$ and $f \in M'$, a contradiction. Hence, $M' \cap \mathrm{Hb}(P)$ is a model of P^M but $M' \cap \mathrm{Hb}(P) \subset M$ implying that M is not a minimal model of P^M.

($\Leftarrow$) Assume that M is not a minimal model of P^M. As M is a model of P^M, there is a minimal model $Min \subset M$ of P^M.

We show that $M' = Min \cup \{\hat{a} \mid a \in \mathrm{Hb}(P) - Min\}$ coincides with the minimal model Min' of $\mathrm{Test}(P, M)^{M'}$, i.e., M' a stable model of $\mathrm{Test}(P, M)$. Now $\mathrm{Test}(P, M)^{M'} =$

$$\begin{aligned}
&\{a \leftarrow B \mid A \leftarrow B \in P^M, \\
&\qquad a \in A \cap M \cap Min, B \subseteq M\} \cup \\
&\{\hat{a} \leftarrow \mid a \in \mathrm{Hb}(P) - Min\} \cup \\
&\{f \leftarrow B \mid A \leftarrow B \in P^M, A \cap Min = \emptyset, B \subseteq M\} \cup \\
&\{f \leftarrow M\}
\end{aligned}$$

It is easy to check that (i) M' is a model of $\mathrm{Test}(P, M)^{M'}$. Furthermore, (ii) $M' \subseteq Min'$ holds which can be established as follows. We notice that for all $a \in \mathrm{Hb}(P)$, $a \in Min'$ implies $a \in Min$. Consider $A \leftarrow B \in P^M$. If B is true in Min', then B is true in Min and, thus, $B \subseteq M$ and some $a \in A \cap M \cap Min$. Then $a \leftarrow B \in \mathrm{Test}(P, M)^{M'}$ and $a \in Min'$. Hence, Min' is a model of P^M which implies $Min \subseteq Min'$. Then $M' \subseteq Min'$ holds. Now (i) and (ii) imply $Min' = M'$. Hence, M' is a stable model of $\mathrm{Test}(P, M)$. □

Example 4.3 *Consider a disjunctive program P and* *its generator* $\mathrm{Gen}(P)$*:*

$$P = \{a \vee b \leftarrow \sim c\}$$

$$\begin{aligned}\mathrm{Gen}(P) = \{ & a \leftarrow \sim\hat{a}, \sim c; \quad b \leftarrow \sim\hat{b}, \sim c; \\ & \hat{a} \leftarrow \sim a; \quad \hat{b} \leftarrow \sim b; \quad \hat{c} \leftarrow \sim c; \\ & f \leftarrow \sim f, \sim a, \sim b, \sim c \quad \}\end{aligned}$$

For stable models $\{a, b, \hat{c}\}$ *and* $\{b, \hat{a}, \hat{c}\}$ *of* $\mathrm{Gen}(P)$ *the corresponding model candidates are* $M_1 = \{a, b, \hat{c}\} \cap \mathrm{Hb}(P) = \{a, b\}$ *and* $M_2 = \{b, \hat{a}, \hat{c}\} \cap \mathrm{Hb}(P) = \{b\}$ *and their test programs:*

$$\begin{aligned}\mathrm{Test}(P, M_1) = \{ & a \leftarrow \sim\hat{a}; \quad b \leftarrow \sim\hat{b}; \\ & \hat{a} \leftarrow \sim a; \quad \hat{b} \leftarrow \sim b; \quad \hat{c} \leftarrow \sim c; \\ & f \leftarrow \sim f, \sim a, \sim b; \quad f \leftarrow \sim f, a, b \ \} \\ \mathrm{Test}(P, M_2) = \{ & b \leftarrow \sim\hat{b}; \\ & \hat{a} \leftarrow \sim a; \quad \hat{b} \leftarrow \sim b; \quad \hat{c} \leftarrow \sim c; \\ & f \leftarrow \sim f, \sim a, \sim b; \quad f \leftarrow \sim f, b \ \}\end{aligned}$$

$\mathrm{Test}(P, M_1)$ *has a stable model (e.g.,* $\{a, \hat{b}, \hat{c}\}$*) indicating that* M_1 *is not a stable model of* P *but* $\mathrm{Test}(P, M_2)$ *has no stable models and, hence,* M_2 *is a stable model of* P*.*

The simple generate and test paradigm can be optimized by building model candidates gradually. This means that we start from the empty partial interpretation and extend the interpretation step by step. An interesting observation is that the technique for testing minimality can be used to rule out a partial model candidate of $\mathrm{Gen}(P)$ at any stage of the search and not just when a total model of the program P has been found. This can be done by treating a partial interpretation M as a total interpretation where undefined atoms are taken to be false and using the $\mathrm{Test}(P, M)$ program.

