Journal of Logic and Computation (JLC), 1997

Deterministic and Non-Deterministic Stable Models

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

Carlo Zaniolo
CS Dept.
University of California
LosAngeles, CA 90024, USA
zaniolo@cs.ucla.edu

Abstract

Stable models have been first introduced in the domain of total interpretations (*T-stable* models), where the existence of multiple T-stable models for the same program provides a powerful mechanism to express non-determinism. Stable models have been later extended to the domain of partial interpretations (*P-stable* models). In this paper, we show that the presence of multiple P-stable models need not be a direct manifestation of non-determinism, for it can be instead an expression of assorted degrees of undefinedness. To separate the two factors, non-determinism and undefinedness, this paper introduces the notion of *deterministic* stable models and *strictly non-deterministic* ones. Deterministic stable models form an interesting family, having a lattice structure where the well-founded model serves as the bottom; the top of the lattice, the *maximum deterministic* stable model, resolves differences between any two P-stable models in the family. On the other hand, every two models in a family of strictly non-deterministic P-stable models have unreconcilable differences, so that one must be chosen to the exclusion of the other. One such strictly non-deterministic family is constituted by the T-stable models. The paper characterizes two other interesting families: the maximal stable (*M-stable*) models (i.e., those not contained in any other P-stable model), and the least-undefined stable (*L-stable*) models (i.e., maximal stable models with the minimal set of undefined atoms). The paper studies the properties of models in these classes, and characterizes the computational complexity of finding stable models for the various types of `DATALOG` programs.

Keywords: logic programming, stable models, nondeterministic semantics, `DATALOG`, data complexity

1 Introduction

The problem of providing a formal semantics to logic programs where rules contain negated goals represents a key of research issue in areas such as logic programming, non-monotonic reasoning and deductive databases [22, 6]. Significant progress has been made recently on this topic, largely as a result of a renewed interest in deductive databases [4, 7, 39]. For instance, the concept of *stratified programs* [1, 9, 24, 40] is now regarded as a standard notion, efficiently supported in systems such as NAIL! [39], $\mathcal{LDL}$ [25, 11] and CORAL [30], as part of their "bottom-up" execution strategy. Much of recent research focuses on going beyond the limitations of stratified programs. Toward this goal, the concept of *locally stratified* programs was proposed in [28] followed by the important notions of *well-founded models* [41] and various types of stable models: *T-stable* (*total stable*) models [14], *P-stable* (*partial stable*) models (corresponding to the three-valued stable models of [29], the strongly-founded models of [34], the complete scenarios of [12], and the pure models of [21]), and *M-stable* (*maximal stable*) models (corresponding to the partial stable models of [34], the preferred extensions of [12], the regular models of [43], the maximal pure models of [21] and the maximal stable classes of [5]).

The fact that multiple stable models may exist for the same program has caused some resistance to the idea of adopting stable model semantics as the canonical meaning of a logic program is traditionally based on a unique model. Insistence on a unique model is consistent with the history of the field, where, in the past, progress was made by the definition of a preference relation between alternative minimal models leading to the identification of a canonical one. On the other hand, it has been shown that the existence of several alternative stable models for a logic program is not an actual drawback but it can be exploited to express non-determinism in a purely declarative framework — for instance, as shown in [15, 34], stable models provide a formal declarative semantics to non-deterministic pruning constructs of deductive databases, such as the choice construct of [20]. As discussed in [2, 3], in addition to enable a simpler and more natural writing of programs, non-determinism provides a higher expressive power—for instance, it has been shown that multiple stable models can be used to express $\mathcal{NP}$-hard problems [32, 33, 36]. Observe that a non-deterministic model-based semantics does not necessarily imply a non-deterministic query semantics; for instance, various types of deterministic semantics can be used in the presence of alternative stable models, such as the *possibility* semantics and the *certainty semantics*, which are known in AI as membership and entailment semantics, respectively.

In this paper, we study the conditions under which multiple P-stable models express non-determinism. We shall first show that the presence of several P-stable models could be only an expression of assorted degrees of undefinedness, as in the case where there exists a maximum P-stable model resolving their differences. Non-determinism instead requires that there exists a subset of P-stable models that is a *strictly non-deterministic family*, i.e., any two models in the family have unreconcilable differences, so that one must be chosen to the exclusion of others.

We shall show that strict non-determinism is fully captured by the family of M-stable models, i.e., the existence of multiple M-stable models is a necessary and sufficient condition for non-determinism. As a consequence, M-stable models represent the largest family of P-stable models that is strictly non-deterministic. Obviously any non-singleton subfamily of M-stable models is a strictly non-deterministic family as well; so is the subfamily of T-stable models. Another interesting strictly non-deterministic subfamily of M-stable models is represented by *L-stable* (*least undefined stable*) models, i.e., the M-stable models with the minimal set of undefined atoms. We shall also elucidate the properties of L-stable models that supply them with a very high expressive power, as discussed in [33].

We finally analyze the maximal amount of determinism that is embedded in the family of P-stable models, i.e., we study the family of deterministic P-stable models that ensures determinism at the price of more undefinedness. We find that, for each logic program, this family has the structure of a complete lattice: the bottom of the lattice coincides with the well-founded model, while top of the lattice defines the *maximum deterministic* model—a concept previously unexplored, its interesting semantic properties notwithstanding.

The paper is organized as follows. In Section 2, we review the notion of stable models in the domain of total interpretations, and in Section 3, we analyze the properties of P-stable models that will be relevant in understanding the issue of non-determinism. Our analysis, in Section 4, introduces the notions of deterministic and non-deterministic families of P-stable models. We present the results about non-deterministic families of P-stable models in Section 5 and, in Section 6, we study the family of deterministic P-stable models. In Section 7 we investigate the so-called *data complexity* [42] of recognizing a stable model of a given type and of actually finding it. Finally, in Section 8, we draw the conclusion.

For the sake of presentation the proofs of all lemmata and facts are reported in the appendix.

2 Preliminary Definitions and Total Stable Models

Let us start by recalling basic concepts and notation of logic programming with negation [22].

A *term* is a variable, a constant, or a complex term of the form $f(t_1, ..., t_n)$, where $t_1, ..., t_n$ are terms and f is a *function symbol* with finite arity $n \geq 0$. An *atom* is a formula of the language that is of the form $p(t_1, ..., t_n)$ where p is a predicate symbol of a finite arity $n \geq 0$, and $t_1, ..., t_n$ are terms (*arguments* of the atom). A *literal* is either an atom (*positive literal*) or its negation (*negative* *literal*). An atom A, and its negation, i.e., the literal $\neg A$, are said to be the *complement* of each other. In general, if B is a literal, then $\neg B$ denotes the complement of B.

A *rule* r is a formula of the language of the form

$$Q \leftarrow Q_1, ..., Q_m.$$

where Q is a atom (*head* of the rule) and $Q_1, ..., Q_m$ are literals (*goals* of the rule) representing the *body* of the rule. Let $H(r)$ and $G(r)$ represent, respectively, the head of r and the set of all goals of r. A rule without negative goals (i.e., a Horn clause) is called *positive* and a rule with no goals is called a *fact*.

A *logic program* is a finite set of rules. A logic program is called *positive* when all its rules are positive.

Given a logic program $\mathcal{LP}$, the *Herbrand universe* for $\mathcal{LP}$, denoted $U_{\mathcal{LP}}$, is the set of all possible ground terms recursively constructed by using the constants and the function symbols occurring in $\mathcal{LP}$. If there is no constant in the program then one is added arbitrarily. The *Herbrand Base* of $\mathcal{LP}$, denoted $B_{\mathcal{LP}}$, is the set of all possible ground atoms whose predicate symbols occur in $\mathcal{LP}$ and whose arguments are elements of $U_{\mathcal{LP}}$. If $\mathcal{LP}$ contains function symbols with arity greater than 0 then both $U_{\mathcal{LP}}$ and $B_{\mathcal{LP}}$ are countably infinite; otherwise, they are finite.

A term, atom, literal, rule or program is *ground* if it is free of variables.

A *ground instance* of a rule $r \in \mathcal{LP}$ is a rule obtained from r by replacing every variable $X \in r$ by $\phi(X)$, where ϕ is a mapping from all variables occurring in r to $U_{\mathcal{LP}}$. The set of all ground instances of r are denoted by $ground(r)$; accordingly, $ground(\mathcal{LP})$ denotes $\cup_{r \in \mathcal{LP}} ground(r)$. With a little abuse of definition, also $ground(\mathcal{LP})$ will be called a logic program although it can be countably infinite.

Let X be a set of ground literals; then $\neg X$ denotes the set $\{\neg A | A \in X\}$, and X^+ (resp., X^-) denotes the set of all positive (resp., negative) literals in X. Moreover, $\overline{X}$ denotes all elements of the Herbrand Base which do not occur in X, i.e., $\overline{X} = \{A | A \in B_{\mathcal{LP}}$ and neither A nor $\neg A$ is in $X\}$.

A set of ground literals I is a (*partial*) *interpretation* if $I^+ \cap \neg I^- = \emptyset$. $\overline{I}$ is the set of *undefined* elements; if $\overline{I} = \emptyset$ then I will be called *total*.

The *immediate consequence transformation* $\mathbf{T}_{\mathcal{LP}} : 2^{B_{\mathcal{LP}} \cup \neg B_{\mathcal{LP}}} \to 2^{B_{\mathcal{LP}}}$ is defined as follows:[1] for each $X \subseteq B_{\mathcal{LP}} \cup \neg B_{\mathcal{LP}}$, $\mathbf{T}_{\mathcal{LP}}(X) = \{A | A = H(r), r \in ground(\mathcal{LP})$ and $G(r) \subseteq X\}$.

The transformation $\mathbf{T}_{\mathcal{LP}}$ is monotone and continuous in the complete lattice $\langle 2^{B_{\mathcal{LP}} \cup \neg B_{\mathcal{LP}}}, \subseteq \rangle$ and, then, the least fixpoint of $\mathbf{T}_{\mathcal{LP}}$, denoted by $\mathbf{T}^{\infty}_{\mathcal{LP}}(\emptyset)$,

[1] Observe that the immediate consequence transformation is traditionally defined in the domain of interpretations.

exists and coincides with

$$\mathbf{T}^{\omega}_{\mathcal{LP}}(\emptyset) = \bigcup_{i<\omega} \mathbf{T}^{i}_{\mathcal{LP}}(\emptyset),$$

where ω is the first limit ordinal after 0, $\mathbf{T}^{0}_{\mathcal{LP}}(\emptyset) = \emptyset$, and for each $i > 0$, $\mathbf{T}^{i}_{\mathcal{LP}}(\emptyset) = \mathbf{T}_{\mathcal{LP}}(\mathbf{T}^{i-1}_{\mathcal{LP}}(\emptyset))$ [38, 22].

Definition 1 (Total Models) A total interpretation M of a logic program $\mathcal{LP}$ is a *total model* of $\mathcal{LP}$ if $M^+ \supseteq \mathbf{T}_{\mathcal{LP}}(M)$; moreover, M is a *minimal model* if there exists no other total model N of $\mathcal{LP}$ such that N^+ is a proper subset of M^+.

Total models can be also characterized in terms of the negative literals using the well-known concept of unfounded set.[2]

Given $X \subseteq (B_{\mathcal{LP}} \cup \neg B_{\mathcal{LP}})$ and $Y \subseteq B_{\mathcal{LP}}$, Y is an *unfounded set w.r.t.* X if for each rule $r \in ground(\mathcal{LP})$ with $H(r)$ in Y, $G(r) \cap (\neg X \cup Y) \neq \emptyset$, i.e., the body is false or contains some element in Y. The union of any two unfounded sets w.r.t. X is an unfounded set w.r.t. X as well; the union of all unfounded sets w.r.t. X, denoted by $\mathbf{U}_{\mathcal{LP}}(X)$, is called the *greatest unfounded set.* Notice that an unfounded set w.r.t. X is an unfounded set also w.r.t. every superset of X; so $\mathbf{U}_{\mathcal{LP}}$ is a monotonic transformation from $2^{B_{\mathcal{LP}} \cup \neg B_{\mathcal{LP}}}$ to $2^{B_{\mathcal{LP}}}$.

Fact 2 A total interpretation I of a logic program $\mathcal{LP}$ is a total model of $\mathcal{LP}$ if and only if $\neg I^-$ is an unfounded set w.r.t. I.

It is well-known that a positive logic program has a unique minimal model (the *least* total model) which represents its natural meaning and is equal to $M_T = \mathbf{T}^{\infty}_{\mathcal{LP}}(\emptyset) \cup \neg(B_{\mathcal{LP}} - \mathbf{T}^{\infty}_{\mathcal{LP}}(\emptyset))$. In case of non-positive programs, the existence of a least total model is not guaranteed as M_T is not necessarily a model. Therefore, the issue of what model to base the semantics of the logic program on becomes much more complex. Total stable models, introduced in [14] and defined next, represents an interesting solution to the problem.

The definition of total stable model is based on the principle of *foundness* (also called "justification" in [43]), i.e., every positive conclusion must be derivable by the rules possibly using negative literals as additional axioms. The original foundness condition formulated in [14] can be rephrased as follows in the domain of partial interpretations.

