Deterministic Semantics for Disjunctive Logic Programs

Gianluigi Greco, Sergio Greco, Ester Zumpano

DEIS Department

Università della Calabria

87030 Rende, Italy

{ggreco, greco, zumpano}@si.deis.unical.it

Abstract

In this paper we propose a deterministic semantics for disjunctive logic programs based on partial stable models. Deterministic models are (partial) stable models which are not contradicted by any other stable model, i.e. M is a deterministic model if there is no stable model N such that $M \cup N$ is not an interpretation. For instance, for not disjunctive programs, the well-founded model which coincides with the intersection of all partial stable models, is a deterministic model.

The deterministic semantics of disjunctive programs is based on the assumption that atoms appearing in the head of disjunctive rules may be undefined also if the body of the rule is true, whenever we are not able to say with certainty which atoms are true and which atoms are false. Under such an assumption, all programs have at least one deterministic model and there exists a deterministic model, called max-deterministic, which contains all the others deterministic models; this model, consisting of the maximal set of true and false atoms which does not contradict others partial models, defines the semantics of disjunctive programs.

1 Introduction

Total stable (T-stable) models have been first introduced to define the semantics of normal (not disjunctive) logic programs [11]. General programs may have zero, one or more T-stable models. This means that there are programs which do not have semantics and programs which have a nondeterministic semantics (programs with multiple stable models). Subsequently, stable models have been extended to partial interpretations (*P-stable models* corresponding to the three-valued stable models of

Work partially supported by a MURST grant under the projects "Data-X" and "D2I "and by the CNR Research Institute ISI.

[19] and the strongly-founded models of [24]). Every program has at least one partial stable model and their intersection is a model (namely the well-founded model).

Total stable models have been further extended for disjunctive programs [12, 20]. In particular, every disjunctive program may have zero or multiple models and positive (negation free) programs have at least one stable model. Partial stable models have been extended for disjunctive programs too and every program has at least one partial model [8].

The fact that a program may have several P-stable models does not necessarily impose a non-deterministic semantics [2]; indeed, various types of deterministic semantics are known, e.g. the *possibility* semantics (based on the union of all models) and the *certainty semantics* (based on their intersection). For not disjunctive programs, certainty semantics is actually captured by one P-stable model only, namely, the *well founded model* [29], that coincides with the intersection of all P-stable models. On the other hand, possibility semantics cannot be captured by a single P-stable model since the union of all P-stable models is not in general a model.

A P-stable model M is *deterministic* if for each other P-stable model N, $N \cup M$ is an interpretation, i.e. M does not contradict any other P-stable model [24]. The family of deterministic models admit a minimum (the *well founded model*) and a maximum (the *max-deterministic model*). Thus, the max-deterministic model is defined as the maximum of the family of deterministic P-stable models, i.e. it includes each model in the family and, therefore, it coincides with the union of all deterministic models. Actually, the family of deterministic models has the structure of a complete lattice: the bottom of the lattice coincides with the well-founded model, while the top of the lattice is the max-deterministic.

However, disjunctive stable models have a strange anomaly since there are programs which have the same semantics under total stable model semantics and different semantics under the partial stable model semantics.

Consider, for instance the following two programs $P_1 = \{\mathtt{a} \leftarrow \neg \mathtt{b},\ \ \mathtt{b} \leftarrow \neg \mathtt{a}\}$ and $P_2 = \{\mathtt{a} \vee \mathtt{b} \leftarrow\}$. Under the total stable model semantics, both P_1 and P_2 have two stable models: $M_1 = \{\mathtt{a}, \neg \mathtt{b}\}$ and $M_2 = \{\neg \mathtt{a}, \mathtt{b}\}$, that is, they have different syntax but the same semantics. The motivation of this is that stable (and minimal) model semantics interprets the disjunction in the head of the rule of P_2 as 'exclusive' and, therefore, P_2 could be rewritten as P_1.

Moreover, if we consider partial stable models, P_1 has three models, namely M_1, M_2 and $M_3 = \{\ \}$ whereas P_2 has two minimal models: M_1 and M_2. Thus, P_1 and P_2 have now different semantics, P_1 has a deterministic model (M_3) whereas P_2 does not have deterministic semantics.

In this paper we propose a simple variation of partial stable model semantics for disjunctive logic programs which interprets exclusive disjunction in the same way, independently from the syntax (e.g. P_1 and P_2 above have the same semantics, i.e. the same set of partial stable models).