Proposition 4.4 *Let* P *be a disjunctive program and* M *a total interpretation. If* $\mathrm{Test}(P, M)$ *has a stable model, then there is no (total) stable model* M' *of* P *such that* $M \subseteq M'$*.*

Proof. Let $\mathrm{Test}(P, M)$ have a stable model. As shown in the proof of Proposition 4.2, then there is a model M'' of P^M with $M'' \subset M$. Consider any total interpretation M' such that $M \subseteq M'$ and M' is a model of $P^{M'}$. Now M' is not a minimal model of $P^{M'}$ as $P^{M'} \subseteq P^M$ and, hence, M'' is a model of $P^{M'}$ but $M'' \subset M \subseteq M'$. □

Notice that for a total interpretation M, Proposition 4.4 can only be used for eliminating stable models of P extending M. For guaranteeing the existence of a stable model of P, a total model of P needs to be found making Proposition 4.2 applicable.

5 EXPERIMENTS

We compare *dlv* (Leone et al. 1999), a state-of-the-art implementation of the stable model semantics for disjunctive logic programs, with an implementation of the generate and test approach (GnT) of the previous section. The implementation is based on *smodels* (Simons 1999b), a program that computes stable models of normal logic programs, and it is available (Simons 1999a).

We create two instances of *smodels*, one that generates the models and one that tests if they are minimal. Each time we find a model we check if it is minimal. If it is not, then we backtrack and perform the minimality test (Proposition 4.4) by viewing the resulting partial model as a total model. If the test fails again, then we repeat and backtrack until the test succeeds. Next, we resume the search for more models. The implementation of the procedure consists of less than 400 lines of code on top to the *smodels* system.

The two systems are tested on 3-SAT problems from the phase transition region. Given a propositional formula in conjunctive normal form whose clause to atom ratio is 4.258, we translate each clause $a_1 \vee \cdots \vee a_n \vee \neg b_1 \vee \cdots \vee \neg b_m$ into a rule

$$a_1 \vee \cdots \vee a_n \leftarrow b_1, \ldots, b_m.$$

In order to make the problem Σ_2^p-complete, we search for minimal models that contain some specified atoms (Eiter and Gottlob 1995). In particular, we create the rules

$$f \leftarrow \sim f, \sim c_i$$

for $i = 1, \ldots, \lfloor 2n/100 \rfloor$ and for random atoms c_i, where n is the number of atoms in the formula.

We generate random problems that have between 100 and 340 atoms. For each problem size we test fifty programs and measure the maximum, average, and minimum time it takes to decide whether a stable model exists. We have also tested the strategic companies problem from (Koch and Leone 1999), but we do not include the results as it turns out that the problem is almost trivial for both *dlv* and the GnT approach. All tests were run under Linux 2.2.12 on 450 MHz Pentium III computers with 256 MB of memory. The test results are shown in Figure 1.

For this set of benchmark problems, our GnT approach is very competitive when compared *dlv* except for two test problems (one having 270 atoms and the other 300 atoms) where the number of generated model candidates is orders of magnitude higher than in any of the other 49 test problems of the same size.