Let $\mathcal{LP}$ a logic program and I be an interpretation for $\mathcal{LP}$. Let $pos(\mathcal{LP}, I)$ denote the positive program obtained from $ground(\mathcal{LP})$ by deleting (a) each rule r such that $H(r) \in \overline{I}$ or $\neg(G(r))^- \cap I^+ \neq \emptyset$ (i.e., the head is undefined

[2] Also here the domain is a complete lattice instead of the family of all interpretations as in the original definition.

or some negative goal is false) and (b) all negative goals from the remaining rules. Then I is *pre-founded* if $T^{\infty}_{pos(\mathcal{LP},I)}(\emptyset) = I^+$, thus all elements in I^+ are justified. The notion of foundness is obtained by also requiring that every derivable element be in I^+; therefore, I is *founded* if it is pre-founded and $I^+ \supseteq \mathbf{T}_{\mathcal{LP}}(I)$.

FACT 3 Let $\mathcal{LP}$ be a logic program and I be an interpretation for $\mathcal{LP}$. Then (i) if I is pre-founded then $I^+ \subseteq \mathbf{T}_{\mathcal{LP}}(I)$ and (ii) if I is founded then $I^+ = \mathbf{T}_{\mathcal{LP}}(I)$.

The two notions coincide in the domain of total interpretations.

FACT 4 Let $\mathcal{LP}$ be a logic program and I be a total pre-founded interpretation for $\mathcal{LP}$. Then I is founded.

By definition a total founded interpretation I is indeed a total model as $I^+ \supseteq \mathbf{T}_{\mathcal{LP}}(I)$.

DEFINITION 5 (T-STABLE MODELS) A total founded interpretation of a program $\mathcal{LP}$ is called a *total stable* model for $\mathcal{LP}$.

As shown in [14], T-stable models are minimal models. Also T-stable models can be characterized in terms of unfounded sets.

FACT 6 A total interpretation I of a logic program $\mathcal{LP}$ is a T-stable model if and only if $\neg I^- = \mathbf{U}_{\mathcal{LP}}(I)$.

A logic program may have zero, one or several T-stable models. The family of T-stable models for a logic program $\mathcal{LP}$ is denoted by $\mathcal{TS}_{\mathcal{LP}}$ or, simply, by $\mathcal{TS}$ if $\mathcal{LP}$ is understood.

EXAMPLE 7 Consider the following logic program having 0-arity predicate symbols:

$$\begin{array}{lll} \mathtt{a} & \leftarrow & \neg\mathtt{b}. \\ \mathtt{u} & \leftarrow & \mathtt{b}. \\ \mathtt{u} & \leftarrow & \mathtt{a}, \neg\mathtt{v}. \\ \mathtt{v} & \leftarrow & \mathtt{a}, \neg\mathtt{u}. \end{array}$$

There are five total models: $M_1 = \{a, \neg b, u, \neg v\}$, $M_2 = \{a, \neg b, \neg u, v\}$, $M_3 = \{a, \neg b, u, v\}$, $M_4 = \{\neg a, b, u, \neg v\}$, and $M_5 = \{\neg a, b, u, v\}$. But only M_1 and M_2 are T-stable.

The fact that, in the domain of total interpretations, there might be multiple stable models to choose from, suggests that the notion of non-determinism is implicit in the stable model semantics. However, for a thorough treatment of the subject we need to move to the domain of partial interpretations.

3 Partial Stable Models

Given an interpretation I, if I is not total then the fact that $I^+ \supseteq \mathbf{T}_{\mathcal{LP}}(I)$ no longer implies that $\neg I^-$ is an unfounded set w.r.t. I and conversely. Therefore, both conditions must be explicitly enforced to get the notion of partial model.

Definition 8 (Partial Models) An interpretation I of a logic program $\mathcal{LP}$ is a *(partial) model* for $\mathcal{LP}$ if both $I^+ \supseteq \mathbf{T}_{\mathcal{LP}}(I)$ and $\neg I^-$ is an unfounded set w.r.t. I.

Fact 9 Every partial model of a program $\mathcal{LP}$ is a subset of some total model.

Note that the definition of partial model is stricter than that used in [41], which would accept any subset of some total model. Also observe that partial models coincides with the *three-valued models* of [29].

Example 10 Consider the following program:

$$\begin{array}{lcl} \texttt{p(a)} & \leftarrow & \neg\texttt{p(b)}. \\ \texttt{p(c)} & \leftarrow & \texttt{p(b)}. \\ \texttt{p(d)} & \leftarrow & \neg\texttt{p(d)}. \end{array}$$

$X = \{p(a), \neg p(b), \neg p(c)\}$ is a partial model as $\{p(b), p(c)\}$ is an unfounded set w.r.t. X and $\{p(a)\} = \mathbf{T}_{\mathcal{LP}}(X)$. Note that X is a subset of the total minimal model $M = \{p(a), \neg p(b), \neg p(c), p(d)\}$. $Y = \{p(a), \neg p(c)\}$ is also a subset of M but is not a partial model as $\{p(c)\}$ is not an unfounded set w.r.t. Y.

As for models, the notion of T-stable model is extended to the domain of partial interpretations by enforcing, in addition to foundness for positive literals, the condition that the negative literals be the greatest unfounded set.

Definition 11 (P-Stable Models) Let $\mathcal{LP}$ be a logic program and M a partial model of it. Then M is *P-stable* if (i) it is founded and (ii) $\neg M^- = \mathbf{U}_{\mathcal{LP}}(M)$.

As shown in [35], the concept of P-stable model is equivalent to that of three-valued stable model [29], of strongly-founded model [34], of complete scenario [12] and of pure model [21]. Definition 11 shows that this important concept can be effectively captured in the framework of two-valued logic. In fact, as also pointed out in [21], P-stable models can share the same concise representation as total models which only stores positive ground literals: total models assume false every ground atom that is not defined true while P-stable models assign a false value only to the elements in the greatest unfounded set.

We denote the family of all P-stable models for a given logic program $\mathcal{LP}$ by $\mathcal{PS}_{\mathcal{LP}}$ or, simply, $\mathcal{PS}$ if $\mathcal{LP}$ is understood. Obviously $\mathcal{TS} \subseteq \mathcal{PS}$, i.e., a T-stable model is also P-stable, but a P-stable model may be neither T-stable nor a subset of some T-stable model.

EXAMPLE 12 Take the program in Example 10. We have that the $\{p(a), \neg p(b), \neg p(c)\}$ is the unique P-stable model and is not T-stable. The partial model $\{\neg p(b), \neg p(c), p(d)\}$ does not satisfy the condition (i) of Definition 11 and the partial model $\{p(a), \neg p(b)\}$ does not satisfy the condition (ii).

For the program of Example 7, we have three P-stable models: one is the empty set, the others are the T-stable models M_1 and M_2.

We next show that the P-stable models of a given program $\mathcal{LP}$ are fixpoints of the transformation $\mathbf{W}_{\mathcal{LP}}$ in the complete lattice $\langle 2^{B_{\mathcal{LP}} \cup \neg B_{\mathcal{LP}}}, \subseteq \rangle$, defined in [41] as: for each $X \subseteq B_{\mathcal{LP}} \cup \neg B_{\mathcal{LP}}$, $\mathbf{W}_{\mathcal{LP}}(X) = \mathbf{T}_{\mathcal{LP}}(X) \cup \neg \mathbf{U}_{\mathcal{LP}}(X)$.[3] Furthermore, let α be any countable ordinal. We recursively define $\mathbf{W}^{\alpha}_{\mathcal{LP}}(X)$ as

1. X if $\alpha = 0$;
2. $\mathbf{W}_{\mathcal{LP}}(\mathbf{W}^{\gamma+1}_{\mathcal{LP}}(X))$ if α is a successor ordinal $\gamma + 1$;
3. $\cup_{\gamma<\alpha} \mathbf{W}^{\gamma}_{\mathcal{LP}}(X)$ if α is a limit ordinal different from 0.

Finally, we define $\mathbf{W}^{\infty}_{\mathcal{LP}}(X) = \cup_{\alpha} \mathbf{W}^{\alpha}_{\mathcal{LP}}(X)$, where α ranges over all countable ordinals. As both $\mathbf{T}_{\mathcal{LP}}$ and $\mathbf{U}_{\mathcal{LP}}$ are monotonic, $W_{\mathcal{LP}}$ is a monotonic transformation in a complete lattice and, then, there exists the least fixpoint $\mathbf{W}^{\infty}_{\mathcal{LP}}(\emptyset)$ of $\mathbf{W}_{\mathcal{LP}}$ [38, 22]. Moreover, as $B_{\mathcal{LP}} \cup \neg B_{\mathcal{LP}}$ is a countable set, there exists a countable ordinal α for which $\mathbf{W}^{\alpha}_{\mathcal{LP}}(\emptyset) = \mathbf{W}^{\alpha+1}_{\mathcal{LP}}(\emptyset) = \mathbf{W}^{\infty}_{\mathcal{LP}}(\emptyset)$. The least fixpoint of $W_{\mathcal{LP}}$ is called the *well-founded model* [41] and is actually a P-stable model.

Observe now that, as shown in the next example, $\mathbf{W}_{\mathcal{LP}}$ is not continuous. Therefore, there are cases (actually very artificial, as discussed in [31, 41]), where $\mathbf{W}^{\infty}_{\mathcal{LP}}(\emptyset)$ is different from $\mathbf{W}^{\omega}_{\mathcal{LP}}(\emptyset)$.

EXAMPLE 13 Consider the following program:

$$
\begin{array}{lcl}
\mathtt{wins(X)} & \leftarrow & \neg\mathtt{looses(X)}. \\
\mathtt{looses(X)} & \leftarrow & \mathtt{move(Y,X)}, \neg\mathtt{wins(Y)}. \\
\mathtt{move(s(X),0).} & & \\
\mathtt{move(s(X),s(s(X))).} & &
\end{array}
$$

The Herbrand universe is the set of natural numbers, 0 has infinitely many incoming moves, but each one descends finitely to $s(0)$, which has no incoming move. We have that for each natural i, $i > 0$, $\neg looses(i)$ and $wins(i)$ are

[3]Note that, in the original definition of [41], $\mathbf{W}_{\mathcal{LP}}$ is not closed in the sense that the domain is the family of all interpretations while the range is the family of all sets of ground literals. In our definition, domain and range coincide and each of them forms a complete lattice.

in $M = \mathbf{W}^{\omega}_{\mathcal{LP}}(\emptyset)$, whereas $looses(0)$ and $wins(0)$ are in $\overline{M}$. On the other hand, $\neg looses(0)$ is in $\mathbf{W}_{\mathcal{LP}}(M) = \mathbf{W}^{\omega+1}_{\mathcal{LP}}(\emptyset)$, thus $\mathbf{W}_{\mathcal{LP}}$ is not continuous. Moreover, $wins(0)$ is in $\mathbf{W}_{\mathcal{LP}}(\mathbf{W}_{\mathcal{LP}}(M)) = \mathbf{W}^{\omega+2}_{\mathcal{LP}}(\emptyset)$, which turns out to be the least fixpoint of $\mathbf{W}_{\mathcal{LP}}$.

PROPOSITION 14 Every P-stable model of a logic program $\mathcal{LP}$ is a fixpoint of $\mathbf{W}_{\mathcal{LP}}$.

PROOF. Since M is a P-stable model, M is founded and $M^- = \neg\mathbf{U}_{\mathcal{LP}}(M)$. By Fact 3, $M^+ = \mathbf{T}_{\mathcal{LP}}(M)$. It follows that $M = \mathbf{W}_{\mathcal{LP}}(M)$.

Proposition 14 has been proved in [41] for the case of total interpretations. As shown in the next example, the condition that a partial model is a fixpoint of $\mathbf{W}_{\mathcal{LP}}$ is not sufficient for it to be P-stable.

EXAMPLE 15 Consider the following program $\mathcal{LP}$:

```
p ← ¬p.
a ← p.
a ← b.
b ← a.
```

The partial model $M = \{a, b\}$ is a fixpoint of $\mathbf{W}_{\mathcal{LP}}$ but is not P-stable. Observe that M is also a partial model for the program $\mathcal{LP}'$ consisting of the last two rules above, but M is not a fixpoint for $\mathbf{W}_{\mathcal{LP}'}$; in fact $N = \mathbf{W}_{\mathcal{LP}'}(M) = \{a, b, \neg a, \neg b\}$. Note that N is not an interpretation although it is a fixpoint of $\mathbf{W}_{\mathcal{LP}'}$.

As shown in the previous example, a fixpoint of $\mathbf{W}_{\mathcal{LP}}$ is not necessarily an interpretation. But, as proven next, if a fixpoint of $\mathbf{W}_{\mathcal{LP}}$ is an interpretation, then it is also a model although not necessarily P-stable; to achieve P-stability it is sufficient to require pre-foundness.