The following example presents another program where disjunction is interpreted in a different way under total and partial semantics:

Example 1.1 Consider the following program P_1

$$\begin{array}{lll} \mathtt{a} \vee \mathtt{b} & \leftarrow & \\ \mathtt{c} & \leftarrow & \neg\mathtt{d},\ \mathtt{a},\ \mathtt{b}. \\ \mathtt{d} & \leftarrow & \neg\mathtt{c}. \\ \mathtt{e} & \leftarrow & \neg\mathtt{e}. \end{array}$$

Total stable model semantics interprets the first rule as an exclusive disjunction but, because of the last rule, P_1 has not total semantics since it only has the two P-stable models $M_1 = \{\mathtt{a}, \neg\mathtt{b}, \neg\mathtt{c}, \mathtt{d}\}$ and $M_2 = \{\neg\mathtt{a}, \mathtt{b}, \neg\mathtt{c}, \mathtt{d}\}$. By rewriting, the first rule into the two rules

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

we get a not disjunctive program P_2 which has four partial stable models, namely: M_1, M_2, $M_3 = \{\}$ and $M_4 = \{\neg\mathtt{c}, \mathtt{d}\}$. The models M_3 and M_4 are deterministic because the atoms in these two models are not contradicted by any other model (for instance, the model M_4 is deterministic since there is no model containing either $\mathtt{c}$ or $\neg\mathtt{d}$). Moreover, the program does not have total stable model since we cannot assign truth value to $\mathtt{e}$. □

Deterministic stable models for not disjunctive programs have been proposed in [25] and their complexity and expressivity has been studied in [14]. Observe that for not disjunctive programs, the deterministic semantics is strongly connected with intrinsic stable models and intrinsic fixpoint introduced by Fitting [9, 10]. More specifically, deterministic stable models coincide with *intrinsic stable models* [9] (based on *intrinsic fixpoint* [10]) whereas the max-deterministic model coincides with the *prudently brave* model [9] based on the *largest intrinsic fixpoint* [10]. In this paper we extend definitions and results on deterministic semantics to disjunctive programs.

We recall that for disjunctive programs deterministic semantics have been proposed as well. We mention here the *static semantics* proposed by Przymusinski in [21] and the *disjunctive well-founded semantics D-WFS* proposed by Brass and Dix in [4]. The two semantics are based on very different intuitions and are constructed by completely different means. While D-WFS is defined as the least semantics that is invariant under certain natural program transformations, the static semantics is obtained by first translating a logic program into a belief theory in the *Autoepistemic Logic of Beliefs (AEB)* and then constructing its least static expansion (defined as the least fixed point of a natural monotonic operator). Although the two semantics derive from very different ideas, both of them have been shown to share a number of natural and intuitive properties and, in particular, both of them extend the well-founded semantics of normal logic programs [5]. The deterministic semantics here studied is different and based on a different interpretation of disjunctive rules.

The rest of the paper is organized as follows. In Section 2 we recall basic concepts on the syntax and semantics of disjunctive logic programming. In Section 3 we introduce weak partial stable models for disjunctive programs. In section 4 we present an

exact characterization of the nondeterministic semantics for disjunctive programs. In Section 5 we introduce deterministic semantics for disjunctive programs. Finally, in Section 6, we present our conclusions.

2 Preliminary Definitions

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

$$A_1 \vee \cdots \vee A_k \leftarrow B_1, \cdots, B_m, \neg C_1, \cdots, \neg C_n, \qquad k + m + n > 0.$$

$A_1, \cdots, A_k, B_1, \cdots, B_m, C_1, \cdots, C_n$ are atoms of the form $p(t_1, ..., t_h)$, where p is a *predicate* of arity h and $t_1, ..., t_h$ are terms. A term is either a constant, a variable or a structure of the form $f(t_1, ..., t_h)$ where f is a function symbol and $t_1, ..., t_h$ are terms. The disjunction $A_1 \vee \cdots \vee A_n$ is the *head* of r, while the conjunction $B_1, \cdots, B_m, \neg C_1, \cdots, \neg C_n$ is the *body* of r. Moreover, if $k = 1$ we say that the rule is normal, i.e. not disjunctive or $\vee$-free, whereas if $n = 0$ we say that the rule is positive, i.e. $\neg$-free.

We denote by $Head(r)$ the set $\{A_1, ..., A_k\}$ of head atoms, and by $Body(r)$ the set $\{B_1, \cdots, B_m, \neg C_1, \cdots, \neg C_n\}$ of body literals. We often use upper-case letters, say L, to denote literals. As usual, a literal is an atom A or a negated atom $\neg A$; in the former case, it is positive, and in the latter negative. Two literals are *complementary*, if they are of the form A and $\neg A$, for some atom A. For a literal L, $\neg L$ denotes its complementary literal, and for a set S of literals, $\neg S = \{\neg L \mid L \in A\}$. Moreover, $Body^+(r)$ and $Body^-(r)$ denote the set of positive and negative literals occurring in $Body(r)$, respectively.

