Expressive Power and Complexity of Partial Models for Disjunctive Deductive Databases [1]

Thomas Eiter [a,2]

[a] *AG Informatik, University of Gießen, Arndtstraße 2, D-35392 Gießen, Germany. Email: eiter@informatik.uni-giessen.de*

Nicola Leone [b,3]

[b] *Information Systems Department, Technical University of Vienna, Paniglgasse 16, A-1040 Vienna, Austria. Email: leone@dbai.tuwien.ac.at*

Domenico Saccá [c]

[c] *DEIS, University of Calabria, I-87030 Rende, Italy. Email: sacca@si.deis.unical.it*

Preprint submitted to Elsevier Preprint *8 February 1999*

This paper investigates the expressive power and complexity of partial model semantics for disjunctive deductive databases. In particular, partial stable, regular model, maximal stable (M-stable), and least undefined stable (L-stable) semantics for function-free disjunctive logic programs are considered, for which the expressiveness of queries based on possibility and certainty inference is determined. On the complexity side, we determine the data and expression complexity of query evaluation. The analysis pays particular attention to the impact of syntactical restrictions on programs in the form of limited use of disjunction and negation.

It appears that the considered semantics capture complexity classes at the lower end of the polynomial hierarchy. In fact, each class Σ_i^P, Π_i^P $1 \leq i \leq 3$ is captured by some semantics using appropriate syntactical restrictions. Partial stable models have exactly the same expressive power and complexity as total stable models (Σ_2^P resp. Π_2^P), while a higher degree of expressiveness is obtained by the semantics which minimize undefinedness (M-stable, regular, and L-stable semantics). In particular, L-stable semantics has the highest expressive power (Σ_3^P resp. Π_3^P). An interesting result in this course is that, in contrast with total stable models, negation is for partial stable models more expressive than disjunction.

For the data complexity of queries, we obtain completeness results for the classes Σ_i^P, Π_i^P, $i \leq 3$, and, for the expression complexity, completeness results for the analogous classes at the lower end of the weak exponential hierarchy.

The results of this paper complement and extend previous results, and contribute to a more complete picture of the computational aspects of disjunctive logic programming and databases, which supports in choosing an appropriate setting that fits the needs in practice.

1 Introduction

The study of integrating databases with logic programming opened in the past the research field of *deductive databases.* Basically, a deductive database is a logic program without function symbols, i.e., a datalog program (extended with negation) [55,14]. A number of advanced deductive database systems have been developed that utilize logic programming and extensions thereof for querying relational databases [13,15,36,43,45].

The need for representing disjunctive (or incomplete) information led to *disjunctive deductive databases* [40], for which a generalization of the closed world assumption (CWA) had to be devised, whose complexity has been first analyzed in [17]. Disjunctive deductive databases can be basically seen as disjunctive logic programs without function symbols, i.e., *disjunctive datalog (DATALOG*$^{\vee,\neg}$*) programs* (simply programs in the following) [26].

Several alternative semantics for (disjunctive) programs based on total models have been proposed, e.g. [28,40,44,51] (see [3,19,38] for comprehensive surveys). A widely accepted semantics is the extension of the stable model semantics [27] to disjunctive programs [28,44]. This semantics coincides with the minimal model semantics [40] on negation-free ($\neg$-free) programs and with the perfect model semantics [44] on stratified programs. Stable model semantics for disjunctive program has quite high expressive power, as it captures the complexity class Σ_2^P (i.e., it allows to express *all* (and only) database properties that are decidable in non-deterministic polynomial time with an oracle in NP) [29,22].

Despite its relevance, a severe drawback of total stable semantics is that it does not assign a model to each program. In particular, meaningful programs may have no total stable model. To overcome this drawback, a number of partial model semantics have been recently proposed, which relax the notion of total stable model and assign a meaning to a wider class of programs [5,23,44,49,48,58,59]. In a sense, these partial models "approximate" total stable models.

The first relaxation of total stable was the notion of *partial stable model* (also

[1] An abstract of this paper has been presented at the Workshop on Logic in Databases (LID '96), San Miniato, Italy, July 1996.