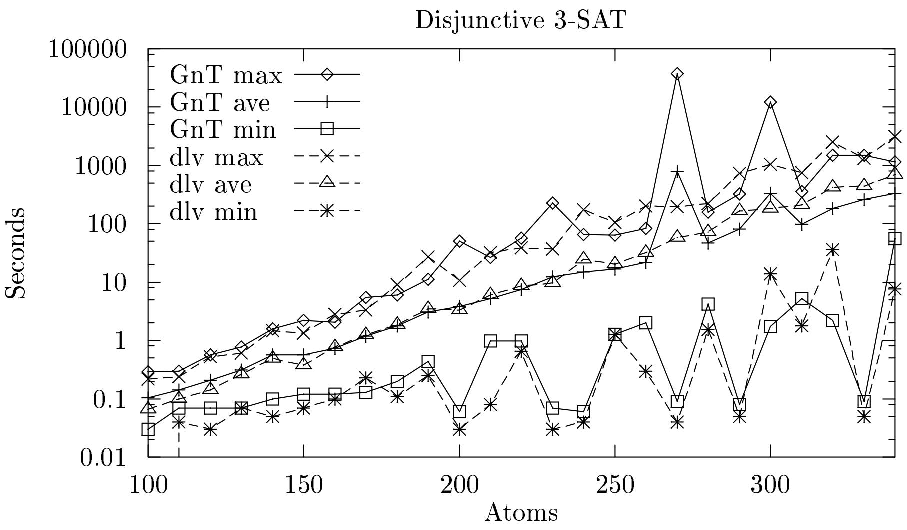

Figure 1: Experimental Results

6 CONCLUSIONS

The paper presents an approach to implementing partial and disjunctive stable models using an implementation of stable models for disjunction-free programs as the core inference engine. The approach is based on unfolding partiality and disjunctions from a logic program in two separate steps. In the first step partial stable models are captured by total stable models using a simple linear program transformation. Thus, reasoning tasks concerning partial models can be solved using an implementation of total models. This also sheds new light on the relationship between partial and total stable models by establishing a close correspondence. In the second step a generate and test approach is developed for computing total stable models of disjunctive programs using a core engine capable of computing stable models of normal programs. We have developed an implementation of the approach using *smodels* as the core engine. The extension is fairly simple consisting of a few hundred lines of code. The approach turns out to be competitive even against a state-of-the-art special purpose system for disjunctive programs. The efficiency of the approach comes partly from the fact that normal programs can capture essential properties of disjunctive stable models that help decrease the computational complexity of the generate and test phases in the approach. However, a major part of the success can be accounted for by the efficiency of the core engine. This suggests that more efforts should be spent in developing efficient core engines.

Acknowledgements

The work of the first, second and third authors has been funded by the Academy of Finland (Project 43963), that of the third author by the Helsinki Graduate School in Computer Science and Engineering. We thank Tommi Syrjänen for implementing the front-end for the *smodels* system.

References

Cholewiński, P., Marek, V. and Truszczyński, M. (1996). Default reasoning system DeReS, *Proceedings of the 5th International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann Publishers, Cambridge, MA, USA, pp. 518–528.

Dimopoulos, Y., Nebel, B. and Koehler, J. (1997). Encoding planning problems in non-monotonic logic programs, *Proceedings of the Fourth European Conference on Planning*, Springer-Verlag, Toulouse, France, pp. 169–181.

Dung, P. (1995). An argumentation theoretic foundation for logic programming, *Journal of Logic Programming* **22**: 151–177.

Eiter, T. and Gottlob, G. (1995). On the computational cost of disjunctive logic programming: Propositional case, *Annals of Mathematics and Artificial Intelligence* **15**: 289–323.

Eiter, T., Leone, N. and Saccà, D. (1997). On the partial semantics for disjunctive deductive databases, *Annals of Mathematics and Artificial Intelligence* **17**(1/2): 59–96.

Eiter, T., Leone, N. and Saccà, D. (1998). Expressive power and complexity of partial models for disjunctive deductive databases, *Theoretical Computer Science* **206**(1/2): 181–218.

Eshghi, K. and Kowalski, R. (1989). Abduction compared with negation by failure, *Proceedings of the 6th International Conference on Logic Programming*, MIT Press, pp. 234–254.

Gelfond, M. and Lifschitz, V. (1988). The stable model semantics for logic programming, *Proceedings of the 5th International Conference on Logic Programming*, The MIT Press, Seattle, USA, pp. 1070–1080.

Heljanko, K. (1999). Using logic programs with stable model semantics to solve deadlock and reachability problems for 1-safe Petri nets, *Fundamenta Informaticae* **37**(3): 247–268.

Koch, C. and Leone, N. (1999). Stable model checking made easy, *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, pp. 70–75.

Leone, N. et al. (1999). Dlv, release 1999-11-24, `http://www.dbai.tuwien.ac.at/proj/dlv/`. A Disjunctive Datalog System.

Leone, N., Rullo, L. and Scarcello, F. (1997). Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation, *Information and Computation* **135**(2): 69–112.