A *(disjunctive) logic program* is a finite set of rules. A $\neg$-free (resp. $\vee$-free) program is called *positive* (resp. *normal*). A term, (resp. an atom, a literal, a rule or a program) is *ground* if no variables occur in it.

The *Herbrand Universe* $U_\mathcal{P}$ of a program $\mathcal{P}$ is the set of all ground terms which can be constructed by using function symbols and constants appearing in $\mathcal{P}$, and its *Herbrand Base* $B_\mathcal{P}$ is the set of all ground atoms constructed from the predicates appearing in $\mathcal{P}$ and the ground terms from $U_\mathcal{P}$. A rule r' is a *ground instance* of a rule r, if r' is obtained from r by replacing every variable in r with some ground term in $U_\mathcal{P}$. We denote by $ground(\mathcal{P})$ the set of all ground instances of the rules in $\mathcal{P}$.

Given a program $\mathcal{P}$ and two predicate symbols (resp. ground atoms) p and q, we write $p \rightarrow q$ if there exists a rule where q occurs in the head and p in the body or there exists a predicate (resp. ground atom) s such that $p \rightarrow s$ and $s \rightarrow q$. If $p \rightarrow q$ then we say that q *depends on* p; also we say that q *depends on* any rule where p occurs in the head. A predicate (resp. ground atom) p is said to be recursive if $p \rightarrow p$.

An interpretation of $\mathcal{P}$ is any subset of $B_\mathcal{P}$. The value of a ground atom L w.r.t. an interpretation I, $value_I(L)$, is $true$ if $L \in I$ and $false$ otherwise. The value of a ground negated literal $\neg L$ is $\neg value_I(L)$. Assuming the order $false <$

true, the truth value of a conjunction of ground literals $C = L_1, \ldots, L_n$ is the minimum over the values of the L_i, i.e. $value_I(C) = min(\{value_I(L_i) \mid 1 \leq i \leq n\})$, while the value $value_I(D)$ of a disjunction $D = L_1 \vee ... \vee L_n$ is its maximum, i.e. $value_I(D) = max(\{value_I(L_i) \mid 1 \leq i \leq n\})$; if $n = 0$, then $value_I(C) = true$ and $value_I(D) = false$. Finally, a ground rule r is *satisfied* by I if $value_I(Head(r)) \geq value_I(Body(r))$. Thus, a rule r with empty body is satisfied by I if $value_I(Head(r)) = true$ whereas a rule r' with empty head is satisfied by I if $value_I(Body(r)) = false$. An interpretation M for $\mathcal{P}$ is a model of $\mathcal{P}$ if M satisfies each rule in $ground(\mathcal{P})$. We also assume the existence of the two predefined atoms T and F whose truth value is $true$ and $false$, respectively ($value_I(T) = true$ and $value_I(F) = false$ for all interpretations I).

Minker proposed in [18] a model-theoretic semantics for positive programs $\mathcal{P}$, which assigns to $\mathcal{P}$ the set of its *minimal models* $\mathcal{MM}(\mathcal{P})$, where a model M for $\mathcal{P}$ is minimal, if no proper subset of M is a model for $\mathcal{P}$. Accordingly, the program $\mathcal{P} = \{a \vee b \leftarrow\}$ has the two minimal models $\{a\}$ and $\{b\}$, i.e. $\mathcal{MM}(\mathcal{P}) = \{\ \{a\},\ \{b\}\ \}$.

The more general *disjunctive stable model semantics* also applies to programs with (unstratified) negation [12, 20]. Disjunctive stable model semantics generalizes stable model semantics, previously defined for normal programs [11].

Definition 2.1 For any interpretation M, denote with $\mathcal{P}^M$ the ground positive program derived from $ground(\mathcal{P})$ i) by removing all rules that contain a negative literal $\neg a$ in the body and $a \in M$, and ii) by removing all negative literals from the remaining rules. An interpretation M is a (disjunctive) total stable model of $\mathcal{P}$ if and only if $M \in \mathcal{MM}(\mathcal{P}^{\mathcal{M}})$. □

For normal $\mathcal{P}$, the total stable model semantics assigns to $\mathcal{P}$ the set $\mathcal{TS}(\mathcal{P})$ of its *total stable models*. It is well known that stable models are minimal models (i.e. $\mathcal{TS}(\mathcal{P}) \subseteq \mathcal{MM}(\mathcal{P})$) and that for negation free programs minimal and stable model semantics coincide (i.e. $\mathcal{TS}(\mathcal{P}) = \mathcal{MM}(\mathcal{P})$).

2.1 P-stable Models

As for total stable models [11, 20], the definition of partial stable models requires that every positive literal in an interpretation must be derivable from the rules possibly using negative literals as additional axioms.