PROPOSITION 16 Let $\mathcal{LP}$ be a logic program and let I be an interpretation of $\mathcal{LP}$ such that $I = \mathbf{W}_{\mathcal{LP}}(I)$. Then

1. I is a model of $\mathcal{LP}$, and
2. if I is pre-founded then I is P-stable.

PROOF. (1) Since $I = \mathbf{W}_{\mathcal{LP}}(I)$, $I^+ = \mathbf{T}_{\mathcal{LP}}(I)$ and $\neg I^- = \mathbf{U}_{\mathcal{LP}}(I)$ by definition of $\mathbf{W}_{\mathcal{LP}}$. Hence, as I is an interpretation by hypothesis, I is a model.

(2) Suppose now that I is pre-founded. As $I^+ = \mathbf{T}_{\mathcal{LP}}(I)$ by definition of fixpoint for $\mathbf{W}_{\mathcal{LP}}$, I is also founded. Therefore, as also $\neg X^- = \mathbf{U}_{\mathcal{LP}}(X)$, X is P-stable.

4 Deterministic and Non-Deterministic Models

In this section we clarify and formalize the concept of non-determinism in logic programs with negation. In particular, we show that the presence of multiple P-stable models is a necessary condition, but not sufficient condition for non-determinism in the domain of partial interpretations. The following definitions are useful for a precise characterization of the issues.

Definition 17 (Mutually Exclusive Interpretations) Two interpretations I_1 and I_2 of a logic program $\mathcal{LP}$ are *mutually exclusive* if $I_1 \cup I_2$ is not an interpretation.

Definition 18 (Deterministic and Non-deterministic Families) Let $\mathcal{LP}$ be logic program and $\mathcal{F}$ be a family of interpretations of $\mathcal{LP}$. Then $\mathcal{F}$ is *deterministic* if no two distinct interpretations in $\mathcal{F}$ are mutually exclusive; otherwise, $\mathcal{F}$ is said to be *non-deterministic.* A non-deterministic family $\mathcal{F}$ is *strictly non-deterministic* if every two distinct interpretations in $\mathcal{F}$ are mutually exclusive.

Only a strictly non-deterministic family is a "pure" expression of non-determinism as any two elements in the family have unreconcilable differences — thus one must be chosen to the exclusion of others. In a generic family, we instead find both deterministic and non-deterministic subfamilies. The following definition can be used to classify the elements of such a family in two subclasses:

Definition 19 (Deterministic&Non-Deterministic Interpretations) Let $\mathcal{LP}$ be a program and $\mathcal{F}$ be a family of interpretations of $\mathcal{LP}$. An interpretation I in $\mathcal{F}$ will be called *$\mathcal{F}$-deterministic* if no interpretation in $\mathcal{F}$ is mutually exclusive with I; otherwise I is called *$\mathcal{F}$-non-deterministic.*

Clearly, given any family $\mathcal{F}$, the subfamily of all $\mathcal{F}$-deterministic interpretations is a deterministic family.

We next prove that the family $\mathcal{PS}$ of all P-stable models is never strictly non-deterministic, thus "pure" non-determinism can be captured only by suitable subfamilies of $\mathcal{PS}$. Before proving this result, we need a weaker definition of stability and two technical lemmata.

Definition 20 (Pre-stable interpretations) An interpretation I of a logic program $\mathcal{LP}$ is *pre-stable* if both I is pre-founded and $\neg I^-$ is an unfounded set w.r.t. I.

Lemma 21 Let $\mathcal{F}$ be a deterministic family of pre-stable interpretations of a logic program $\mathcal{LP}$. Let I be the union of all interpretations in $\mathcal{F}$. Then I is a pre-stable interpretation of $\mathcal{LP}$.

LEMMA 22 Let $\mathcal{LP}$ be a program and M be a pre-stable interpretation of $\mathcal{LP}$. Then

1. $\mathbf{W}^{\alpha}_{\mathcal{LP}}(M)$, for each countable ordinal α, is a monotonic sequence of pre-stable interpretations, and
2. $\mathbf{W}^{\infty}_{\mathcal{LP}}(M)$ is a P-stable model of $\mathcal{LP}$.

We note that Part (1) of Lemma 22 has been proved in [41] for the case that $M = \emptyset$. We have generalized the result to cope with fixpoints beyond the least fixpoint. The lemma can be used for a simple proof of the following well-known properties of the well-founded model, defined as the least fixpoint of $\mathbf{W}_{\mathcal{LP}}$[41].

PROPOSITION 23 Given a program $\mathcal{LP}$, its well-founded model M has the following properties:

1. M is a P-stable model of $\mathcal{LP}$,
2. M is the intersection of all P-stable models of $\mathcal{LP}$.
3. M is the least $\mathcal{PS}$-deterministic model of $\mathcal{LP}$,

PROOF. Observe that $\emptyset$ is a pre-founded interpretation of $\mathcal{LP}$; therefore, $M = \mathbf{W}^{\infty}_{\mathcal{LP}}(\emptyset)$ is in $\mathcal{PS}$ by Lemma 22. Moreover, since every $N \in \mathcal{PS}$ is a fixpoint of $W_{\mathcal{LP}}$ by Proposition 14 and M is the least fixpoint of $\mathbf{W}_{\mathcal{LP}}$, $M \subseteq N$, i.e., M is the intersection of all P-stable models of $\mathcal{LP}$. Finally, as M is contained in every P-stable model, M is the minimum P-stable model and is mutually exclusive with none of them, thus M is also the least $\mathcal{PS}$-deterministic model.

Parts (1) and (2) of the above proposition have been first proved in [41] and in [29], respectively.

PROPOSITION 24 Given a logic program $\mathcal{LP}$, $\mathcal{PS}$ is neither empty nor strictly non-deterministic.

PROOF. As $M = \mathbf{W}^{\infty}_{\mathcal{LP}}(\emptyset)$ exists by definition of $\mathbf{W}_{\mathcal{LP}}$ and M is in $\mathcal{PS}$ by Part (1) of Proposition 23, $|\mathcal{PS}| \geq 1$. Moreover, by Part (3) of Proposition 23, M is $\mathcal{PS}$-deterministic and, then, $\mathcal{PS}$ is not strictly non-deterministic.

EXAMPLE 25 Consider the following program:

```
b      ←  a(1), ¬c.
c      ←  a(1), ¬b.
p      ←  b, ¬p.
d      ←  a(2), ¬p, ¬e.
e      ←  a(2), ¬p, ¬d.
q      ←  ¬d, ¬q.
a(1).
a(2).
```

The P-stable models are: $M_1 = \{a(1), a(2)\}$, $M_2 = \{a(1), a(2), b, \neg c\}$, $M_3 = \{a(1), a(2), \neg b, c, \neg p\}$, $M_4 = \{a(1), a(2), \neg b, c, \neg p, \neg d, e\}$, $M_5 = \{a(1), a(2), \neg b, c, \neg p, d, \neg e, \neg q\}$. M_1 is the well-founded model and is contained in every P-stable model. M_2, M_4 and M_5 are mutually exclusive; M_3 is mutually exclusive with M_2 but not with M_4 and M_5.

We shall discuss non-deterministic P-stable models in the next section, and we shall return to deterministic P-stable models in Section 6.

5 Non-Deterministic P-Stable Models

Having recognized that the family of P-stable models is never strictly non-deterministic, it is necessary to identify the maximal subfamily that has this property. To this end we introduce the following definition.

DEFINITION 26 (M-STABLE MODELS) Let $\mathcal{LP}$ be a logic program. A P-stable model of $\mathcal{LP}$ is *M-stable* (*maximally stable*) if it is not a proper subset of any other P-stable model of $\mathcal{LP}$.

EXAMPLE 27 The M-stable models of the program of Example 25 are M_2, M_4 and M_5.

M-stable models correspond to the (partial) stable models of [34], the preferred extensions of [12] (as proven in [18]), the regular models of [43], the maximal pure models of [21] and the maximal stable classes of [5] (as shown in [43]). The family of all M-stable models for a program $\mathcal{LP}$ is denoted by $\mathcal{MS}_{\mathcal{LP}}$ or, simply, by $\mathcal{MS}$ if $\mathcal{LP}$ is understood; as shown next, MS is never empty.

THEOREM 28 For each logic program $\mathcal{LP}$, $\langle \mathcal{PS}, \subseteq \rangle$ is a Noetherian meet semilattice,[4] whose bottom element is the well founded model for $\mathcal{LP}$ and whose top elements are all M-stable models for $\mathcal{LP}$.

[4] A meet (i.e., lower) semilattice is Noetherian if does not contain any infinitely ascending chain, i.e., every chain has a top element.

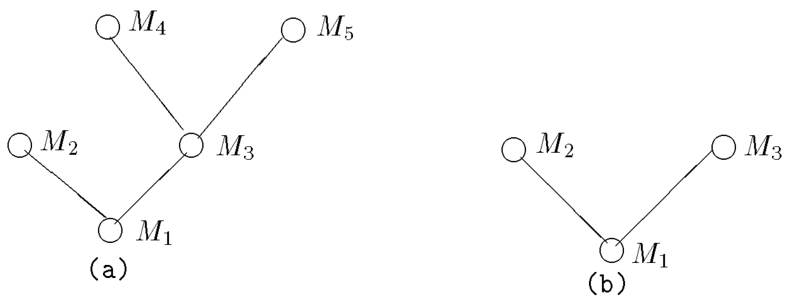

Figure 1: *Semilattices of P-stable Models*

PROOF. Obviously, $\langle \mathcal{PS}, \subseteq \rangle$ is a partial order. Let $\mathcal{X}$ be any non-empty subset of $\mathcal{PS}$. We first prove that the greatest lower bound for $\mathcal{X}$, denoted by $glb(\mathcal{X})$, exists. Let $\mathcal{F}$ be the family of all lower bounds for $\mathcal{X}$, i.e., $\mathcal{F} = \{M | M \in \mathcal{PS}$ and $M \subseteq N$ forall $N \in \mathcal{X}\}$. By Proposition 23, the well-founded model is in $\mathcal{F}$; so $\mathcal{F}$ is not empty. Moreover, any two models in $\mathcal{F}$ are not mutually exclusive as both are subsets of a same model. Hence, $\mathcal{F}$ is a deterministic family of P-stable models; therefore, as P-stable models are also pre-stable, the union of all models in $\mathcal{F}$, say U, is a pre-stable interpretation by Lemma 21. By Part (2) of Lemma 22, $S = \mathbf{W}^{\infty}_{\mathcal{LP}}(U)$ is a P-stable model. Moreover, for each M in $\mathcal{F}$, $M \subseteq S$ since (i) $M = \mathbf{W}_{\mathcal{LP}}(M)$ by Proposition 14, (ii) $M \subseteq U$ by construction and (iii) $U \subseteq S$ by Part (1) of Lemma 22; thus S is an upper bound for $\mathcal{F}$. Consider now any N in $\mathcal{X}$. By construction, $U \subseteq N$. Hence, since $\mathbf{W}_{\mathcal{LP}}$ is monotonic, $N = \mathbf{W}_{\mathcal{LP}}(N)$ by Proposition 14 and $S = \mathbf{W}^{\infty}_{\mathcal{LP}}(U)$, it follows that $S \subseteq N$, thus S is a lower bound for $\mathcal{X}$. Therefore, $S = glb(\mathcal{X})$ and, then, $\langle \mathcal{PS}, \subseteq \rangle$ is a meet semilattice and the bottom element is the well-founded model by Proposition 23. To see that it is actually Noetherian, consider any maximal chain of it; we prove that the top element exists. Let $\mathcal{X}$ be the family of all models in the chain. Obviously $\mathcal{X}$ is a deterministic family of P-stable models; therefore, the union of all models in $\mathcal{X}$, say U, is a pre-stable interpretation by Lemma 21. By Part (2) of Lemma 22, $S = \mathbf{W}^{\infty}_{\mathcal{LP}}(U)$ is a P-stable model and by Part (1) of the same lemma $N \subseteq S$. It follows, that S is the top element in the chain and is M-stable. This concludes the proof.

COROLLARY 29 For every logic program $\mathcal{LP}$, $|PS| \geq |MS| \geq 1$.

PROOF. It immediately follows from Proposition 23 and Theorem 28.

The fact that P-stable models form a semilattice was also proved in [12] (see Theorem 3 in that paper).

Example 30 The semilattice of the P-stable models of the program of Example 25 is shown in Figure 1(a).

We next show that $\mathcal{MS}$ captures non-determinism because the union of any two M-stable models is not an interpretation.

Theorem 31 For every logic program $\mathcal{LP}$, any two distinct models in $\mathcal{MS}$ are mutually exclusive.

Proof. Let M_1, M_2 be any two distinct M-stable models of $\mathcal{LP}$. We show that M_1 and M_2 are mutually exclusive by contradiction. Then assume that they are not, thus $\{M_1, M_2\}$ is a deterministic family of P-stable models models of $\mathcal{LP}$. Therefore, $M_1 \cup M_2$, is a pre-stable interpretation by Lemma 21 and $N = \mathbf{W}^{\infty}_{\mathcal{LP}}(M_1 \cup M_2)$ is a P-stable model by Part (2) of Lemma 22. Moreover, by Part (1) of Lemma 22 $M_1 \cup M_2 \subseteq N$. By definition of M-stable model, neither $M_1 \subset M_2$ nor $M_2 \subset M_1$. Hence, both $M_1 \subset N$ and $M_2 \subset N$ — a contradiction with the fact that M_1 and M_2 are M-stable, i.e., they are not contained in any other P-stable model. It follows that M_1 and M_2 are mutually exclusive and this concludes the proof.