Liu, X., Ramakrishnan, C. and Smolka, S. A. (1998). Fully local and efficient evaluation of alternating fixed points, *in* B. Steffen (ed.), *Proceedings of the 4th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, Springer-Verlag, Lisbon, Portugal, pp. 5–19.

Marek, W. and Truszczyński, M. (1991). Autoepistemic logic, *Journal of the ACM* **38**: 588–619.

Marek, W. and Truszczyński, M. (1999). Stable models and an alternative logic programming paradigm, *The Logic Programming Paradigm: a 25-Year Perspective*, Springer-Verlag, pp. 375–398.

Niemelä, I. (1996). A tableau calculus for minimal model reasoning, *Proceedings of the Fifth Workshop on Theorem Proving with Analytic Tableaux and Related Methods*, Springer-Verlag, Terrasini, Italy, pp. 278–294.

Niemelä, I. (1999). Logic programming with stable model semantics as a constraint programming paradigm, *Annals of Mathematics and Artificial Intelligence* **25**(3,4): 241–273.

Niemelä, I. and Simons, P. (1997). Smodels – an implementation of the stable model and well-founded semantics for normal logic programs, *Proceedings of the 4th International Conference on Logic Programming and Non-Monotonic Reasoning*, Springer-Verlag, Dagstuhl, Germany, pp. 420–429.

Przymusinski, T. (1990). Extended stable semantics for normal and disjunctive logic programs, *Proceedings of the 7th International Conference on Logic Programming*, MIT Press, pp. 459–477.

Saccà, D. and Zaniolo, C. (1990). Stable models and non-determinism in logic programs with negation, *Proceedings of the 9th ACM Symposium on Principles of Database Systems*, pp. 205–217.

Sagonas, K., Swift, T. and Warren, D. (1996). An abstract machine for computing the well-founded semantics, *in* M. Maher (ed.), *Proceedings of the Joint International Conference and Symposium on Logic Programming*, The MIT Press, Bonn, Germany, pp. 274–288.

Simons, P. (1999a). Disjunctive logic programming by generate and test, `http://www.tcs.hut.fi/Software/smodels/`. Available as examples/example4.cc in the smodels source distribution.

Simons, P. (1999b). Smodels version 2.25, `http://www.tcs.hut.fi/Software/smodels/`. A system for computing the stable models of logic programs.

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

A APPENDIX: PROOF OF THEOREM 3.4

The main result of Section 3 is a theorem showing a one-to-one correspondence between the partial stable models M of a program P and the stable models N of its translation. Essentially, we prove that M and N form a pair that satisfies the correspondence equations CE1–CE4 given in Definition 3.1 by the intended representation of atoms and their truth values.

It is worth noting that although we assume ground programs, since our proofs do not rely on the assumption that the given program be finite, all of the results apply to the non-ground case as well.

The strategy to prove Theorem 3.4 is as follows: first, in two separate lemmas, we show the correspondence, in each direction, between unfounded sets for P and $\mathrm{Tr}(P)$ under M and N, respectively; these results will then be used to establish the required maximal unfoundedness, and foundedness in case of partial stable models. Before doing that we present two simple facts which can be verified easily.

Proposition A.1 *Let P be a program, N be a total interpretation. Then, X is an unfounded set for P w.r.t. N iff X is an unfounded set for P^N w.r.t. N.*

Proposition A.2 *Let M be a partial interpretation of a program P and N be a total interpretation of its translation $\mathrm{Tr}(P)$. Assume M and N satisfy the CEs. Then, M is a model of P iff N is a model of $\mathrm{Tr}(P)$.*

In the following, by abuse of terminology, an unfounded set X for $\mathrm{Tr}(P)$ is said to satisfy the consistency rules $\{a^\bullet \leftarrow a \mid a \in \mathrm{Hb}(P)\} \subseteq \mathrm{Tr}(P)$ iff $a^\bullet \in X$ implies $a \in X$. Since an unfounded set is about atoms that can be assumed false, this ensures that if the head of a consistency rule is false, the body must be false. An atom $a^\bullet$ is said to be *marked*, and an ordinary atom a is then said to be *unmarked*.