For total interpretations foundness alone is sufficient to characterize (2-valued) stable models, which are indeed defined as the total founded interpretations of the program [11, 12, 20]. However, as for normal programs [24], foundness alone is not sufficient to single out partial stable models, as some conditions on the false literals are needed. Intuitively, what must be imposed on the false literals for a founded interpretation I to be a (partial) stable model is the consistency with the closed world assumption. In other words, every false literal in I must be definitely not derivable from the rules of the program (assuming I), and no further literal

must be declarable false without violating some rule of the program. We assume here that atoms may also be undefined and the following order among truth values: $false < undefined < true$. We also assume the existence of the predefined atom U whose truth value is undefined (i.e. $value_I(U) = undefined$ for all interpretation I).

Definition 2.2 Let $\mathcal{P}$ be a disjunctive program and let M be a partial interpretation. Then, $\mathcal{P}^M$ denotes the ground program derived from $ground(\mathcal{P})$ by replacing every negative body literal $\neg C$ with its truth value in I, i.e. by replacing every $\neg C$ which is true (resp. false, undefined) in I with T (resp. F, U).

A (partial) interpretation M for $\mathcal{P}$ is a (partial) stable model for $\mathcal{P}$ if M is a minimal model for $\mathcal{P}^M$. □

Observe that in the construction of $\mathcal{P}^M$ rules having F in the body can be deleted and that the atom T can be deleted from the body of rules. Therefore, the above definition of $\mathcal{P}^M$ generalizes the definition of $\mathcal{P}^M$ introduced in the previous subsection.

2.2 Restricted Classes of P-stable Models

As for normal ($\vee$-free) logic programs [23], P-stable models are grouped into three main families: T-stable, M-stable, and L-stable models.

Definition 2.3 [8] A P-stable model M is:

(a) *T-stable* (*Total stable*) if M is a total interpretation;

(b) *M-stable* (*Maximal stable*) if there exists no P-stable model N of $\mathcal{P}$ such that $N \supset M$.

(c) *L-stable* (*Least-undefined stable*) if the set of its undefined atoms is minimal, i.e. there exists no P-stable model N of $\mathcal{P}$ such that $\overline{N} \supset \overline{M}$. □

Note that the T-stable models coincide with the 2-valued (total disjunctive) stable models of [20]. Given a program $\mathcal{P}$, the class of all partial (resp. total, maximal, least-undefined) stable models of $\mathcal{P}$ will be denoted by $\mathcal{PS}(\mathcal{P})$ (resp. $\mathcal{TS}(\mathcal{P})$, $\mathcal{MS}(\mathcal{P})$, $\mathcal{LS}(\mathcal{P})$) or simply by $\mathcal{PS}$ (resp. $\mathcal{TS}$, $\mathcal{MS}$, $\mathcal{LS}$) if the program $\mathcal{P}$ is understood.

Example 2.4 The P-stable models of the program $\mathcal{P}_1$ in Example 1.1 are $M_1 = \{\mathtt{a},\ \neg\mathtt{b},\ \neg\mathtt{c},\ \mathtt{d}\}$ and $M_2 = \{\neg\mathtt{a},\ \mathtt{b},\ \neg\mathtt{c},\ \mathtt{d}\}$ which also are M-stable and L-stable but not T-stable since the atom `e` is undefined. However, if we delete the last rule defining `e`, both the models M_1 and M_2 become T-stable.

Consider the following program $\mathcal{P}_2$.

$$\begin{aligned} \mathtt{a} &\leftarrow \neg\mathtt{b} \\ \mathtt{b} &\leftarrow \neg\mathtt{a} \\ \mathtt{c} \vee \mathtt{d} &\leftarrow \mathtt{a} \\ \mathtt{d} &\leftarrow \mathtt{c} \end{aligned}$$

The P-stable models of $\mathcal{P}_2$ are: $M_1 = \{\mathtt{b},\ \neg\mathtt{a},\ \neg\mathtt{c},\ \neg\mathtt{d}\}$, $M_2 = \{\mathtt{a},\ \mathtt{d},\ \neg\mathtt{b},\ \neg\mathtt{c}\}$, $M_3 = \{\neg\mathtt{c}\}$ and $M_4 = \{\ \}$. M_1 and M_2 are also M-stable models, while M_3 is not (as it is a subset of the P-stable model M_1, as well as of M_2). Both M_1 and M_2 are also L-stable models for $\mathcal{P}_2$. Moreover, M_1 and M_2 are also T-stable models, as they are total interpretations. □

It is worth noting that an L-stable is not total in general. For instance, both the L-stable models M_1 and M_2 of Example 1.1 are not T-stable.