Corollary 32 For every logic program $\mathcal{LP}$, $\mathcal{PS}$ is a deterministic family if and only if $|\mathcal{MS}| = 1$.

Proof. If $|\mathcal{MS}| > 1$, there exist at least two M-stable models and they are mutually exclusive by Theorem 31. Therefore, as M-stable models are also P-stable by definition, $\mathcal{PS}$ is not a deterministic family. On the other hand, if $|\mathcal{MS}| = 1$ then the semilattice of the P-stable models collapses into a single chain. Therefore, every other P-stable model is a subset of the unique M-stable model; so, no two P-stable models are mutually exclusive, i.e., $\mathcal{PS}$ is a deterministic family.

Every subfamily of $\mathcal{MS}$ with multiple elements is strictly non-deterministic:

Corollary 33 Let $\mathcal{LP}$ be a logic program and $\mathcal{F} \subseteq \mathcal{MS}$. Then, $\mathcal{F}$ is a strictly non-deterministic family if and only if $|\mathcal{F}| > 1$.

Proof. The proof of the only-if part is trivial. Suppose now that $|\mathcal{F}| > 1$. As any two elements in $\mathcal{F}$ are also in $\mathcal{MS}$, they are mutually exclusive by Theorem 31; so $\mathcal{F}$ is a strictly non-deterministic family.

Corollary 33 implies that non-determinism is also present in every subfamily of M-stable models with two or more members. One of such subfamilies is the family of T-stable models, denoted by $\mathcal{TS}$.

PROPOSITION 34 For every logic program $\mathcal{LP}$, $\mathcal{TS} \subseteq \mathcal{MS}$.

PROOF. T-stable models are obviously P-stable; as they are total interpretation, they are also M-stable.

We next introduce another interesting subfamily of M-stable models, called *L-stable* models (for *least-undefined stable* models):

DEFINITION 35 (L-STABLE MODELS) Let $\mathcal{LP}$ a program. A P-stable model M of P is *L-stable* if for every P-stable model N of $\mathcal{LP}$, $\overline{N} \not\subset \overline{M}$, i.e., M leaves undefined a minimal set of elements of the Herbrand base.

Thus L-stable models are the P-stable models whose sets of undefined atoms are minimal. Let $\mathcal{LS}_{\mathcal{LP}}$ denote family of L-stable models for a given program $\mathcal{LP}$; as usual, we shall omit the reference to $\mathcal{LP}$ whenever no confusion arises.

PROPOSITION 36 Let $\mathcal{LP}$ be a program. Then $|\mathcal{LS}| \geq 1$, $\mathcal{LS} \subseteq \mathcal{MS}$, and $(\mathcal{TS} \neq \mathcal{LS}) \equiv (\mathcal{TS} = \emptyset)$.

PROOF. Let M be an L-stable model of $\mathcal{LP}$ and, by contradiction, assume that M is not M-stable. Then there exists a P-stable model N such that $M \subset N$. Hence $\overline{N} \subset \overline{M}$ — a contradiction. So M is indeed M-stable and, then, $\mathcal{LS} \subseteq \mathcal{MS}$. Let us now prove that $\mathcal{LS}$ is not empty. Consider the following partial order for $\mathcal{PS}$: given M and N in $\mathcal{PS}$, $M < N$ if $\overline{N} \subset \overline{M}$. Using the same argument of the proof of Theorem 28, it is now easy to see that the partial order is Noetherian and the top element of every maximal chain is an L-stable model. The existence of at least one chain follows from the existence of the well-founded model for every logic program. Hence, $|\mathcal{LS}| \geq 1$.

Let us now prove that $(\mathcal{TS} \neq \mathcal{LS}) \equiv (\mathcal{TS} = \emptyset)$. If $\mathcal{TS} = \emptyset$ then $TS \neq LS$ since we have shown that $|\mathcal{LS}| \geq 1$. Suppose now that $\mathcal{TS} \neq \mathcal{LS}$; we show that $\mathcal{TS} = \emptyset$ by contradiction. Let us then assume that $\mathcal{TS} \neq \emptyset$. We have that for every M in $\mathcal{TS}$, $\overline{M} = \emptyset$; therefore, M is L-stable and, then, $\mathcal{TS} \subset \mathcal{LS}$. Hence $\mathcal{LS} - \mathcal{TS} \neq \emptyset$. Let M be any L-stable model in $\mathcal{LS} - \mathcal{TS}$ and N be any T-stable model. But, as $\overline{N} = \emptyset$ and $\overline{M} \neq \emptyset$, $\overline{N} \subset \overline{M}$ and, then, M is not L-stable — a contradiction. So $\mathcal{TS} = \emptyset$.

It turns out that for every program that has some T-stable model, $\mathcal{LS} = \mathcal{TS}$: thus L-stability is the most appropriate extension of the notion of T-stability to the domain of partial interpretations. This represents a rather unexpected result, in view of the fact that so many previous authors have instead claimed that M-stable models represent the natural extension of T-stable models.

EXAMPLE 37 Let us now remove the fact $a(2)$ from the program of Example 25. Thus we obtain the program:

```
b     ←   a(1),  ¬c.
c     ←   a(1),  ¬b.
p     ←   b,  ¬p.
d     ←   a(2),  ¬p,  ¬e.
e     ←   a(2),  ¬p,  ¬d.
q     ←   ¬d,  ¬q.
a(1).
```

Given any P-stable model, $a(2)$ and d are false; so q is left undefined. Therefore, there exists no T-stable model. The P-stable models are: $M_1 = \{a(1), \neg a(2), \neg d, \neg e\}$, $M_2 = \{a(1), \neg a(2), \neg d, \neg e, b, \neg c\}$, $M_3 = \{a(1), \neg a(2), \neg d, \neg e, \neg b, c, \neg p\}$. M_1 is the well-founded model; M_2 and M_3 are the M-stable models; M_3 is also L-stable but not T-stable. The semilattice of P-stable models is shown in Figure 1(b).

The relevance of L-stable models is confirmed by an expressive power that is higher than the ones of M-stable and T-stable models as proven in [33].

6 Deterministic P-Stable Models

In the previous section, we have shown that the family of M-stable models captures the whole potential non-determinism of P-stable models and that any non-deterministic semantics for logic programs should refer to this family, or to some desirable subfamily, such as that of T-stable models or L-stable models. In this section, we study the properties of the family of $\mathcal{PS}$-deterministic models.

The family of $\mathcal{PS}$-deterministic models is never empty for it contains the well-founded model. Traditionally, researchers seeking a canonical deterministic semantics for logic programs have focused on the notion of well-founded model, which is both the smallest P-stable model and the intersection of all P-stable models (see Proposition 23). A more general approach consists in taking the agreement set between all its M-stable models, and find P-stable models that belong to this agreement set. We next prove that this is actually a characterization of $\mathcal{PS}$-deterministic models:

THEOREM 38 A P-stable model of a logic program $\mathcal{LP}$ is $\mathcal{PS}$-deterministic if and only if it is contained in the intersection of all M-stable models of $\mathcal{LP}$.

PROOF. Let $\mathcal{LP}$ be a program and M be a P-stable model of it.

(*Only-if part*) Suppose that M is $\mathcal{PS}$-deterministic. We proceed by contradiction by assuming that there exists an M-stable model, say N, such that M is not contained in N. By definition of M-stable model, also N is not contained in M. Moreover, M and N are not mutually exclusive by definition of deterministic model, thus $\{M, N\}$ is a deterministic family of P-stable models. Therefore, $M \cup N$, is a pre-stable interpretation by Lemma 21 and $K = \mathbf{W}^{\infty}_{\mathcal{LP}}(M \cup N)$ is a P-stable model by Part (2) of Lemma 22; moreover, by Part (1) of Lemma 22 $N \cup M \subseteq K$. But M is not contained in N; so $N \subset K$ — a contradiction with the hypothesis that N is M-stable. It follows that M is contained in N.

(*If part*) Suppose that M is contained in every M-stable model of $\mathcal{LP}$. Then M is not mutually exclusive with any of the M-stable models. But any P-stable model is a subset of some M-stable model; therefore, M is not mutually exclusive with any P-stable model as well. It follows that M is a $\mathcal{PS}$-deterministic model.

We point out that, as shown in the next two examples, the intersection of all M-stable models need not be a P-stable model or, even, a model at all.

EXAMPLE 39 Consider the following program:

$$
\begin{array}{lcl}
\mathtt{a} & \leftarrow & \neg\mathtt{b}. \\
\mathtt{b} & \leftarrow & \neg\mathtt{a}. \\
\mathtt{q}_1 & \leftarrow & \mathtt{a}. \\
\mathtt{q}_2 & \leftarrow & \mathtt{b}. \\
\mathtt{u} & \leftarrow & \neg\mathtt{q}_1, \neg\mathtt{q}_2.
\end{array}
$$

There are two M-stable models: $M_1 = \{b, \neg a, q_2, \neg q_1, \neg u\}$ and $M_2 = \{a, \neg b, q_1, \neg q_2, \neg u\}$ (note that both models are total). Their intersection $\{\neg u\}$ is not a partial model. The only $\mathcal{PS}$-deterministic model is the empty set.

EXAMPLE 40 Consider the following program:

$$
\begin{array}{lcl}
\mathtt{a}. & & \\
\mathtt{p} & \leftarrow & \neg\mathtt{q}, \mathtt{a}. \\
\mathtt{q} & \leftarrow & \neg\mathtt{p}, \mathtt{a}. \\
\mathtt{r} & \leftarrow & \mathtt{p}. \\
\mathtt{r} & \leftarrow & \mathtt{q}.
\end{array}
$$

Here there are two M-stable models $\{a, p, r, \neg q\}$ and $\{a, q, r, \neg p\}$. In this case, their intersection $\{a, r\}$ is a model but not P-stable. The only $\mathcal{PS}$-deterministic model is $\{a\}$. Thus, both a and r are deterministic implications but only a is founded.

We next prove that the family of $\mathcal{PS}$-deterministic models, denoted by $\mathcal{DS}$, has an additional property: there exists a maximum element in the family which, therefore, can resolve all differences among $\mathcal{PS}$-deterministic models.

THEOREM 41 For every logic program $\mathcal{LP}$, $\langle \mathcal{DS}, \subseteq \rangle$ is a complete lattice.

PROOF. Obviously $\langle \mathcal{DS}, \subseteq \rangle$ is a partial order; to show that it is a complete lattice, we have to prove that, given any non-empty (possibly infinite) subset $\mathcal{X}$ of $\mathcal{DS}$, there exists a greatest lower bound and a lowest upper bound for $\mathcal{X}$, denoted $glb(\mathcal{X})$ and $lub(\mathcal{X})$, respectively. Let us first prove that $glb(\mathcal{X})$ exists. By Theorem 28, a greatest lower bound, say L, for $\mathcal{X}$ exists in the semilattice $\langle \mathcal{PS}, \subseteq \rangle$. But, as all models in $\mathcal{X}$ are $\mathcal{PS}$-deterministic, also L is $\mathcal{PS}$-deterministic and, then, $L = glb(\mathcal{X})$. Let us now prove that also $lub(\mathcal{X})$ exists. Let M be the union of all models in $\mathcal{X}$. As all models in $\mathcal{X}$ are P-stable and P-stable models are pre-stable, M is a pre-founded interpretation by Lemma 21; hence, by Part (2) of Lemma 22, $N = \mathbf{W}^{\infty}_{\mathcal{LP}}(M)$ is a P-stable model and by Part (1) of the same lemma, for each $K \in \mathcal{X}$, $K \subseteq N$. Moreover, since M is the union of $\mathcal{PS}$-deterministic models, also M is $\mathcal{PS}$-deterministic. Hence, given any M-stable model S of $\mathcal{LP}$, $M \subseteq S$ by Theorem 38. Therefore, since $\mathbf{W}_{\mathcal{LP}}$ is monotonic, $N = \mathbf{W}^{\infty}_{\mathcal{LP}}(M) \subseteq \mathbf{W}_{\mathcal{LP}}(S)$. But $S = \mathbf{W}_{\mathcal{LP}}(S)$ by Proposition 14; so $N \subseteq S$. Since we have taken an arbitrary M-stable model S, N is a $\mathcal{PS}$-deterministic model by Theorem 38, i.e., M is in $\mathcal{DS}$ and is an upper bound of $\mathcal{X}$. Consider now any Q in $\mathcal{DS}$ that is an upper bound for $\mathcal{X}$, i.e., $M \subseteq Q$. Since $\mathbf{W}_{\mathcal{LP}}$ is monotonic, $N = \mathbf{W}^{\infty}_{\mathcal{LP}}(M) \subseteq \mathbf{W}_{\mathcal{LP}}(Q)$. But again $Q = \mathbf{W}_{\mathcal{LP}}(Q)$ by Proposition 14; so $M \subseteq Q$. It follows that $N = lub(\mathcal{X})$ and this concludes the proof.