We also use the following notations: given a rule r, $H(r)$ refers to the set of head atoms in r, $B^+(r)$ refers to the set of positive body literals in r, $B^-(r)$ denotes the set of negative body literals in r, and $B(r)$ is simply $B^+(r) \cup B^-(r)$. Recall that in the translation each rule r: $A \leftarrow B, \sim C$ is translated into two rules $A \leftarrow B, \sim C^\bullet$ and $A^\bullet \leftarrow B^\bullet, \sim C$. The former is referred to as $r_{n'}$ (negative literals marked) and the latter as $r_{p'}$ (positive literals marked).

Lemma A.3 *Let P be a program, N be a total interpretation of the translated program $\mathrm{Tr}(P)$, and M be a partial interpretation of P satisfying the CEs. Then, for any unfounded set X_N for $\mathrm{Tr}(P)$ w.r.t. N satisfying the consistency rules, $X_M = \{a \mid a, a^\bullet \in X_N\}$ is an unfounded set for P w.r.t. M. In addition, if X_N is N-consistent then X_M is M-consistent.*

Proof. The additional claim on consistency is easy to show. Here we prove the main claim. Let $a \in X_M$. We prove that for any rule $r \in P$ such that $a \in H(r)$, one of the conditions for unfoundedness applies to r. We note that $B(r)$ is either false, or true, or undefined in M. Consider all three cases.

A. $B(r)$ **is false in** M**.** Then UF1 applies to r.

B. $B(r)$ **is true in** M**.** We show that either UF2 or UF3 applies to r. We first note that, by the definition of X_M and the fact that $a \in X_M$, we have $a^\bullet \in X_N$. Since X_N is unfounded w.r.t. N, and since in this case UF1 does not apply to $r_{p'}$ (by the CEs, that $B(r)$ is true in M implies that $B(r_{p'})$ is true in N) we know that either UF2 or UF3 applies to $r_{p'}$.

(i) *UF2 applies to $r_{p'}$.* Then, $\exists c^\bullet \in B(r_{p'})$ such that $c^\bullet \in X_N$. Since the rule $c^\bullet \leftarrow c$ is satisfied by X_N, we have $c \in X_N$. That $c, c^\bullet \in X_N$ implies $c \in X_M$. Thus UF2 applies to r.

(ii) *UF3 applies to $r_{p'}$.* Then, $\exists b^\bullet \in H(r_{p'})$ such that $b^\bullet$ is true in N and $b^\bullet \notin X_N$. That $b^\bullet$ is true in N implies that b is not false in M. By the definition of X_M, $b^\bullet \notin X_N$ implies $b \notin X_M$. Thus UF3 applies to r.

C. $B(r)$ **is undefined in** M. Then by the CEs, $B(r_{p'})$ is true in N. Since $a^\bullet \in X_N$ and $a^\bullet \in H(r_{p'})$, either UF2 or UF3 applies to $r_{p'}$. The proof follows from (i) and (ii) above. □

The condition that the consistency rules be satisfied in the above lemma is tight, as illustrated by the following example.

Example A.4 *Consider $P = \{a \vee b \leftarrow c_1, c_2\}$ and its translation*

$$\mathrm{Tr}(P) = \{\, a \vee b \leftarrow c_1, c_2; \quad a^\bullet \vee b^\bullet \leftarrow {c_1}^\bullet, {c_2}^\bullet; \quad a^\bullet \leftarrow a;\ b^\bullet \leftarrow b;\ \ {c_1}^\bullet \leftarrow c_1;\ {c_2}^\bullet \leftarrow c_2 \,\}$$

Let $N = \{a, a^\bullet, b, b^\bullet, c_1, {c_1}^\bullet, {c_2}^\bullet\}$ be an interpretation for $\mathrm{Tr}(P)$. Then the corresponding interpretation for P is $M = \langle\{a, b, c_1\}, \emptyset\rangle$. Consider $X_N = \{a, a^\bullet, b, b^\bullet, c_1, {c_2}^\bullet\}$. It can be verified easily that X_N is unfounded for $\mathrm{Tr}(P)$ w.r.t. N. Also note that since ${c_2}^\bullet \in X_N$ and $c_2 \notin X_N$, X_N does not satisfy the consistency rule ${c_2}^\bullet \leftarrow c_2$. We can verify that the corresponding $X_M = \{a, b\}$ (as defined in the lemma) is *not unfounded for P w.r.t. M; for the only rule in P, its body is undefined thus UF1 does not apply; UF2 does not apply either since no atom in X_M that appears in the body of the rule; and finally UF3 fails to be applicable too since both a and b are in X_M.*