3 Weak partial stable models

Disjunctive rules express a sort of nondeterminism. Take for instance the disjunctive rule $a \vee b \leftarrow$. The standard minimal model semantics states that there are two minimal models $M_1 = \{a, \neg b\}$ and $M_2 = \{\neg a, b\}$. Thus, under the certain version of minimal model semantics we conclude that both a and b are undefined since the literals a, b, $\neg a$ and $\neg b$ do not belong to all models. Moreover, as observed in the introduction, both minimal and stable model semantics interpret the above disjunction as exclusive and it could be rewritten into a not disjunctive program; the rewritten program also has a further model M_3 in which both a and b are undefined.

In this paper we propose a variation of partial stable model semantics called *weak partial (WP) stable model semantics*, based on a different interpretation of disjunctive rules.

Definition 3.1 An interpretation M for a disjunctive program $\mathcal{P}$ is a weak partial model for $\mathcal{P}$, if M is a P-stable model for the program $w(\mathcal{P})$ derived from $\mathcal{P}$ by rewriting every disjunctive rule r

$$a_1 \vee \cdots \vee a_k \leftarrow b_1, ..., b_n$$

into the rule r'

$$a_1 \vee \cdots \vee a_k \leftarrow b_1, ..., b_n, U$$

where U denotes the predefined *unknown* value. □

Observe that the above rule r' is satisfied if $max(\{val(a_1), ..., val(a_k)\}) \geq min(\{val(b_1), ..., val(b_n), undefined\})$.

Analogously, a WP-stable model M for a program $\mathcal{P}$ is:

(a) *WT-stable* (*Weak Total stable*) if M is a T-stable model for $w(\mathcal{P})$;

(b) *WM-stable* (*Weak Maximal stable*) if M is a M-stable model for $w(\mathcal{P})$;

(c) *WL-stable* (*Weak Least-undefined stable*) if M is a L-stable model for $w(\mathcal{P})$.

The family of all weak partial (resp. weak total, weak maximal, weak least-undefined) stable models of $\mathcal{P}$ will be denoted by $\mathcal{WPS}(\mathcal{P})$ (resp. $\mathcal{WTS}(\mathcal{P})$, $\mathcal{WMS}(\mathcal{P})$, $\mathcal{WLS}(\mathcal{P})$) or simply by $\mathcal{WPS}$ (resp. $\mathcal{WTS}$, $\mathcal{WMS}$, $\mathcal{WLS}$) if the program $\mathcal{P}$ is understood.

Lemma 3.2 *For every disjunctive program $\mathcal{P}$*

1. $\mathcal{PS}(\mathcal{P}) \subseteq \mathcal{WPS}(\mathcal{P})$,
2. $\mathcal{TS}(\mathcal{P}) \subseteq \mathcal{WTS}(\mathcal{P})$,
3. $\mathcal{MS}(\mathcal{P}) \subseteq \mathcal{WMS}(\mathcal{P})$ *and*
4. $\mathcal{LS}(\mathcal{P}) \subseteq \mathcal{WLS}(\mathcal{P})$. □

The above proposition states that every P-stable (resp. T-stable, M-stable, L-stable) model for $\mathcal{P}$ is also WP-stable (resp. WT-stable, WM-stable, WL-stable). Moreover, for maximal stable model semantics we also have that every weak maximal stable model is also maximal stable.

Theorem 3.3 *For every disjunctive program $\mathcal{P}$, $\mathcal{MS}(\mathcal{P}) = \mathcal{WMS}(\mathcal{P})$.* □

Since $\mathcal{TS}(\mathcal{P}) \subseteq \mathcal{LS}(\mathcal{P}) \subseteq \mathcal{MS}(Q)$ and $\mathcal{WTS}(\mathcal{P}) \subseteq \mathcal{WLS}(\mathcal{P}) \subseteq \mathcal{WMS}(Q)$, we have the following corollary.

Corollary 3.4 *For every disjunctive program $\mathcal{P}$*

1. $\mathcal{TS}(\mathcal{P}) = \mathcal{WTS}(\mathcal{P})$ *and*
2. $\mathcal{LS}(\mathcal{P}) = \mathcal{WLS}(\mathcal{P})$. □

Moreover, for normal (not disjunctive) programs partial stable models and weak partial stable models coincide.

Corollary 3.5 *For every normal program $\mathcal{P}$, $\mathcal{PS}(\mathcal{P}) = \mathcal{WTS}(\mathcal{P})$.* □

4 Non-Deterministic Stable Models

The family of WP-stable models is never strictly non-deterministic and, therefore, it is necessary to identify the maximal subfamily that has this property.

Recall that for every program $\mathcal{P}$, a WP-stable model of $\mathcal{P}$ is *M-stable* (*weak maximally stable*) if it is not a proper subset of any other P-stable (WP-stable) model of $\mathcal{P}$.