The top of the lattice of $\mathcal{PS}$-deterministic models for a program $\mathcal{LP}$ will be called the *maximum deterministic model* for $\mathcal{LP}$. We note that this model has been shown in [21] to be the greatest lower bound of the family of all M-stable models.

The significance of the maximum $\mathcal{PS}$-deterministic model can be appreciated through the following example based on Raymond Smullyan's puzzles [37].

EXAMPLE 42 We are in an island inhabited by knights and knaves. The knaves always lie, while the knights always tell the truth. Coming upon a pair of individuals, a and b, inspector Craig questions them about their nature. Then individual a answers: "Only one of us is a knight". What can inspector Craig infer about individual b?

Individual a is either a knight or a knave — same for b.

$$
\begin{array}{lcl}
\texttt{nit(a)} & \leftarrow & \neg\texttt{nav(a).} \\
\texttt{nav(a)} & \leftarrow & \neg\texttt{nit(a).} \\
\texttt{nit(b)} & \leftarrow & \neg\texttt{nav(b).} \\
\texttt{nav(b)} & \leftarrow & \neg\texttt{nit(b).}
\end{array}
$$

If a is knight, ergo tells the truth, then b must be knave:

$$\texttt{nav(b)} \quad \leftarrow \quad \texttt{nit(a).}$$

But if a is a knave, and, therefore a liar, b cannot be a knight (otherwise b would not have been mendacious): thus if a is a knave b must be a knave as well:

$$\texttt{nav(b)} \quad \leftarrow \quad \texttt{nav(a).}$$

For the above program, there are two M-stable models: $\{nit(a),\ nav(b),\ \neg nav(a),\ \neg nit(b)\}$ an $\{nav(a),\ nave(b),\ \neg nit(a),\ \neg nit(b)\}$. The maximum $\mathcal{PS}$-deterministic model instead is: $\{nav(b),\ \neg nit(b)\}$, while the well-founded model is empty. Thus the maximum $\mathcal{PS}$-deterministic model is capable of drawing all deterministic implications which follow from the given rules — in this case, for instance, drawing the conclusion that individual b is a knave, no matter what individual a is.

Note that the maximum $\mathcal{PS}$-deterministic model is M-stable if and only if $|\mathcal{MS}| = 1$.

Example 43 Take the following logic program:

$$
\begin{array}{lcl}
\mathsf{a} & \leftarrow & \neg\mathsf{b}. \\
\mathsf{b} & \leftarrow & \neg\mathsf{a}. \\
\mathsf{e} & \leftarrow & \neg\mathsf{a}, \neg\mathsf{b}. \\
\mathsf{a} & \leftarrow & \neg\mathsf{e}.
\end{array}
$$

Here there exists only one M-stable model $\{a, \neg b, \neg e\}$, which is then the maximum $\mathcal{PS}$-deterministic model. The well-founded model is the empty set for it is unable to realize that the program is deterministic as the third rule is always false.

Observe that the $\mathcal{PS}$-deterministic models trade off undefinedness for computability. At one extreme, the bottom of the lattice, we find the well-founded model, which ensures better computability at the expense of more undefinedness. At the other extreme, the top of the lattice, we find the maximum deterministic model, whose clear semantic advantages are counterbalanced by computational drawbacks that are discussed in the next section. It is interesting to observe that, while well-founded models capture the certainty semantics of $\mathcal{PS}$-deterministic models (i.e., the well founded model consists of all ground literals that are in every $\mathcal{PS}$-deterministic model), maximum $\mathcal{PS}$-deterministic models capture the possibility semantics (i.e., the maximum $\mathcal{PS}$-deterministic model consists of all ground literals that are in some $\mathcal{PS}$-deterministic model).

7 Data Complexity of Stable Models

To discuss the computational complexity of stable models, we restrict our attention `DATALOG`$^\neg$, i.e., logic programs with negative literals in the rule bodies but without functions symbols [4, 7, 19, 39]. We assume that the reader is familiar with the basic notions of computational complexity (see for instance [13, 17, 27]).

In a `DATALOG`$^\neg$ program $\mathcal{LP}$, it is customary to distinguish between (i) *EDB predicates*, i.e., extensional predicates which are defined by a (large) number of facts stored in a database and do not occur in the rule heads, and (ii) the *IDB predicates*, i.e., intensional predicates defined by rules. EDB predicate symbols form a relational database scheme $\mathcal{DB}_{\mathcal{LP}}$; thus, they are also seen as relation symbols. A database D on $\mathcal{DB}_{\mathcal{LP}}$ is a set of finite relations $D(r)$ on a countable domain U, one relation for each r in $\mathcal{DB}_{\mathcal{LP}}$. Given a database D on $\mathcal{DB}_{\mathcal{LP}}$, with a little abuse of notation we also see D as the set of facts: $\{r(t).| \ r \in \mathcal{DB}_{\mathcal{LP}} \wedge t \in D(r)\}$. Let $\mathcal{LP}_D$ denotes the logic program $\mathcal{LP} \cup D$: the Herbrand universe of $\mathcal{LP}_D$ is a finite subset of U as all possible constants in $\mathcal{LP}$ are taken from the domain U.

Using the *data complexity* approach of [8, 42], we view the (constant-size) `DATALOG`$^\neg$ program $\mathcal{LP}$ as defining a query function where the database is the input variable; therefore, the computational complexity of recognizing or finding a stable model will be expressed as a function of the size of the input database. The following results are known in the literature:

FACT 44 Given a `DATALOG`$^\neg$ program $\mathcal{LP}$, a database D on $\mathcal{DB}_{\mathcal{LP}}$, and an interpretation M for $\mathcal{LP}_D$,

1. there exist polynomial-time algorithms to decide whether (i) M is the well-founded model for $\mathcal{LP}_D$, (ii) M is a P-stable model for $\mathcal{LP}_D$, and (iii) M is a T-stable model for $\mathcal{LP}_D$;

2. deciding whether M is a M-stable or L-stable model for $\mathcal{LP}_D$ is *co*$\mathcal{NP}$-complete;

3. deciding whether a T-stable model for $\mathcal{LP}_D$ exists is $\mathcal{NP}$-complete.

PROOF. The polynomial-time recognition of T-stable models immediately follows from the definition of foundness [14]. The polynomial-time recognitions of well-founded models and of P-stable models have been shown in [41] and in [33], respectively. The proofs of Part (2) are given in [33] and Part (3) was first proven in [23].

EXAMPLE 45 A $\mathtt{DATALOG}^{\neg}$ program to find Hamiltonian circuits in a graph

$$
\begin{array}{lll}
r_1: \mathtt{selected\_edge(X,Y)} & \leftarrow & \mathtt{edge(X,Y)}, \neg\mathtt{diff\_selection(X,Y)}.\\
r_2: \mathtt{diff\_selection(X,Y)} & \leftarrow & \mathtt{selected\_edge(X,Y')}, \mathtt{Y} \neq \mathtt{Y'}.\\
r_3: \mathtt{joined(X,Y)} & \leftarrow & \mathtt{selected\_edge(X,Y)}.\\
r_4: \mathtt{joined(X,Y)} & \leftarrow & \mathtt{selected\_edge(X,Z)}, \mathtt{joined(Z,Y)}.\\
r_5: \mathtt{non\_hamiltonian} & \leftarrow & \mathtt{vertex(X)}, \mathtt{vertex(Y)}, \neg\mathtt{joined(X,Y)}.\\
r_6: \mathtt{diff\_selection(X,Y)} & \leftarrow & \neg\mathtt{non\_hamiltonian}, \mathtt{selected\_edge(X,Y)}.
\end{array}
$$

Consider now the program in Example 45, above, where $\mathcal{DB}_{\mathcal{LP}} = \{vertex, edge\}$ defines the vertices and the edges of a graph. Let a database D on $\mathcal{DB}_{\mathcal{LP}}$ be given—say that D defines a graph $G = \langle V, E \rangle$ with $n > 0$ nodes. Observe that, by means of rules r_1,r_2, any possible T-stable model M of $\mathcal{LP}_D$ selects a maximal subset of V such that no two edges have the same source node; thus M selects at most n edges. Obviously, the selected edges form an Hamiltonian circuit if and only if there is a path between any two nodes, thus if and only if *non_hamiltonian* is false—rule r_5. But, if *non_hamiltonian* is true, then the body of the rule r_6 is true; so all initial selections are invalidated and M looses foundness. It turns out that M preserves T-stability iff *non_hamiltonian* is false, i.e., the edges selected by M form a Hamiltonian circuit. Hence, there exists a T-stable model iff the graph has a Hamiltonian circuit. As the latter problem is well-known to be $\mathcal{NP}$-complete [13], deciding whether there exists a T-stable model for $\mathcal{LP}_D$ is $\mathcal{NP}$-complete as well. Observe also that if there exists no Hamiltonian circuit in the graph, then every P-stable model which makes any edge selection is eventually invalidated by rule r_5; so the only P-stable model is the well-founded model that makes no selection at all, thus the well-founded model is both M-stable and L-stable. Therefore the well-founded model of $\mathcal{LP}_D$ is M-stable (L-stable) iff there exists no Hamiltonian circuit in the graph; so deciding whether the well-founded model of $\mathcal{LP}_D$ is M-stable (L-stable) is $co\mathcal{NP}$-complete. This confirms that deciding whether an interpretation is M-stable or L-stable is $co\mathcal{NP}$-hard.

Next we show that, as for M-stable or L-stable models, the recognition of whether a given interpretation is a $\mathcal{PS}$-deterministic model of $\mathcal{LP}_D$ cannot be done in polynomial time, unless $\mathcal{P} = \mathcal{NP}$.

THEOREM 46 Given a $\mathtt{DATALOG}^{\neg}$ program $\mathcal{LP}$, a database D on $\mathcal{DB}_{\mathcal{LP}}$ and an interpretation M of $\mathcal{LP}$, deciding whether M is a $\mathcal{PS}$-deterministic model of $\mathcal{LP}_D$ is $co\mathcal{NP}$-complete.

PROOF. (*Membership*) Deciding whether M is a $\mathcal{PS}$-deterministic model of $\mathcal{LP}_D$ can be done in two steps: (1) testing whether M is P-stable and (2) verifying that for every P-stable model N of $\mathcal{LP}_D$, M and N are not mutually exclusive. The first step is in $\mathcal{P}$ by Part (1) of Fact 44. Consider the complementary problem of the step (2): does there exists a P-stable model N such that M and N are mutually exclusive? This problem is in $\mathcal{NP}$ since we can just guess an interpretation N and, then, test in polynomial time whether N is P-stable and M and N are mutually exclusive. Hence, Step 2 is in $co\mathcal{NP}$ and, then, the overall test of whether M is a $\mathcal{PS}$-deterministic model is in $co\mathcal{NP}$ as well.

(*Completeness*) Let $\mathcal{LP}'$ be the DATALOG$^\neg$ program obtained from the program in Example 45 by replacing the rule r_6 with the following one:

$$\mathtt{r_{6'}:}\ \mathtt{q} \ \leftarrow \ \mathtt{non\_hamiltonian}, \neg\mathtt{q}.$$

Obviously $\mathcal{DB}_{\mathcal{LP}'} = \mathcal{DB}_{\mathcal{LP}}$. Given a database D on $\mathcal{DB}_{\mathcal{LP}'}$, for each P-stable model M of $\mathcal{LP}'_D$, either $\neg q$ is in M or q is left undefined in $\overline{M}$; moreover, $\neg q$ is in some P-stable model iff *non_hamiltonian* is false, i.e., iff the graph has a Hamiltonian circuit. Consider now the DATALOG$^\neg$ program $\mathcal{LP}'' = \mathcal{LP}' \cup \{r_7, r_8\}$, where r_7 and r_8 are:

$$\begin{aligned}\mathtt{r_7:}\ \mathtt{q} &\leftarrow \neg\mathtt{p}.\\ \mathtt{r_8:}\ \mathtt{p} &\leftarrow \neg\mathtt{q}.\end{aligned}$$

Note that $\mathcal{DB}_{\mathcal{LP}'} = \mathcal{DB}_{\mathcal{LP}''}$. Consider the interpretation $M = \{\neg p, q\} \cup D \cup \neg\overline{D}$ for $\mathcal{LP}''_D$, where D is seen as a set of ground literals and $\overline{D}$ consists of all ground atoms $edge(x, y)$ that are not in D. It is easy to see that M is a P-stable model of $\mathcal{LP}''_D$ and is distinct from the well-founded model $M_w = D \cup \neg\overline{D}$. Consider now any M-stable model N of $\mathcal{LP}''_D$. Obviously, $D \cup \neg\overline{D} \subseteq N$. Hence, M and N are mutually exclusive if and only if $\{p, \neg q\} \subset N$. By Theorem 38, M is $\mathcal{PS}_{\mathcal{LP}''}$-deterministic iff there exists no M-stable model of $\mathcal{LP}''_D$ that contains $\neg q$, i.e., iff the graph has no Hamiltonian circuits. But deciding whether a graph has no Hamiltonian circuits is $co\mathcal{NP}$-complete; so also deciding whether M is $\mathcal{PS}_{\mathcal{LP}''}$-deterministic is $co\mathcal{NP}$-complete.