Lemma A.5 *Let M be a partial model of a program P and N be a total interpretation of $\mathrm{Tr}(P)$ satisfying the CEs. Suppose X_M is a set of ordinary atoms, and $X_N = F \cup U$ where $F = \{a, a^\bullet \mid a \in X_M\}$, and U is any subset of $\{a \mid a \in N^{\mathbf{f}}, a^\bullet \in N^{\mathbf{t}}\}$. Then, if X_M is an M-consistent unfounded set for P w.r.t. M, then X_N is an unfounded set for $\mathrm{Tr}(P)$ w.r.t. N.*

Proof. Any atom in X_N is either marked or unmarked. We prove the theorem by considering these two cases.

I. $a^\bullet \in X_N$**.** First, for the consistency rule $a^\bullet \leftarrow a \in \mathrm{Tr}(P)$, by the definition of X_N, it is clear that if $a^\bullet \in X_N$ then $a \in X_N$, so UF2 applies. For any other rule $r_{p'} \in \mathrm{Tr}(P)$ such that $a^\bullet \in H(r_{p'})$, we prove that one of the three conditions for unfoundedness applies to $r_{p'}$. Since N is a total interpretation, we have that $B(r_{p'})$ is either false in N, in which case UF1 applies to $r_{p'}$, or true in N in which case $B(r)$ is true or undefined in M. Since $a \in X_M$ and $a \in H(r)$, and UF1 does not apply to r, we only need to consider UF2 and UF3. If UF2 applies to r, $\exists c \in B(r)$ such that $c \in X_M$. It follows, by the definition of X_N, that $c^\bullet \in X_N$. Hence UF2 applies to $r_{p'}$. If UF3 applies to r, $\exists b \in H(r)$ such that b is not false in M and $b \notin X_M$. By the CEs, and by the fact that b is not false in M, we know $b^\bullet$ is true in N. Further, by the definition of X_N, $b \notin X_M$ implies $b^\bullet \notin X_N$. Hence UF3 applies to $r_{p'}$.

II. $a \in X_N$**.** For any $r_{n'} \in \mathrm{Tr}(P)$ such that $a \in H(r_{n'})$, $B(r_{n'})$ is either false in N, in which case UF1 applies to $r_{n'}$, or true in N, in which case, by Proposition A.2 we know that M being a model of P implies N is a model of $\mathrm{Tr}(P)$, hence $\exists b \in H(r_{n'})$ such that b is true in N. Two cases arise: there exists one such b that $b \notin X_N$, or for all such b, $b \in X_N$. In the first case, UF3 applies to $r_{n'}$. The proof is completed by showing that the latter case leads to a contradiction.

In this latter case, for each such b, from $b \in X_N$ we get $b \in F$ or $b \in U$. If $b \in F$ then $b, b^\bullet \in F$, hence $b \in X_M$. But b being true in N implies that b is true in M, and by $b \in X_M$ we get $M^{\mathbf{t}} \cap X_M \neq \emptyset$. This contradicts the assumption that X_M is M-consistent. Thus, for any such b, it must be the case that $b \in U$. By the definition of U this means that b must be false in N. A contradiction. This completes the proof. □

We note that M-consistency of X_M is also a necessary condition for the correspondence to hold.

Example A.6 *Consider the following program*

$$P = \{a \vee b \leftarrow;\ a \leftarrow {\sim}a\}$$

and a partial model $M = \langle \{b\}, \emptyset \rangle$ of P. It can be checked easily that $X_M = \{b\}$ is an unfounded set for P w.r.t. M (UF3 applies to the only rule in which b appears in the head). But X_M is not M-consistent. Now consider

$$\begin{aligned}\mathrm{Tr}(P) = \{\, & a \vee b \leftarrow; \quad a^\bullet \vee b^\bullet \leftarrow; \quad a \leftarrow {\sim}a^\bullet; \\ & a^\bullet \leftarrow {\sim}a; \quad a^\bullet \leftarrow a; \quad b^\bullet \leftarrow b \quad \}\end{aligned}$$

The total interpretation corresponding to M above is $N = \{b^\bullet, b, a^\bullet\}$. However, $X_N = \{b, b^\bullet, a\}$ is not unfounded for $\mathrm{Tr}(P)$ w.r.t. N, since for $b \in X_N$ and the first rule in $\mathrm{Tr}(P)$, none of the unfoundedness conditions applies.