[2] Partially supported by Christian Doppler Laboratory for Expert Systems, Technical University of Vienna, Paniglgasse 16, A-1040 Vienna, Austria.

[3] Partially supported by *FWF* (Austrian Science Funds) under the *project P11580-MAT "A Query System for Disjunctive Deductive Databases"*, *Istituto per la Sistemistica e l'Informatica ISI-CNR*, and *Christian Doppler Laboratory for Expert Systems.*

called *3-valued stable* or *P-stable* model) proposed by Przymusinski in [44]. Compared to total stable models, partial stable models conservatively extend the class of programs for which an acceptable model exists; in particular, every disjunction-free ($\vee$-free) program has some partial stable model, while it may lack a total stable model. Moreover, on the class of (disjunctive) stratified programs, partial stable models coincide with total stable models.

Objections to partial stable models came from the observation that every 3-valued model theoretic approach should meet the principle of *minimal undefinedness* [58,59], which prescribes that the *undefined* truth value should be used only when necessary (i.e., a "good" semantics should tend to minimize the set of undefined atoms). The attempt to minimize undefinedness in partial models led to three main notions of partial models: maximal stable, regular, and least undefined stable models.

The *maximal stable* (*M-stable*) models [23,50,47,48] are those partial stable models which are maximal under set inclusion (where a partial model is represented by the set of ground literals true in the model). On disjunction-free programs, M-stable models coincide with the preferred extensions of [20], the regular models of [58], the maximal stable classes of [5], and the M-stable models of [50,48,47], as shown in [24,23,60].

The notion of *regular model* [58,59] is similar in spirit to M-stable model, but is based on a weaker concept of model than partial stability, which has the advantage that every program admits a regular model. On the other hand, as discussed in [23], a drawback of regular models is that they do not obey to the CWA principle.

The *least undefined stable* (*L-stable*) models [23,50,47] are the partial stable models with the minimal degree of undefinedness, i.e., no other partial stable model exists whose undefined atoms constitute a proper subset of the atoms that are undefined in an L-stable model. The relevance of L-stable models is confirmed by the fact that L-stable models differ from total stable models *only if* the program has no total stable model; thus, L-stable models can be considered as the best "approximation" of total stable models.

In this paper, we study the expressive power and the complexity of disjunctive (datalog) programs based on the partial model semantics mentioned above, i.e., the capability of this query language of expressing queries on relational databases (see [2] for background on the subject). In particular, we analyze the expressive power of bound (Boolean) queries resorting to the common modalities of *possibility inference* –a literal is true if it is in some model– and *certainty inference* –a literal is true if it is in every model [1]. (As in many cases, results for general database queries can be easily derived from these results.) The main points of interest are:

– The expressive powers of the different partial model semantics vs. total model semantics, and their complexity.
– The impact of minimizing undefinedness on expressive power and complexity.
– The impact of syntactical restrictions. In particular, the power of disjunction vs the power of negation; the effect of stratified negation vs arbitrary negation; and, the effect of limited disjunction (in particular, *head cycle free* disjunction [6]) vs unrestricted disjunction.

The main results on the expressiveness can be summarized as follows:

(i) Partial stable models have the same expressive power as total stable models. Under possibility and certainty inference, they capture the complexity classes Σ_2^P and Π_2^P, i.e., they can express precisely the database collections with complexity in Σ_2^P and Π_2^P, respectively (see Section 3 for details).
(ii) Under certainty inference, M-stable models and regular models are more powerful than the total stable models, as the former capture the class Π_3^P while the latter capture Π_2^P; under possibility inference, both have the same power as total stable models and capture the class Σ_2^P.
(iii) L-stable models are always more expressive than total stable models, since, under possibility and certainty inference, they capture the classes Σ_3^P and Π_3^P, respectively.
(iv) For partial models that minimize undefinedness (M-stable, L-stable, and regular models), negation is more expressive than disjunction under certainty inference and for L-stable models, also under possibility inference. Indeed, $\vee$-free programs with negation can express all of Σ_2^P (or Π_2^P); however, only a fragment thereof can be expressed by $\neg$-free programs with disjunction. By contrast, under total model semantics $\vee$-free programs with negation can define only database collections in NP (or coNP), while $\neg$-free programs with disjunction can define database collections that are Σ_2^P-hard (or Π_2^P-hard) to recognize.
(v) Allowing headcycle-free disjunction (on $\neg$-free programs) increases the expressibility to a strict fragment of NP (resp. coNP), which contrasts with the well-known fact that stratified negation (on $\vee$-free programs) does not increase the expressibility beyond polynomial time computability. Interestingly, the combination of headcycle-free disjunction and stratified negation captures NP (or coNP), for all variants of partial models that we consider.