We now turn to the problem of recognizing whether a given interpretation is the maximum $\mathcal{PS}$-deterministic model. Observe that this problem can be formulated as a maximal problem for which the recognition of a feasible solution is in $co\mathcal{NP}$. We can then use the analysis of [10] to immediately derive the following upper bound to the problem: recognizing the maximum $\mathcal{PS}$-deterministic model is in Π^p_2. Next we show that this bound is not tight although we do not provide a precise characterization of the complexity.

THEOREM 47 Given a DATALOG$^\neg$ program $\mathcal{LP}$, a database D on $\mathcal{DB}_{\mathcal{LP}}$ and an interpretation M of $\mathcal{LP}_D$, deciding whether M is a the maximum $\mathcal{PS}$-deterministic model of $\mathcal{LP}_D$ is in Δ_2^p and is $\mathcal{D}^p$-hard.

PROOF. (*Membership*) By Theorem 46, the preliminary test of whether M is a $\mathcal{PS}$-deterministic model of $\mathcal{LP}_D$ can be done through a call to an $\mathcal{NP}$ oracle. Suppose that this test succeeds; so, we only have to decide whether M is maximum. Assume that $\overline{M} \neq \emptyset$ because otherwise M would be immediately recognized maximum. For each A in $\overline{M}$, we invoke an $\mathcal{NP}$ oracle to test whether there exist two P-stable models M_1 and M_2 such that $A \in M_1$ and $\neg A \in M_2$.[5] Let X be the set of $A \in M$ for which the test fails. We now call an $\mathcal{NP}$ oracle to test whether there exists a P-stable model M_1 such that $M \subset M_1$ and for each $B \in M_1 - M$, either B or $\neg B$ is in X. If this test succeeds, then M_1 is $\mathcal{PS}$-deterministic by the construction of X; then, M is not the maximum $\mathcal{PS}$-deterministic model since $M \subset M_1$. Otherwise, every P-stable model containing M includes a literal B such that neither B nor $\neg B$ is in X and, therefore, B is contradicted by some other P-stable models; thus M is the maximum $\mathcal{PS}$-deterministic model. Hence, the overall recognition can be done in polynomial time using a linear number of calls to an $\mathcal{NP}$ oracle; thus, the recognition is in Δ_2^p.

(*Hardness*) By Theorem 46, recognizing whether M is the maximum $\mathcal{PS}$-deterministic is *co*$\mathcal{NP}$-hard. We next show that is also $\mathcal{NP}$-hard. To this end, consider the DATALOG$^\neg$ program $\mathcal{LP}''$ in the proof of Theorem 46. Given a database D on $\mathcal{DB}_{\mathcal{LP}}$ (i.e., a graph), it is easy to see that the well-founded model of $\mathcal{LP}''_D$ is the maximum $\mathcal{PS}$-deterministic model if and only if the graph has a Hamiltonian circuit. So deciding whether M is the maximum $\mathcal{PS}$-deterministic model is also $\mathcal{NP}$-hard. It is now simple to show that the recognition is $\mathcal{D}^p$-hard. One only needs to take a known $\mathcal{D}^p$-hard problem, representing the two subproblems in $\mathcal{NP}$ and in *co*$\mathcal{NP}$ with two programs as for the Hamiltonian circuit problem (see [32] for some samples of such representations), combining them into a unique program and, then, performing the maximum $\mathcal{PS}$-deterministic test.

Our next order of business is to investigate the complexity of the *search* problem for stable models. It is well known that the well-founded model can be found in polynomial time [41]. On the other hand, the other types of stable models cannot be found in polynomial time unless $\mathcal{P} = \mathcal{NP}$ because of the hardness of their recognition or, for the case of T-stable models, of their existential test. Actually, one might wonder whether the search complexity

[5]Note that this oracle is indeed an $\mathcal{NP}$ oracle since, by Fact 44, testing whether an interpretation is a P-stable model is in $\mathcal{P}$ — the same consideration holds for all the oracles used in the proofs.

blows up beyond $\mathcal{FP}^{\mathcal{NP}}$, as it is frequently the case of search problems whose decision test is $co\mathcal{NP}$-hard—we recall that $\mathcal{FP}^{\mathcal{NP}}$ is the class of search problems that can be solved in polynomial time by a machine with access to an $\mathcal{NP}$ oracle. However:

THEOREM 48 Given a $\texttt{DATALOG}^{\neg}$ program $\mathcal{LP}$ and a database D on $\mathcal{DB}_{\mathcal{LP}}$, finding a T-stable, M-stable or L-stable model or the maximum $\mathcal{PS}$-deterministic model for $\mathcal{LP}_D$ is in $\mathcal{FP}^{\mathcal{NP}}$.

PROOF. Since testing whether an interpretation is a T-stable model is in $\mathcal{P}$, finding a T-stable model is clearly in $\mathcal{FP}^{\mathcal{NP}}$. Nevertheless, for, we next show how a T-stable model can be effectively found. We first perform the existence check by invoking an $\mathcal{NP}$ oracle; then, if the test succeeds, we compute the well founded model M_w, set $M = M_w$ and repeat the following steps until after $\overline{M}$ becomes empty: (a) take any A in $\overline{M}$, (b) call an $\mathcal{NP}$ oracle to test whether there exists a T-stable model M_1 such that $M_1 \supseteq (M \cup \{A\})$, and (c) add A to M if the test succeeds or $\neg A$ otherwise. It is easy to see that the resulting M is a T-stable model. Observe that the NP oracle is called $k+1$ times, where k is the number of elements of the Herbrand base that are left undefined in the well-founded model. Note that k is at most equal to the cardinality of the Herbrand Base.

To find an M-stable model, we compute the well founded model M_w, set $M = M_w$ and $K = \overline{M}$ and repeat the following steps until after K becomes empty: (a) take any A in K and remove it from K, (b) call an $\mathcal{NP}$ oracle to test whether there exists a P-stable model M_1 such that $M_1 \supseteq (M \cup \{A\})$, and (c) add A to M if the test succeeds or otherwise perform the two additional steps: (c_1) call an $\mathcal{NP}$ oracle to test whether there exists a P-stable model M_1 such that $M_1 \supseteq (M \cup \{\neg A\})$ and (c_2) add $\neg A$ to M if the test succeeds. It is easy to see that the resulting M is an M-stable model. Observe also that the $\mathcal{NP}$ oracle is called $2 \times k$ times, where k is defined as above.

To find an L-stable model, we compute the well founded model M_w and proceed in two phases. In the first phase we single out a subset H of the Herbrand Base such that there exists an L-stable model M for which $\overline{M} = H$ and in the second phase we actually determine one such model. The first phase is implemented as follows. We set $L = \overline{M_w}$ and $H = \emptyset$ and repeat the following steps until L becomes empty: (a) take any A in L and remove it from L, (b) call an $\mathcal{NP}$ oracle to test whether there exists a P-stable model M_1 such that $\overline{M_1} \supseteq H$ and either A or $\neg A$ is in M_1, and (c) add A to H if the test does not succeed. It is easy to see that the resulting H has the wanted properties. We then pass to the second phase. We set $M = M_w$ and $K = \overline{M} - H$ and repeat the following steps until after K becomes empty: (a) take any A in K and remove it from K, (b) call an $\mathcal{NP}$ oracle to test whether there exists a P-stable model M_1 such that $M_1 \supseteq (M \cup \{A\})$ and $\overline{M_1} = H$, and (c) add A to M if the test succeeds or otherwise add $\neg A$ to M. It is easy to see that the resulting M is an M-stable model for which $\overline{M} = H$ and is, therefore, L-stable by the way H has been constructed. Observe that the $\mathcal{NP}$ oracle is called at most $2 \times k$ times as in the search of an M-stable model.

Finally, to find the maximum $\mathcal{PS}$-deterministic model M_m, we also proceed in two phases after computing M_w. In the first phase, we determine the set X of elements A in $\overline{M_w}$ such that there exist no two P-stable models M_1 and M_2 for which $A \in M_1$ and $\neg A \in M_2$ (we proceed as in the proof of Theorem 47). In the second phase, we actually determine M_m, as follows. We set $M = M_w$ and $K = X$ and repeat the same steps as in the search for an M-stable model until after K becomes empty. It is easy to see that the resulting M is the minimum $\mathcal{PS}$-deterministic model. Observe that an $\mathcal{NP}$ oracle is called at most $3 \times k$ times, where k is defined as above.

Since finding an M-stable model corresponds to finding the maximal solution of a problem for which a feasible solution (i.e., a P-stable model) can be recognized in polynomial time, we can use the results of [10] to determine tighter upper bounds in the computation.

The results of this section confirm that computational complexity of stable models blows up as soon as we move beyond well-founded model. This does not mean that stable model semantics past the well-founded model cannot provide a solid basis for practical computations. In fact polynomial computations of P-stable models can be achieved for several interesting classes of programs [26]. Furthermore, as shown in [16], it is possible to have have an exponential behavior restricted to problems that are $\mathcal{NP}$-hard and ensure a polynomial behavior for the simpler problems. Unfortunately, a discussion of these interesting topics will take us beyond the scope of this paper.

8 Conclusion

This paper has proposed a reasoned reconstruction of the model-theoretic foundations of stable models semantics for logic programs with negation in the domain of partial interpretations (*P-stable models*), clarifying their underlying semantic properties and relationships to non-determinism. Our analysis has shown that the class of P-stable models for a given program is very broad, and contains elements having assorted semantics and characteristics. Therefore, we identified various subclasses of great semantic significance, beginning with the class of deterministic models and that of non-deterministic models.

The class of deterministic P-stable models for a program $\mathcal{LP}$ forms a complete lattice. The bottom of this lattice coincides with the well-founded model for $\mathcal{LP}$, which has been studied by several authors. The top element of the lattice is the maximum deterministic P-stable model. Albeit overlooked by previous authors, this has great significance from a semantic viewpoint, inasmuch as it minimizes undefinedness without surrendering uniqueness.

Reducing undefinedness beyond the maximum deterministic P-stable model is only possible by accepting a semantics based on a non-deterministic choice among multiple alternate models, which, because of unreconcilable differences, are pairwise mutually exclusive. Non-determinism characterized the original definition of total stable (*T-stable*) models proposed in [14], which, however, lacks universality. Thus, we considered a generalized notion of stable models by replacing the requirement of totality with that of maximality in the family of P-stable models (*M-stable models*) and showed that this defines the largest subfamily of strictly non-deterministic models.

Our investigation uncovered interesting semantic aspects of non-determinism ingrained in M-stable models; e.g., P-stable models form a lower semilattice whose top elements are all the M-stable models, and the family of P-stable models is not deterministic if and only if the number of M-stable models is greater than 1. Thus, the potential non-determinism of P-stable models is fully captured by the family of M-stable models, or by any of its subfamilies that are singled out through stricter conditions of minimal undefinedness. One of such subfamilies is obviously represented by T-stable models. We have then defined another interesting subfamily of M-stable models, composed by the models with minimal set of undefined atoms (*L-stable models*), and we have proven that it differs from the subfamily of T-stable model only when the latter is empty, i.e., L-stable models appear to be the most appropriate extension of the notion of T-stability. The relevance of L-stable models has been confirmed in [33] who has proved that L-stable models have an expressive power higher than any other type of stable model.

Finally, we investigated the complexity of finding an M-stable model, an L-stable model or the maximum $\mathcal{PS}$-deterministic model of a program on a finite universe. We proved these problems can be solved in polynomial time by a machine with access to an $\mathcal{NP}$ oracle.

Acknowledgements

The work was partially supported by a European Union grant under the EU-US project, "DEUS EX MACHINA: non-determinism for deductive databases" (NSF: IRI-9314905). The first author was also supported by a MURST grant (40% share) under the project "Sistemi formali e strumenti per basi di dati evolute". The authors would like to thank Luigi Palopoli for several useful discussions, Vladimir Lifschitz and Francesca Rossi for their comments on an earlier version of this paper, and Allen Van Gelder for having suggested

Example 13.

References

[1] K. Apt, H. Blair and A. Walker. Towards a theory of declarative knowledge. In *Foundations of Deductive Databases and Logic Programming*, J. Minker (ed.), Morgan Kauffman, Los Altos, USA, 1988, 89-142.

[2] S. Abiteboul, E. Simon and V. Vianu. Non-deterministic languages to express deterministic transformations. In *Proc. ACM PODS Symp.*, 1990, pp. 218-229.

[3] S. Abiteboul and V. Vianu. Non-determinism in logic-based languages. *Annals of Mathematics and Artificial Intelligence 3*, 1991, pp. 151-186.

[4] S. Abiteboul, R. Hull and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1994.

[5] V. Baral and V. Subrahmanian. Stable and extension class theory for logic programs and default logic. *Journal of Automated Reasoning*, 1992, pp. 345-366.

[6] N. Bidoit. Negation in Rule-based Database Languages: a Survey. *Theoretical Computer Science 78*, 3, 1991, pp. 3-83.

[7] S. Ceri, G. Gottlob and L. Tanca. *Logic Programming and Databases*. Springer-Verlag, Berlin, Germany, 1990.