Let us establish Theorem 3.4 in two separate theorems. Let M be a partial interpretation of a disjunctive program P and N a total interpretation of $\mathrm{Tr}(P)$ such that CE1–CE4 are satisfied.

Theorem A.7 *If N is a stable model of $\mathrm{Tr}(P)$, then M is a partial stable model of P.*

Proof. We prove that $M^{\mathbf{t}}$ is a minimal total model of P^M, and $M^{\mathbf{f}}$ is a maximal M-consistent unfounded set for P w.r.t. M.

I. Prove that $M^{\mathbf{t}}$ is a minimal total model of P^M. First observe that $P^M \subset \mathrm{Tr}(P)^N$. Let us partition $\mathrm{Tr}(P)^N$ into two disjoint sets $\Phi_1 = P^M$ and $\Phi_2 = \mathrm{Tr}(P)^N - P^M$. Now suppose $M^{\mathbf{t}}$ is not a minimal total model of P^M. This leads to one of the two possibilities

- $M^{\mathbf{t}}$ is not a total model of P^M. Since $P^M \subset \mathrm{Tr}(P)^N$, by the CEs, it follows that N is not a model of $\mathrm{Tr}(P)^N$, contradicting the assumption that N is a stable model of $\mathrm{Tr}(P)$.
- $M^{\mathbf{t}}$ is a model of P^M but not minimal. Then, there exists $S \subset M^{\mathbf{t}}$ which is a (total) model of P^M. We construct a total interpretation: $I = S \cup \{a^\bullet \mid a^\bullet \in N\}$. By $S \subset M^{\mathbf{t}}$, and by the CEs, we have $I \subset N^{\mathbf{t}}$ (i.e., only some marked atoms in $N^{\mathbf{t}}$ are not in I). Since $\Phi_1 = P^M$ and S is a model of P^M, every rule in Φ_1 is satisfied by I. Since the marked atoms in I are precisely those in N, any rule $r \in \Phi_2$ is also satisfied in I. Hence I is a model of $\mathrm{Tr}(P)^N$. It follows from $I \subset N^{\mathbf{t}}$ that $M^{\mathbf{t}}$ is not a minimal model of $\mathrm{Tr}(P)^N$, therefore not a stable model of P. A contradiction.

II. Prove that $M^{\mathbf{f}}$ is a maximal M-consistent unfounded set for P w.r.t. M. Since N is a model of

$\mathrm{Tr}(P)$, $N^{\mathbf{f}}$ satisfies $\{a^\bullet \leftarrow a \mid a \in \mathrm{Hb}(P)\}$. $N^{\mathbf{f}}$ is also M-consistent since $N^{\mathbf{f}} \cap M^{\mathbf{t}} = \emptyset$. It follows from Lemma A.3 that $M^{\mathbf{f}}$ is an M-consistent unfounded set for P w.r.t. M. We show its maximality by proving the contrapositive: if $M^{\mathbf{f}}$ is not a maximal M-consistent unfounded set for P w.r.t. M, then N cannot be a stable model of P.

Assume $M^{\mathbf{f}}$ is not maximal. Then there is an M-consistent unfounded set X for P w.r.t. M such that $X \supset M^{\mathbf{f}}$. That is, there is a such that $a \in X$ but $a \notin M^{\mathbf{f}}$. That $a \notin M^{\mathbf{f}}$ means $a \in M^{\mathbf{t}}$ or $a \in M^{\mathbf{u}}$. In both cases, by the CEs, $a^\bullet \in N^{\mathbf{t}}$, i.e., $a^\bullet \notin N^{\mathbf{f}}$. We construct a set $X' = \{a, a^\bullet \mid a \in X\}$. According to Lemma A.5, that X is an M-consistent unfounded set for P w.r.t. M implies that X' is an unfounded set for $\mathrm{Tr}(P)$ w.r.t N. However, $a \in X$ implies $a^\bullet \in X'$ but $a^\bullet \notin N^{\mathbf{f}}$. Thus $a^\bullet \in N^{\mathbf{t}}$ indicating that N is not unfounded free for $\mathrm{Tr}(P)$. Consequently, N is not a stable model of $\mathrm{Tr}(P)$ by Theorem 2.4. $\square$

Theorem A.8 *If M is a partial stable model of P, then N is a stable model of* $\mathrm{Tr}(P)$.