On the complexity side, we determine the data and expression complexity [57] of query evaluation. The data complexities (i.e., evaluating an arbitrary but fixed query over a given database) of the various query languages we consider yield complete problems for the classes at the lower end of the polynomial hierarchy up to Σ_3^P and Π_3^P. All query languages considered are shown to have data complexity which is complete for a class C from the polynomial hierarchy (however, not every considered query language which is complete for C can also express all of C).

The expression complexity (i.e., evaluating a given query on an arbitrary but fixed database) parallels the data complexity in the weak exponential hierarchy NEXP $= \Sigma_1^E$, $\Sigma_2^E =$ NEXP$^{\mathrm{NP}}$, $\Sigma_3^E =$ NEXP$^{\Sigma_2^P}$, ... and yields complete problems for these classes.

There are many papers in the literature concerned with the complexity and expressive power of various semantics for datalog, see e.g. [10,39,52,35,12,21,22,47,48]. Our paper complements and extends the results in [22] (which considers disjunctive datalog but with *total* models only) and those in [47] (which takes into account partial models but only for classical datalog *without disjunction*). Indeed, our paper integrates the two analysis inasmuch as it analyzes the expressive power of several types of partial stable models for disjunctive datalog. In addition, it covers the regular semantics [58,59], whose computational properties have not been studied previously.

Furthermore, our paper investigates the different sources of complexity and expressiveness (in particular, restricted forms of negation and of disjunction), and provides in this line an in-depth picture of the complexity and expressiveness for various fragments of disjunctive datalog, ranging on the classes in the lower end of the polynomial hierarchy, from P to the third level. It is worth noting that the languages analyzed in [22,47] do not exceed the second level of the hierarchy.

The analysis of the impact of syntactic restrictions carried out in this paper adds new insights to previous studies on complexity and expressiveness of restricted forms of disjunction (*headcycle-free* programs) [6–8] and negation (*stratified negation*) [33], and contrasts the relative effect of those restrictions. The results may support in choosing an appropriate fragment of the query language under a suitable semantics, which fits the needs in practice.

Note that for normal programs the well-founded semantics [56] is often viewed as a competing approach to the stable model semantics in [27], and is understood as an approximation thereof. In fact, the well-founded model of a normal program is a P-stable model, and, moreover, the unique minimal P-stable model [49]. Thus, well-founded semantics is intimately related to P-stable semantics on normal programs. This is not the case with disjunctive programs, however. In fact, while several attempts to extend the well-founded semantics to disjunctive programs have been made, e.g. [4,46], there is to date no general consensus about a proper formulation. Hence, we do not address such approaches here. Furthermore, we stress that, for the sake of presentation, our analysis is based on boolean queries only; non-boolean queries can be treated similar as in [22,47,48].

The remainder of the paper is structured as follows. Section 2 gives a short review of the definition of partial model semantics for disjunctive logic programs, and points out some basic relationships between different concepts of models. In Section 3, we first describe the formal framework for the use of function-free disjunctive logic programs (i.e., $DATALOG^{\vee,\neg}$ programs) for defining queries and recall the basic concepts about expressive power of a query language; after that, we determine the expressive power of the different $DATALOG^{\vee,\neg}$ query languages for each of the partial model semantics in Section 2. In Section 4, we address the issue of query complexity, and determine the data and expression complexity [57] of the $DATALOG^{\vee,\neg}$ query languages. Section 5 is dedicated to the effect of syntactical restrictions on query programs, and discusses the expressive power under limited use of disjunction and negation. Finally, Section 6 gives a short summary and concludes the paper.