[8] A. Chandra and D. Harel. Structure and Complexity of Relational Queries. *Journal of Computer and System Sciences 25*, 1, 1982, pp. 99-128.

[9] A. Chandra and D. Harel. Horn clause and Generalizations. *Journal of Logic Programming 2*, 1, 1985, pp. 1-15.

[10] Z-Z. Chen and S. Toda. The Complexity of Selecting Maximal Solutions. In *Proc of 8th IEEE Conf. on "Structure in Complexity Theory"*, 1993, pp. 313-325.

[11] D. Chimenti et al. The LDL System Prototype. *IEEE Journal on Data and Knowledge Engineering*, March 1990.

[12] P. Dung. Negation as Hypotheses: an abductive foundation for logic programming. In *Proc. 8th Conf. on Logic Programming*, MIT Press, 1991, pp. 3-17.

[13] M. Garey and D.S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness.* W.H. Freeman, New York, USA, 1979.

[14] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proc. 5th Int. Conf. and Symp. on Logic Programming*, MIT Press, Cambridge, 1988, pp. 1070-1080.

[15] F. Giannotti, D. Pedreschi, D. Saccà. and C. Zaniolo. Non-Determinism in Deductive Databases. In *Proc. 2nd Conference on Deductive and Object-Oriented Databases – DOOD 91, LNCS 566 (Eds: Delobel, C., Kifer M., Masunaga Y.),* Springer- Verlag, December 1991, pp. 129-146.

[16] S. Greco, D. Saccà and C. Zaniolo. DATALOG Queries with Stratified Negation and Choice: from $\mathcal{P}$ to $\mathcal{D}^p$. In *Proc. of the Int. Conf. on Database Theory, LNCS n. 893,* Springer-Verlag, 1995, pp.82-96.

[17] D.S. Johnson. A Catalog of Complexity Classes. In *Handbook of Theoretical Computer Science*, Vol. 1, J. van Leewen (ed.), North-Holland, 1990.

[18] A. Kakas and P. Mancarella. Preferred Extensions are Partial Stable Models. *Journal of Logic Programming*, 1992, pp. 341-348.

[19] P.C. Kanellakis. Elements of Relational Database Theory. In *Handbook of Theoretical Computer Science*, Vol. 2, J. van Leewen (ed.), North-Holland, 1991.

[20] R. Krishnamurthy and S.A. Naqvi. Non-Deterministic Choice in Datalog. In *Proc. 3rd Int. Conf. on Data and Knowledge Bases*, Morgan Kaufmann Pub., Los Altos, 1988, pp. 416-424.

[21] E. Laenens, D. Vermeir and C. Zaniolo. Logic Programming Semantics Made Easy. In *Proceedings of the ICALP'92 conference*, Vienna, April, 1992.

[22] J.W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, Berlin, Germany, 1987.

[23] W. Marek and M. Truszczynski. Autoepistemic Logic. *Journal of the ACM 38*, 3, 1991, pp. 588-619.

[24] S.A. Naqvi. A Logic for Negation in Database Systems. In *Foundations of Deductive Databases and Logic Programming,* (Minker, J., ed.), Morgan Kaufman, Los Altos, 1987.

[25] S. Naqvi and S. Tsur, *A Logical Data Language for Data and Knowledge Bases*, Computer Science Press, New York, 1989.

[26] L. Palopoli and C. Zaniolo. Polynomial-Time Computable Stable Models. *Annals of Mathematics and Artificial Intelligence*, Vol. 17, No. 3/4, pp. 261-290, December 1996.

[27] C. Papadimitriou. *Computational Complexity.* Addison-Wesley, Reading, MA, USA, 1994.

[28] T.C. Przymusinski. On the Declarative and Procedural Semantics of Deductive Databases and Logic Programs. In *Foundations of Deductive Databases and Logic Programming,* (Minker, J. ed.), Morgan Kaufman, Los Altos, 1987, pp. 193-216.

[29] T.C. Przymusinski. Well-founded Semantics Coincides with Three-valued Stable Semantics. *Foundamenta Informaticae 13*, 1990, pp. 445-463.

[30] R. Ramakrisnhan, D. Srivastava and S. Sudanshan, "CORAL — Control, Relations and Logic", in *Proceedings of 18th VLDB Conference*, 1992.

[31] K. Ross. A Procedural Semantics for Well-Founded Negation in Logic Programs. In *Proc. ACM Symp. on Principles of Database Systems*, pp. 22-31, 1989.

[32] D. Saccà. Multiple Total Stable Models are Definitely Needed to Solve Unique Solution Programs. *Information Processing Letters 58*, 5, 1996, pp. 249-254.

[33] D. Saccà. The Expressive Powers of Stable Models for Bound and Unbound Queries. *Journal of Computer and System Sciences*, to appear.

[34] D. Saccà and C. Zaniolo. Stable Models and Non-Determinism in Logic Programs with Negation. In *Proc. ACM PODS Symp.*, 1990, pp. 205-218.

[35] D. Saccà and C. Zaniolo. Partial Models and Three-Valued Models in Logic Programs with Negation. In *Proc. 1st Int. Conf. on Logic Programs and Non-monotonic Reasoning*, MIT Press, 1991, pp. 87-104.

[36] J.S. Schlipf. The Expressive Powers of the Logic Programming Semantics. *Journal of Computer and System Science 51,* 1995, pp. 64-86

[37] R.M. Smullyan. *What is the Name of this Book?: the Riddle of Dracula and Other Logical Puzzles.* Prentice Hall, 1978.

[38] A. Tarski. A Lattice Theoretical Fixpoint Theorem and its Application. *Pacific Journal of Mathematics 5*, pp. 285-309, 1955.

[39] J.D. Ullman. *Principles of Database and Knowledge Base Systems.* Vol 1-2, Computer Science Press, 1989.

[40] A. Van Gelder. Negation as Failure using tight derivations for general logic programs. *Journal of Logic Programming 6*, 1, 1989, pp. 109–133.

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

[42] M.Y. Vardi. The Complexity of Relational Query Languages. In *Proc. ACM Symp. on Theory of Computing*, 1982, pp. 137-146.

[43] J. You and L.Y. Yuan. On the Equivalence of Semantics for Normal Logic Programs. *Journal of Logic Programming*, 1995, pp. 211-222.

Appendix

A Proofs of Lemmata and Facts

Before proving the first result, we need to state the following claim — other claims will be later introduced as needed.

CLAIM 49 Let $\mathcal{LP}$ be a logic program and I be an interpretation for $\mathcal{LP}$ such that $\neg I^-$ is an unfounded set w.r.t. I. Then $\mathbf{T}_{\mathcal{LP}}(I) \cap \neg I^- = \emptyset$.

PROOF. We carry out the proof by contradiction and, therefore, we assume that $\mathbf{T}_{\mathcal{LP}}(I) \cap \neg I^- \neq \emptyset$. Let A be any element in $\mathbf{T}_{\mathcal{LP}}(I) \cap \neg I^-$; then there exists a ground rule of $\mathcal{LP}$, say r, such that $H(r) = A$ and $G(r) \subseteq I$. Hence, $G(r) \cap (\neg I \cup \neg I^-) = \emptyset$ and, then, $\neg I^-$ is not an unfounded set w.r.t. I — a contradiction. It follows that $\mathbf{T}_{\mathcal{LP}}(I) \cap \neg I^- = \emptyset$.

FACT 2
A total interpretation I of a logic program $\mathcal{LP}$ is a total model of $\mathcal{LP}$ if and only if $\neg I^-$ is an unfounded set w.r.t. I.

PROOF. Assume first that $\neg I^-$ is an unfounded set w.r.t. I. Then $\mathbf{T}_{\mathcal{LP}}(I) \cap \neg I^- = \emptyset$ by Claim 49. Hence, as I is total, $\mathbf{T}_{\mathcal{LP}}(I) \subseteq I^+$, thus I is a total model. Assume now that I is a total model and let A be any element in $\neg I^-$. Take any rule $r \in ground(\mathcal{LP})$ with $H(r) = A$. Since I is total, $G(r) \cap \neg I \neq \emptyset$ because otherwise $H(r)$ would be in $\mathbf{T}_{\mathcal{LP}}(I)$. It follows that $\neg I^-$ is an unfounded set w.r.t. I.

CLAIM 50 Let $\mathcal{LP}$ be a logic program and I be an interpretation for $\mathcal{LP}$. Then $\mathbf{T}^{\infty}_{pos(\mathcal{LP},I)}(\emptyset) \subseteq \mathbf{T}_{\mathcal{LP}}(I) \cap (I^+ \cup \neg I^-)$.

PROOF. Let $X = T^{\infty}_{pos(\mathcal{LP},I)}(\emptyset)$. By definition of fixpoint, $X = \mathbf{T}_{pos(\mathcal{LP},I)}(X)$. Then, for each A in X, there exists a rule r in $pos(\mathcal{LP}, I)$ such that $H(r) = A$ and $G(r) \subseteq X$. Let r' be the original rule in $ground(\mathcal{LP})$ from which r has been constructed. By definition of $pos(\mathcal{LP}, I)$, $G(r') = G(r) \wedge \neg A_1, \ldots, \neg A_n$, $G(r) \subseteq I^+$ and each $\neg A_i$ is in I^-. Hence $G(r') \subseteq I$ and, then, $A = H(r')$ is in $\mathbf{T}_{\mathcal{LP}}(I)$. It follows that $X \subseteq \mathbf{T}_{\mathcal{LP}}(I)$. The fact that also $X \subseteq I^+ \cup \neg I^-$ follows from the definition of $pos(\mathcal{LP}, I)$.

FACT 3
Let $\mathcal{LP}$ be a logic program and I be an interpretation for $\mathcal{LP}$. Then (i) if I is pre-founded then $I^+ \subseteq \mathbf{T}_{\mathcal{LP}}(I)$ and (ii) if I is founded then $I^+ = \mathbf{T}_{\mathcal{LP}}(I)$.

PROOF. Suppose that I is pre-founded. Then, by definition of pre-foundness, $I^+ = T^{\infty}_{pos(\mathcal{LP},I)}(\emptyset)$; then, by Claim 50, $I^+ \subseteq \mathbf{T}_{\mathcal{LP}}(I)$. Suppose now that I is also founded, i.e., also $I^+ \supseteq \mathbf{T}_{\mathcal{LP}}(I)$. It follows that $I^+ = \mathbf{T}_{\mathcal{LP}}(I)$.

FACT 4
Let $\mathcal{LP}$ be a logic program and I be a total pre-founded interpretation for $\mathcal{LP}$. Then I is founded.

PROOF. We proceed by contradiction and we assume that I is not founded, i.e., $I^+ \neq \mathbf{T}_{\mathcal{LP}}(I)$. Then, by Fact 3, $I^+ \subset \mathbf{T}_{\mathcal{LP}}(I)$. Let A be any element in $\mathbf{T}_{\mathcal{LP}}(I) - I^+$ and r be a rule in $ground(\mathcal{LP})$ such that $A = H(r)$ and $G(r) \subseteq I$; such a rule exists because $A \in \mathbf{T}_{\mathcal{LP}}(I)$. As I is total, r is not removed in the construction of $pos(\mathcal{LP}, I)$ but is modified into a rule r' with $H(r') = A$ and $G(r') = G(r) \cap I^+$. Then $A \in \mathbf{T}_{pos(\mathcal{LP},I)}(I^+)$ — a contradiction with the hypothesis that I is pre-founded, i.e., $I^+ = T_{pos(\mathcal{LP},I)}(I^+)$. It follows that I is founded.

CLAIM 51 Let $\mathcal{LP}$ be a logic program and I be a pre-founded interpretation for $\mathcal{LP}$. Then $I^+ \cap \mathbf{U}_{\mathcal{LP}}(I) = \emptyset$ and $\mathbf{T}_{\mathcal{LP}}(I) \cap \mathbf{U}_{\mathcal{LP}}(I) = \emptyset$.