Proof. Since M is a partial stable model of P, we know by Theorem 2.5 that (i) $M^{\mathbf{t}}$ is a minimal model of P^M and (ii) $M^{\mathbf{f}}$ is a maximal M-consistent unfounded set for P w.r.t. M. We note again that it follows by the CEs that $\mathrm{Tr}(P)^N = P^M \cup P_m$ where P_m is the union of the sets of rules $\{a^\bullet \leftarrow a \mid a \in \mathrm{Hb}(P)\}$ and

$$\{H(r_{p'}) \leftarrow B^+(r_{p'}) \mid r_{p'} \in \mathrm{Tr}(P), M(B^-(r_{p'})) = \mathbf{t}\}.$$

That is, $\mathrm{Tr}(P)^N$ can be partitioned into two disjoint sets of rules where P^M consists of the rules that only involve unmarked atoms and P_m consists of the rest of the rules with marked atoms in their heads. Similarly, N can be partitioned into two disjoint subsets, N_m and N_u, where N_m consists of the marked atoms in N and N_u the unmarked atoms in N.

Note that $M^{\mathbf{t}}$ is the total model obtained from M by making the undefined false. This is precisely N_u (which contains those atoms that are either false or undefined in M). Then, when $M^{\mathbf{t}}$ is a minimal model of P^M, it is precisely the case that N_u is a minimal model of the subset P^M of $\mathrm{Tr}(P)^N$.

Assume that N is not a stable model of $\mathrm{Tr}(P)$. Equivalently, N is not unfounded free for $\mathrm{Tr}(P)$. Then, there is an unfounded set X for $\mathrm{Tr}(P)$ w.r.t. N such that $X \cap N^{\mathbf{t}} \neq \emptyset$. Without loss of generality, assume X is maximal. From Proposition A.1, we know that X is a maximal unfounded set for $\mathrm{Tr}(P)^N$ w.r.t. N. Since N_u is a minimal model of the subset P^M of $\mathrm{Tr}(P)^N$, and since P^M is independent of P_m, it is clear that ${N_u}^{\mathbf{f}} \subseteq X$. In addition, any atom that is in $X \cap N^{\mathbf{t}}$ must be marked. We show that this implies that $M^{\mathbf{f}}$ cannot be a maximal M-consistent unfounded set for P w.r.t. M, generating a contradiction.

Let $a^\bullet \in X \cap N^{\mathbf{t}}$. We define a set

$$S_X = \{a, a^\bullet \mid a^\bullet \in N^{\mathbf{f}}\} \cup \{a \mid a^\bullet \in X \text{ and } a \in M^{\mathbf{u}}\}$$

Note that $a \in M^{\mathbf{u}}$ implies $a^\bullet \in N^{\mathbf{t}}$. Further, one can verify easily that $a^\bullet \in X$ implies $a \in X$ (as X is unfounded, for $a^\bullet \leftarrow a \in \mathrm{Tr}(P)$, if $a \in N^{\mathbf{f}}$, by ${N_u}^{\mathbf{f}} \subseteq X$, we get $a \in X$; otherwise the application of UF2 requires $a \in X$). Thus, if $a \in S_X$ then $a \in M^{\mathbf{f}}$ or $a \in M^{\mathbf{u}}$, i.e., $S_X \cap M^{\mathbf{t}} = \emptyset$, hence S_X is consistent with M. From the assumption $\exists a^\bullet \in X \cap N^{\mathbf{t}}$, we have $M^{\mathbf{f}} \subset S_X$. By Lemma A.3, that X is an unfounded set for $\mathrm{Tr}(P)$ w.r.t. N implies that that S_X is an unfounded set for P w.r.t. M. In addition, S_X is M-consistent. However, from $M^{\mathbf{f}} \subset S_X$ we conclude that $M^{\mathbf{f}}$ is not a maximal M-consistent unfounded set for P w.r.t. M.

Thus N must be a stable model of $\mathrm{Tr}(P)$. $\square$