2 Partial Model Semantics

In this section we review from [23,44,59], the basic definitions and characterizations of partial model semantics for disjunctive logic programs. We assume that the reader is familiar with the basic concepts of logic programming [37]. For a more detailed account of (disjunctive) logic programming and databases, see [14,38,41,2]; for overviews of expressibility and complexity results for normal and disjunctive logic programming, see [29,11,53].

2.1 Disjunctive Logic Programs

A *rule* r is a clause of the form

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

$a_1, \cdots, a_n, b_1, \cdots, b_m$ are atoms of the form $p(t_1, ..., t_n)$, where p is a *predicate* of arity n and the terms $t_1, ..., t_n$ are constants or variables. The disjunction $a_1 \vee \cdots \vee a_n$ is the *head* of r, while the conjunction $b_1, ..., b_k, \neg b_{k+1}, ..., \neg b_m$ is the *body* of r. We denote by $H(r)$ the set $\{a_1, ..., a_n\}$ of the head atoms, and by $B(r)$ the set $\{b_1, ..., b_k, \neg b_{k+1}, ..., \neg b_m\}$ of the body literals. (As usual, a literal is an atom p or a negated atom $\neg p$; in the former case, it is positive, and in the latter negative.) Moreover, $B^+(r)$ and $B^-(r)$ denote the set of positive and negative literals occurring in $B(r)$, respectively.

A *disjunctive logic program* (simply *program* hereafter) 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.

We often use upper-case letters, say L, to denote literals. Two literals are *complementary* , if they are of the form p and $\neg p$, for some atom p. For a literal L, $\neg L$ denotes its complementary literal, and for a set A of literals, $\neg.A = \{\neg L \mid L \in A\}$.

The *Herbrand Universe* $U_{\mathcal{LP}}$ of a program $\mathcal{LP}$ is the set of all constants appearing in $\mathcal{LP}$, and its *Herbrand Base* $B_{\mathcal{LP}}$ is the set of all ground atoms constructed from the predicates appearing in $\mathcal{LP}$ and the constants from $U_{\mathcal{LP}}$. A rule r' is a *ground instance* of a rule r, if r' is obtained from r by replacing every variable X with $\sigma(X)$, where σ is a map from the variables occurring in r into $U_{\mathcal{LP}}$. We denote by $ground(\mathcal{LP})$ the set of all ground instances of the rules in $\mathcal{LP}$.

An interpretation of $\mathcal{LP}$ is a consistent set of ground literals, i.e., a subset $I \subseteq B_{\mathcal{LP}} \cup \neg.B_{\mathcal{LP}}$ such that $I \cap \neg.I = \emptyset$. We denote by I^+ and I^- the set of positive and negative literals occurring in I, respectively, and by $\overline{I}$ the set $B_{\mathcal{LP}} - (I \cup \neg.I)$. The interpretation I is *total* if $\overline{I}$ is empty; otherwise, I is *partial*.

Following [44], we define a 3-valued logic with values T *(True)*, F *(False)*, and U *(Undefined)*, ordered by $F < U < T$. Let $\mathcal{LP}$ be a program and I an interpretation. The value of a ground literal L in I, $value_I(L)$, is T if $L \in I$, F if $L \in \neg.I$, and U otherwise (i.e., if $L \notin I \cup \neg.I$). For ground literals $L_1, \ldots, L_n$, the value of the conjunction C of all L_i 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 the disjunction D of all L_i is their maximum, i.e., $value_I(D) = max(\{value_I(L_i) \mid 1 \leq i \leq n\})$; if $n = 0$, then $value_I(C) = T$ and $value_I(D) = F$. Finally, a ground rule r is *satisfied* by I if $value_I(H(r)) \geq value_I(B(r))$.

An interpretation M for $\mathcal{LP}$ is a *3-valued* model of $\mathcal{LP}$ if M satisfies each rule in $ground(\mathcal{LP})$; if M is total, it is a *total model.* A model $M_1 \neq M$ is *smaller than* M, denoted $M_1 \prec M$, iff $M_1^+ \subseteq M^+$ and $M_1^- \supseteq M^-$. A model M is *minimal* iff there does not exist a smaller model M_1. Among total models, a model M is *minimal* iff there exists no smaller total model M_1. We denote the set of the minimal total models by $\mathrm{MM}^+(\mathcal{LP})$.