Theorem 4.1 *For each logic program $\mathcal{P}$, $\langle \mathcal{WPS}(\mathcal{P}), \subseteq \rangle$ is a (not empty) Noetherian meet semi-lattice,*[1] *whose top elements are all M-stable models for $\mathcal{P}$.* □

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

The bottom of the semilattice $\langle \mathcal{WPS}(\mathcal{P}), \subseteq \rangle$ is a weak partial model called *well-founded.*

Theorem 4.2 *For every logic program $\mathcal{P}$, any two distinct models in $\mathcal{WMS}(\mathcal{P})$ (and $\mathcal{MS}(\mathcal{P})$) are mutually exclusive.* □

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

Corollary 4.3 *For every logic program $\mathcal{P}$, $\mathcal{WPS}(\mathcal{P})$ is a deterministic family if and only if $|\mathcal{WMS}(\mathcal{P})| = 1$.* □

Every subfamily of $\mathcal{WMS}$ with multiple elements is strictly non-deterministic and $\mathcal{WTS} \subseteq \mathcal{WMS}$.

5 Deterministic WP-Stable Models

In the previous section, we have shown that the family of WM-stable models captures the whole potential non-determinism of WP-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 $\mathcal{WPS}$ models which are deterministic.

Definition 5.1 A WP-stable model M for a program $\mathcal{P}$ is said to be deterministic if there is no WP-stable model N for $\mathcal{P}$ such that $M \cup N$ is not consistent. □

In the following $\mathcal{WPS}$-deterministic models are denoted by $\mathcal{WDS}$ models.

Theorem 5.2 *A WP-stable model of a logic program $\mathcal{P}$ is $\mathcal{WDS}$ if and only if it is contained in the intersection of all WM-stable models of $\mathcal{P}$.* □

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

Example 5.3 Consider the following program:

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

There are three WP-stable models: $M_1 = \{\mathtt{b}, \neg\mathtt{a}, \ \mathtt{q}_2, \neg\mathtt{q}_1, \neg\mathtt{u}\}$, $M_2 = \{\mathtt{a}, \neg\mathtt{b}, \mathtt{q}_1, \neg\mathtt{q}_2, \neg\mathtt{u}\}$ and $M_3 = \{\ \}$. M_1 and M_2 are M-stable and also T-stable. The intersection of $M_1 \cap M_2 = \{\neg\mathtt{u}\}$ is not a weak partial model. The only $\mathcal{WDS}$ model is the empty set. □

Example 5.4 Consider the following program:

$$\begin{array}{lll} \mathtt{a}. & & \\ \mathtt{p} \vee \mathtt{q} & \leftarrow & \mathtt{a}. \\ \mathtt{r} & \leftarrow & \mathtt{p}. \\ \mathtt{r} & \leftarrow & \mathtt{q}. \end{array}$$

Here there are two M-stable models $M_1 = \{\mathtt{a}, \mathtt{p}, \mathtt{r}, \neg\mathtt{q}\}$ and $M_2 = \{\mathtt{a}, \mathtt{q}, \mathtt{r}, \neg\mathtt{p}\}$. In this case, their intersection $\{\mathtt{a}, \mathtt{r}\}$ is a model but not WP-stable. The only $\mathcal{WDS}$ model is $M_3 = \{\mathtt{a}\}$. Thus, both a and r are deterministic implications but only a is founded (supported). □

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

Theorem 5.5 *For every logic program $\mathcal{P}$, $\langle \mathcal{WDS}, \subseteq \rangle$ is a complete lattice.* □

Thus, for any two $\mathcal{WDS}$ models M_1, M_2 , $M_1 \cap M_2$ and $M_1 \cup M_2$ are both deterministic models. The top of the lattice of $\mathcal{WDS}$ models for a program $\mathcal{P}$ will be called the *maximum deterministic model* for $\mathcal{P}$ (denoted by $\mathcal{MDS}$) whereas the bottom is called well-founded (denoted by $\mathcal{WF}$). Well-founded and max-deterministic models for disjunctive programs generalize well-founded and max-deterministic models for not disjunctive programs.

Proposition 5.6 *Let $\mathcal{P}$ be a disjunctive program. Then*

1. *The well-founded model M is the least $\mathcal{WDS}$ model of $\mathcal{P}$,*
2. *$\mathcal{WDS}$ is neither empty nor strictly non-deterministic.* □

Traditionally, for not disjunctive programs, 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. A more general approach consists in taking the agreement set among all its M-stable models, and find P-stable models that belong to this agreement set [25].