PROOF. Let us first prove that $I^+ \cap \mathbf{U}_{\mathcal{LP}}(I) = \emptyset$. We proceed by contradiction and we assume, then, that $X = I^+ \cap \mathbf{U}_{\mathcal{LP}}(I) \neq \emptyset$ and let $Y = I^+ - X$. Since $I^+ = \mathbf{T}^{\infty}_{pos(\mathcal{LP},I)}(\emptyset)$ by definition of pre-foundness, $X \subseteq \mathbf{T}^{\infty}_{pos(\mathcal{LP},I)}(\emptyset)$. Let i be the last natural such that $\mathbf{T}^{i}_{pos(\mathcal{LP},I)}(\emptyset)$ contains no elements of X. Since $\mathbf{T}_{pos(\mathcal{LP},I)}$ is monotone and continuous in a complete lattice, such an i exists. Then $\mathbf{T}^{i}_{pos(\mathcal{LP},I)}(\emptyset) \subseteq Y$ and $\mathbf{T}^{i+1}_{pos(\mathcal{LP},I)}(\emptyset) \cap X \neq \emptyset$. As $\mathbf{T}^{i+1}_{pos(\mathcal{LP},I)}(\emptyset) = \mathbf{T}_{pos(\mathcal{LP},I)}(\mathbf{T}^{i}_{pos(\mathcal{LP},I)}(\emptyset))$, there exists $A \in X$ and a rule r in $pos(\mathcal{LP}, I)$ such that $H(r) = A$ and $G(r) \subseteq Y$. Let r' be the original rule in $ground(\mathcal{LP})$ from which r has been constructed. By definition of $pos(\mathcal{LP}, I)$, $G(r') = G(r) \wedge \neg A_1, \ldots, \neg A_n$ and each $\neg A_i$ is in I^-; so $G(r') \subseteq Y \cup I^-$ and $A =$

$H(r')$. Hence, neither $G(r') \cap \neg I = \emptyset$ (because all literals in $G(r')$ are in I) nor $G(r') \cap \mathbf{U}_{\mathcal{LP}}(I) = \emptyset$ (because all positive literals in $G(r')$ are in Y). Therefore, as A is in $X \subseteq \mathbf{U}_{\mathcal{LP}}(I)$, $\mathbf{U}_{\mathcal{LP}}(I)$ is not an unfounded set w.r.t. I — a contradiction. It follows that $I^+ \cap \mathbf{U}_{\mathcal{LP}}(I) = \emptyset$. To complete the proof, we have to show that $(\mathbf{T}_{\mathcal{LP}}(I) - I^+) \cap \mathbf{U}_{\mathcal{LP}}(I) = \emptyset$. Let A be any element in $\mathbf{T}_{\mathcal{LP}}(I) - I^+$. Since A is in $\mathbf{T}_{\mathcal{LP}}(I)$, there exists a rule r in $ground(\mathcal{LP})$ such that $H(r) = A$ and $G(r) \subseteq I$. Hence $G(r) \cap \neg I = \emptyset$. Moreover, as $I^+ \cap \mathbf{U}_{\mathcal{LP}}(I) \neq \emptyset$ and every positive goal of $G(r)$ is in I^+, $G(r) \cap \mathbf{U}_{\mathcal{LP}}(I) = \emptyset$; therefore, A is not in $\mathbf{U}_{\mathcal{LP}}(I)$. We can then conclude that $\mathbf{T}_{\mathcal{LP}}(I) \cap \mathbf{U}_{\mathcal{LP}}(I) = \emptyset$.

FACT 6
Let $\mathcal{LP}$ be a logic program and I be a total interpretation for $\mathcal{LP}$. Then I is a T-stable model if and only if $\neg I^- = \mathbf{U}_{\mathcal{LP}}(I)$.

PROOF. Let us first assume that I is a T-stable model. Then M is founded and $I^+ \cap \mathbf{U}_{\mathcal{LP}}(I) = \emptyset$ by Claim 51; so, as I is total, $\neg I^- = \mathbf{U}_{\mathcal{LP}}(I)$. Let us now assume that $\neg I^- = \mathbf{U}_{\mathcal{LP}}(I)$. By Fact 2, I is a total model, i.e., $\mathbf{T}_{\mathcal{LP}}(I) \subseteq I^+$. By Claim 50, $X = \mathbf{T}^{\infty}_{pos'(\mathcal{LP},I)}(\emptyset) \subseteq \mathbf{T}_{\mathcal{LP}}(I)$; so $X \subseteq I^+$. We prove by contradiction that $I^+ = X$. Suppose then that $Y = I^+ - X \neq \emptyset$. Take any A in Y and any rule r in $ground(\mathcal{LP})$ for which $H(r) = A$ and $G(r) \subseteq I$; such a rule exists since $A \in \mathbf{T}_{\mathcal{LP}}(I)$. We have that $G(r) \cap Y \neq \emptyset$ because otherwise from the corresponding rule r' in $pos(\mathcal{LP}, I)$ we would derive A in $\mathbf{T}^{\infty}_{pos(\mathcal{LP},I)}(\emptyset)$. Hence, Y is an unfounded set w.r.t. I — then, as $Y \subseteq I^+$, we get a contradiction with the hypothesis that $\neg I^- = \mathbf{U}_{\mathcal{LP}}(I)$. It follows that $Y = I^+ - X = \emptyset$ and, therefore, I is pre-founded and, then, founded by Fact 4.

FACT 9
Every partial model of a program $\mathcal{LP}$ is a subset of some total model.

PROOF. Let M be a partial model of $\mathcal{LP}$; we show that $N = M \cup \overline{M}$ is a total model. By construction, N is consistent and every element of $B_{\mathcal{LP}}$ occurs in N; therefore N is a total interpretation of $\mathcal{LP}$. By definition, $\neg M^-$ is an unfounded set w.r.t. M; therefore, as $M \subseteq N$, $\neg M^-$ is an unfounded set also w.r.t. N. But $N^- = M^-$; so, N is a total model of $\mathcal{LP}$ by Fact 2.

CLAIM 52 Let $\mathcal{LP}$ be a logic program and I be an interpretation for $\mathcal{LP}$. Then for each interpretation $J \supseteq I$,

$$\mathbf{T}^{\infty}_{pos(\mathcal{LP},I)}(\emptyset) \subseteq T^{\infty}_{pos(\mathcal{LP},J)}(\emptyset).$$

PROOF. It follows from the fact that, by definition of pos, $pos(\mathcal{LP}, I) \subseteq pos(\mathcal{LP}, J)$.

LEMMA 21
Let $\mathcal{F}$ be a deterministic family of pre-stable interpretations of a logic program $\mathcal{LP}$. Let I be the union of all interpretations in $\mathcal{F}$. Then I is a pre-stable interpretation of $\mathcal{LP}$.

PROOF. Let I be the union of all interpretations in $\mathcal{F}$. By definition of deterministic family, I is an interpretation of $\mathcal{LP}$. Let N be any pre-stable interpretation in $\mathcal{F}$. By definition of pre-stability, $\neg N^-$ is an unfounded set w.r.t. N. So, as $N \subseteq I$, $\neg N^-$ is an unfounded set also w.r.t. I; therefore, $\neg I^-$ is a union of unfounded sets w.r.t. I. It follows that $\neg I^-$ is an unfounded set w.r.t. I. Moreover, as every N in $\mathcal{F}$ is pre-founded, $I^+ = \cup_{N \in \mathcal{F}} \mathbf{T}^{\infty}_{pos(\mathcal{LP},N)}(\emptyset)$. But for each $N \in \mathcal{F}$, $N \subseteq I$; so $\mathbf{T}^{\infty}_{pos(\mathcal{LP},N)}(\emptyset) \subseteq \mathbf{T}^{\infty}_{pos(\mathcal{LP},I)}(\emptyset)$ by Claim 52. Hence, $I^+ \subseteq \mathbf{T}^{\infty}_{pos(\mathcal{LP},I)}(\emptyset)$. Moreover, as $\neg I^-$ is an unfounded set w.r.t. I, $\mathbf{T}_{\mathcal{LP}}(I) \cap \neg I^- = \emptyset$ by Claim 49 and, then, $\mathbf{T}^{\infty}_{pos(\mathcal{LP},I)}(\emptyset) \subseteq I^+$ by Claim 50. Therefore, $I^+ = \mathbf{T}^{\infty}_{pos'(\mathcal{LP},I)}(\emptyset)$, i.e., I is pre-founded. It follows that I is pre-stable.

LEMMA 22
Let $\mathcal{LP}$ be a program and M be a model of $\mathcal{LP}$. If M is pre-founded then

1. $\mathbf{W}^{\alpha}_{\mathcal{LP}}(M)$, for each countable ordinal α, is a monotonic sequence of pre-founded models, and
2. $\mathbf{W}^{\infty}_{\mathcal{LP}}(M)$ is a P-stable model of $\mathcal{LP}$.

PROOF. (1) Let M_α denote $\mathbf{W}^{\alpha}_{\mathcal{LP}}(M)$ for each countable ordinal α. We prove this part of the lemma by transfinite induction.

($\alpha = 0$ — *basis of the transfinite induction*). $M_0 = M$ is a pre-stable interpretation of $\mathcal{LP}$ by hypothesis.

($\alpha > 0$ — *inductive step*) Suppose that for each $\beta < \alpha$, M_β is a pre-stable interpretation and for each γ, $\beta < \gamma < \alpha$, $M_\beta \subseteq M_\gamma$ (*inductive hypothesis*). There are two possible cases:

(i) $\alpha = \gamma + 1$ is a successor ordinal; so $M_\alpha = \mathbf{W}_{\mathcal{LP}}(M_\gamma)$. Since M_γ is pre-stable by inductive hypothesis, $\neg M_\gamma^-$ is an unfounded set w.r.t. M_γ and, then, $\neg M_\gamma^- \subseteq \mathbf{U}_{\mathcal{LP}}(M_\gamma) = \neg M_\alpha^-$. Moreover, $M_\gamma^+ \subseteq \mathbf{T}_{\mathcal{LP}}(M_\gamma) = M_\alpha^+$ by Fact 3; hence, $M_\gamma \subseteq M_\alpha$. Let us now show that M_α is pre-stable interpretation. As M_γ is a pre-stable interpretation by inductive hypothesis and, then, pre-founded, $\mathbf{T}_{\mathcal{LP}}(M_\gamma) \cap \mathbf{U}_{\mathcal{LP}}(M_\gamma) = \emptyset$ by Claim 51; so, as $M_\alpha^+ = \mathbf{T}_{\mathcal{LP}}(M_\gamma)$ and $\neg M_\alpha^- = \mathbf{U}_{\mathcal{LP}}(M_\gamma)$, M_α is an interpretation. Moreover, as $\neg M_\alpha^-$ is an unfounded set w.r.t. M_γ, $\neg M_\alpha^-$ is an unfounded set also w.r.t. M_α because $M_\gamma \subseteq M_\alpha$. It remains to prove that M_α is pre-founded. To this end, observe that, as $M_\gamma \subseteq M_\alpha$,

$\mathbf{T}^{\infty}_{pos(\mathcal{LP},M_\gamma)}(\emptyset) \subseteq \mathbf{T}^{\infty}_{pos(\mathcal{LP},M_\alpha)}(\emptyset)$ by Claim 52. Moreover, as M_γ is pre-founded, $M^+_\gamma = \mathbf{T}^{\infty}_{pos(\mathcal{LP},M_\gamma)}(\emptyset)$; so $M^+_\gamma \subseteq \mathbf{T}^{\infty}_{pos(\mathcal{LP},M_\alpha)}(\emptyset)$. Consider now any element $A \in M^+_\alpha$. Since $M^+_\alpha = \mathbf{T}_{\mathcal{LP}}(M_\gamma)$, there exists a rule $r \in ground(\mathcal{LP})$ such that $H(r) = A$ and $G(r) \subseteq M_\gamma$. Hence there exists a rule r' in $pos(\mathcal{LP}, M_\alpha)$ such that $H(r') = A$ and $G(r') \subseteq M^+_\gamma$. Then A is in $\mathbf{T}_{pos(\mathcal{LP},M_\alpha)}(M^+_\gamma)$. Hence, as $M^+_\gamma \subseteq \mathbf{T}^{\infty}_{pos(\mathcal{LP},M_\alpha)}(\emptyset)$, $M^+_\alpha \subseteq \mathbf{T}^{\infty}_{pos(\mathcal{LP},M_\alpha)}(\emptyset)$. But $\mathbf{T}^{\infty}_{pos(\mathcal{LP},M_\alpha)}(\emptyset) \subseteq \mathbf{T}_{\mathcal{LP}}(M_\alpha) \cap (M^+_\alpha \cup \neg M^-_\alpha)$ by Claim 50 and $\mathbf{T}_{\mathcal{LP}}(M_\alpha) \cap \neg M^-_\alpha = \emptyset$ by Claim 51; so $\mathbf{T}^{\infty}_{pos(\mathcal{LP},M_\alpha)}(\emptyset) \subseteq M^+_\alpha$ and, then, $M^+_\alpha = T^{\infty}_{pos(\mathcal{LP},M_\alpha)}(\emptyset)$. It follows that M_α is a pre-stable interpretation for $\mathcal{LP}$.

(ii) α is a limit ordinal; so $M_\alpha = \cup_{\gamma<\alpha} M_\gamma$. Hence, for each $\beta < \alpha$, $M_\beta \subseteq M_\alpha$. It remains to prove that M_α is a pre-stable interpretation. To this end, observe that, by inductive hypothesis, $\mathcal{F} = \{M_\beta | \beta < \alpha\}$ is a family of pre-stable interpretations such that for each pair of interpretations in $\mathcal{F}$, one of them is contained in the other. Therefore $\mathcal{F}$ is a deterministic family of pre-stable interpretations; so, by Lemma 21, M is a pre-stable interpretation of $\mathcal{LP}$.

(2) Let $N = \mathbf{W}^{\infty}_{\mathcal{LP}}(M)$. Then there exists a countable ordinal β such that $N = \mathbf{W}^{\beta}_{\mathcal{LP}}(N)$; so, as proven in Part (1) of this lemma, N is a pre-stable interpretation of $\mathcal{LP}$. Finally, as N is a fixpoint of $\mathbf{W}_{\mathcal{LP}}$, N is a P-stable model by Part (2) of Proposition 16.