2.2 P-stable Models

As for total stable models [27,44], 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.

An interpretation M for $\mathcal{LP}$ is *justified* (or *founded* [23]) iff $M^+ = N^+$ for some $N \in \mathrm{MM}^+(\mathcal{LP}(M))$, where the program $\mathcal{LP}(M)$ is obtained from $ground(\mathcal{LP})$ as follows: (i) remove all rules r such that $B^-(r) \not\subseteq I^-$, and (ii) remove all negative literals from the remaining rules.

Example 1 Let $\mathcal{LP}_1$ be the following program:

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

Interpretation $M_1 = \{a,\ \neg b,\ \neg c\}$ is founded. Indeed, $\mathcal{LP}_1(M_1)$ is the following program:

$$a \vee b \leftarrow$$

and there exists a minimal total model N of $\mathcal{LP}_1(M_1)$ such that $N^+ = M_1^+$, namely $N = \{a,\ \neg b\}$. For $M_2 = \{b,\ c,\ \neg a,\ \neg d\}$, $\mathcal{LP}_1(M_2)$ is as follows:

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

Also M_2 is founded, since $N = \{b, c, \neg a\}$ is a minimal total model of $\mathcal{LP}_1(M_2)$ and $N^+ = M_2^+$. $\Box$

For total interpretations foundedness alone is sufficient to characterize (2-valued) stable models, which are indeed defined as the total founded interpretations of the program [27,28,44]. However, as for normal programs [49], foundedness 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. This consistency with the CWA is precisely formalized by the notion of maximal unfounded set.

Definition 2 [23] Let I be an interpretation for $\mathcal{LP}$. A set $X \subseteq B_{\mathcal{LP}}$ of ground atoms is an *unfounded set* for $\mathcal{LP}$ w.r.t. I if, for each $a \in X$, at least one of the following conditions holds for each rule $r \in ground(\mathcal{LP})$ such that

$a \in H(r)$:

(i) $B(r) \cap (\neg.I \cup X) \neq \emptyset$, or
(ii) $H(r) \not\subseteq (\neg.I \cup X)$.

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

Thus, if X is a consistent unfounded set (i.e., $I \cup \neg.X$ is an interpretation), for every rule r having an element of X in the head, either the body of r is false w.r.t. $I \cup \neg.X$ (Condition 1 of Definition 2), or the head of r is not false w.r.t. $I \cup \neg.X$ (Condition 2 of Definition 2). Thus, if we think in terms of classical (2-valued) interpretations, it is possible to extend I by assuming that all atoms in X are false,[4] without explicitly violating r; in fact, rule r is still satisfiable: if Condition 1 applies, then r is definitely satisfied (as the body is false), and if Condition 2 applies, r can be satisfied by assuming true some atom in $H(r) - X$. Since the atoms in X are "defined" only by such rules r (under the intended assignment-based reading of rules), intuitively every atom in X should be false, since the assumption that it is true is not much founded, according to a principle of closed world assumption.

Example 3 Consider the program $\mathcal{LP} = \{a \vee b \leftarrow\}$ and the interpretation $I = \emptyset$. The set $X = \{a\}$ is a (consistent) unfounded set of $\mathcal{LP}$ w.r.t. I. Indeed, the unique rule of $ground(\mathcal{LP})$ (with a in the head) satisfies Condition 2 of Definition 2, as $H(r) - (X \cup \neg.I) = \{b\}$. Likewise, we can easily verify that $Y = \{b\}$ is a (consistent) unfounded set for $\mathcal{LP}$ w.r.t. I (as $H(r) - (Y \cup \neg.I) = \{a\}$).
For the program $\mathcal{LP} = \{\ a \vee b \leftarrow\ ;\ a \leftarrow b\ ;\ b \leftarrow a\ \}$ and the interpretation $I = \{a, b\}$ the only unfounded set is $\emptyset$. $\Box$

Even if Definition 2 generalizes the standard definition of unfounded set [23] for normal programs, the union of all unfounded sets is not necessarily an unfounded set in case of a disjunctive program. Thus, a notion of maximal unfounded set is given instead of that of greatest unfounded set.