Example 5.7 Consider the following program:

$$\begin{array}{lll} \mathtt{b} \vee \mathtt{c} & \leftarrow & \mathtt{a}. \\ \mathtt{p} & \leftarrow & \mathtt{b},\ \neg\mathtt{p}. \\ \mathtt{d} \vee \mathtt{e} & \leftarrow & \mathtt{a},\ \neg\mathtt{p}. \\ \mathtt{q} & \leftarrow & \neg\mathtt{d},\ \neg\mathtt{q}. \\ \mathtt{a}. & & \end{array}$$

The P-stable models are: $M_1 = \{\mathtt{a}\}$, $M_2 = \{\mathtt{a}, \mathtt{b}, \neg\mathtt{c}\}$, $M_3 = \{\mathtt{a}, \neg\mathtt{b}, \mathtt{c}, \neg\mathtt{p}\}$, $M_4 = \{\mathtt{a}, \neg\mathtt{b}, \mathtt{c}, \neg\mathtt{p}, \neg\mathtt{d}, \mathtt{e}\}$, $M_5 = \{\mathtt{a}, \neg\mathtt{b}, \mathtt{c}, \neg\mathtt{p}, \mathtt{d}, \neg\mathtt{e}, \neg\mathtt{q}\}$.

M_1 is the well-founded model and is contained in every WP-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. □

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

Example 5.8 Take the following logic program:

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

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

The significance of the $\mathcal{MDS}$ model can be appreciated through the following example based on Raymond Smullyan's puzzles [27].

Example 5.9 An island is inhabited by knights and knaves: the knaves always lie, while the knights always tell the truth. A pair of individuals, a and b, are asked about their nature and individual a answers: "Only one of us is a knight". What can be inferred about individual b?

Each individual is either a knight or a knave and there are individuals: a and b:

$$\begin{array}{l} \mathtt{ind(a)}. \\ \mathtt{ind(b)}. \\ \mathtt{knight(X)} \vee \mathtt{knave(X)} \ \leftarrow \end{array}$$

If a is knight then he tells the truth; so b is different, i.e. a knave. On the other hand, if a is a knave, and, therefore a liar, then b must be of the same kind, i.e. a knave:

$$\begin{array}{lll} \mathtt{knave(b)} & \leftarrow & \mathtt{knight(a)}. \\ \mathtt{knave(b)} & \leftarrow & \mathtt{knave(a)}. \end{array}$$

The above program has two T-stable models: $M_1 = \{\mathtt{ind(a)}, \mathtt{ind(b)}, \mathtt{knight(a)}, \mathtt{knave(b)}, \neg\mathtt{knave(a)}, \neg\mathtt{knight(b)}\}$ and $M_2 = \{\mathtt{ind(a)}, \mathtt{ind(b)}, \mathtt{knave(a)}, \mathtt{knave(b)}, \neg\mathtt{knight(a)}, \neg\mathtt{knight(b)}\}$.
The max-deterministic P-stable model instead is: $M_3 = \{\mathtt{ind(a)}, \mathtt{ind(b)}, \mathtt{knave(b)}, \neg\mathtt{knight(b)}\}$, while the well-founded model is M_4=$\{\mathtt{ind(a)}, \mathtt{ind(b)}\}$.
The max-deterministic model is capable of drawing all deterministic implications which follow from the given rules — in this case, the conclusion is that individual b is a knave, no matter what individual a is. On the other side, the well-founded model is not able to draw any conclusion. □

Observe that the $\mathcal{WDS}$ models trade off undefinedness for expressivity and 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 advantage and high expressive power are counterbalanced by an higher computational complexity. It is interesting to observe that, while well-founded models capture the certainty semantics of $\mathcal{WDS}$ models (i.e. the well founded model consists of all ground literals that are in every $\mathcal{WPS}$-deterministic model), maximum $\mathcal{WDS}$ models capture the possibility semantics (i.e. the maximum $\mathcal{WDS}$ model consists of all ground literals that are in some $\mathcal{WDS}$ model).

Proposition 5.10 *For every disjunctive program $\mathcal{P}$, every deterministic stable model for $\mathcal{P}$ is also WD-stable.* □

Theorem 5.11 *Let $\mathcal{P}$ be a disjunctive program and M a WDS-stable model for $\mathcal{P}$, then for every P-stable model N for $\mathcal{P}$, $M \cup N$ is consistent.* □

The above theorem is important since it states that WP-deterministic models do not contradict P-stable models, i.e. what is inferred by WP-deterministic models do not contradict what is inferred by P-stable models.

6 Conclusions

In this paper we have proposed a generalization of deterministic semantics for disjunctive logic programs. We have introduced WP-stable models, a generalization of P-stable models for normal programs and have identified subclasses of nondeterministic models (namely MS models) and deterministic models (namely WDS models). Deterministic WP-stable models trade off minimal undefinedness to achieve determinism, but in different degrees. At the bottom, we find the well-founded model, which ensures better computability at the expense of more undefinedness. At the top, we find the max-deterministic model. Thus, the max-deterministic model defines the deterministic semantics for (disjunctive) logic programs.

References

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

[2] Abiteboul, S., Simon, E., and Vianu, V. (1990), Non-deterministic languages to express deterministic transformations. *Proc. of the 9th ACM Symposium on Principles of Database Systems*, pp. 218-229.

[3] Baral, C., and Subrahmanian, V.S., (1992), Stable and Extension Class Theory for Logic Programs and Default Logic. *Journal of Automated Reasoning*, 8, pp. 345-366.

[4] Brass, S., and Dix, J. (1994), A Disjunctive Semantics Bases on Unfolding and Bottom-Up Evaluation. *GI Jahrestagung*, pp. 83-91.

[5] Brass, S., Dix, J., Niemela, I., Przymusinski, T. C. (2001), On the equivalence of the static and disjunctive well-founded semantics and its computation. *Theoretical Computer Science*, 258(1-2), pp. 523-553.

[6] Dung, P. (1991), Negation as Hypotheses: An Abductive Foundation for Logic Programming. *Proc. Int. Conf. on Logic Programming*, MIT Press, pp. 3-17.

[7] Eiter, T., Gottlob, G. and Mannila, H. (1997), Disjunctive Datalog. *ACM Transaction on Database Systems*, 22(3), pp. 364-418.

[8] Eiter, T., Leone, N., Saccà, D. (1998), Expressive Power and Complexity of Partial Models for Disjunctive Deductive Databases. *Theoretical Computer Science*, 206(1-2), pp. 181-218.

[9] Fitting, M. (1994), On prudent bravery and other abstractions. Unpublished manuscript.

[10] Fitting, M. (1985), A Kripke-Kleene Semantics for Logic Programs. *Journal of Logic Programming*, 2(4), pp. 295-312.

[11] Gelfond, M., and Lifschitz, V. (1988), The Stable Model Semantics for Logic Programming. *Proc. of the 5th Int. Conf. on Logic Programming*, pp. 1070-1080.

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

[13] Greco, S., and Saccà, D. (1997), 'Possible is Certain' is desirable and can be expressive. *Annals of Mathematics and Artificial Intelligence*, 19, pp. 147-168.

[14] Greco, S. and Saccà, D. (1999), Deterministic Semantics for Datalog$^{\neg}$: Complexity and Expressive Power. *Information and Computation*, 153(1), pp. 81-98.

[15] Greco, S., Saccà, D. and Zaniolo, C. (2001), Extending stratified datalog to capture complexity classes ranging from P to QH. *Acta Informatica,* 37(10), pp. 699-725.

[16] Lloyd, J. W. (1987), *Foundations of Logic Programming.* Springer-Verlag, Berlin.

[17] Marek, W., and Truszcynski, M. (1991), Autoepistemic Logic. *Journal of the Association for Computing Machinery*, 38(3), pp. 588-619.

[18] Minker, J. (1982), On Indefinite Data Bases and the Closed World Assumption. *Proc. 6th Conf. on Automated Deduction*, Lecture Notes in Computer Science, Vol. 138, Springer, pp. 292-308.

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

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

[21] Przymusinski, T.C. (1995), Static Semantics for Normal and Disjunctive Logic Programs. *Annals of Mathematics and Artificial Intelligence*, 14(2-4), pp. 323-357.

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

[23] Saccà, D. (1997), The Expressive Powers of Stable Models for Bound and Unbound Queries. *Journal of Computer and System Science*, 54(3), pp. 441-464.

[24] Saccà, D., and Zaniolo, C. (1990), Stable Models and Non-Determinism in Logic Programs with Negation. *Proc. ACM Symposyum on Principles of Database Systems*, pp. 205-218.

[25] Saccà, D., and Zaniolo, C. (1997), Deterministic and Non-Deterministic Stable Models. *Journal of Logic and Computation*, 7(5), pp. 555-579.

[26] Schlipf, J. S. (1990), The Expressive Powers of the Logic Programming Semantics. *Proc. ACM Symposyum on Principles of Database Systems*, pp. 196-204.

[27] Smullyan, R. M. (1978), *What is the name of this book?: The Riddle of Dracula and Other Logical Puzzles*. Prentice Hall.

[28] Ullman, J. D. (1988), *Principles of Database and Knowledge Base Systems*, Computer Science Press.

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

[30] You J. H., and Yuan, L. (1995), On the Equivalence of Semantics for Normal Logic Programming. *Journal of Logic Programming*, 22(3), pp. 211-222.