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

Example 4 Let $\mathcal{LP} = \{a \vee b \leftarrow\}$. Both $X_1 = \{a\}$ and $X_2 = \{b\}$ are maximal unfounded sets for $\mathcal{LP}$ w.r.t. $I = \emptyset$.

X_1 is the only maximal unfounded set for $\mathcal{LP}$ w.r.t. the interpretation $\{b\}$ (note that X_2 is not a consistent unfounded set w.r.t. $\{b\}$). For program $\mathcal{LP}_1$ of Example 1, the interpretation $M_1 = \{a,\ \neg b,\ \neg c\}$ has the single maximal

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

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

Intuitively, a maximal unfounded set X is a maximal set of atoms that can be assumed false without violating the satisfiability of the rules of the program that contain some atom from X. Consistency of the unfounded set X ensures that the assumption of the falsity of the literals in X is consistent with I (that is, $I \cup \neg.X$ is an interpretation). For instance, we have seen in the above examples that for the program $\mathcal{LP} = \{a \vee b \leftarrow\}$ both $\{a\}$ and $\{b\}$ are (consistent) unfounded sets w.r.t. $I = \emptyset$ while $\{a, b\}$ is not an unfounded set. Indeed, we can assume the falsity of either a or b without violating the satisfiability of $a \vee b \leftarrow$ (which can be satisfied by making true b or a, respectively); but we cannot assume the falsity of both a and b at the same time, as in the latter case we would have no way to satisfy rule $a \vee b \leftarrow$.

We are finally ready to provide the notion of P-stable model.

Definition 5 [23] Let $\mathcal{LP}$ be a program and M an interpretation of it. Then M is a *P-stable* (*partial stable*) model of $\mathcal{LP}$ if the following conditions hold:

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

Note that Condition (a) characterizes the true atoms of M, imposing that they have to be "justified" [58], while Condition (b) characterizes the false atoms of M, requiring that they must be all and only the atoms that can be assumed false consistently with M and with the rules of the program (further negative assumptions would make some rule unsatisfiable).

Example 6 Consider the program $\mathcal{LP}_1$ of Example 1. Both $M_1 = \{a,\ \neg b, \neg c\}$ and $M_2 = \{b,\ c,\ \neg a,\ \neg d\}$ are P-stable models of $\mathcal{LP}_1$. Indeed, we have seen in Example 1 that M_1 and M_2 are founded, and Example 4 showed that $\neg.M_1^-$ and $\neg.M_2^-$ are maximal unfounded sets w.r.t. M_1 and M_2, respectively.

Notice that M_1 and M_2 are the only P-stable models of $\mathcal{LP}_1$. For instance, the interpretation $M_3 = \{a, \neg d\}$ is not a P-stable model of $\mathcal{LP}_1$. While M_3 is founded (note that $\mathcal{LP}_1(M_3) = \{a \vee b \leftarrow\ ,\ a \vee c \leftarrow\ \}$), $\neg.M_3^-$ is not a maximal unfounded set w.r.t. M_3; indeed, M_3 has the single maximal unfounded set $X = \{b, c\}$. $\Box$

We gave the definition of P-stable in terms classical 2-valued semantics and unfounded sets, as we use this formulation to derive complexity and expressibility results in next sections; however, as shown in [23], this definition is equivalent to the one provided by Przymusinski in terms of 3-valued logics.

Proposition 7 [23] *M is a P-stable model for a program $\mathcal{LP}$ if and only if*

M is a 3-valued stable model for $\mathcal{LP}$ according with [44].

It is worth noting that a disjunctive program may lack P-stable models, while every normal program has some P-stable model [44].

2.3 Restricted Classes of P-stable Models

As for normal ($\vee$-free) logic programs [50,47,48], P-stable models are grouped into three main families: T-stable, M-stable, and L-stable models.

Definition 8 [23] 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 $\mathcal{LP}$ 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 $\mathcal{LP}$ such that $\overline{N} \supset \overline{M}$. $\Box$

Note that the T-stable models coincide with the 2-valued (total disjunctive) stable models of [44].

Example 9 The M-stable models of the program $\mathcal{LP}_1$ in Example 1 are $M_1 = \{a,\ \neg b, \neg c\}$ and $M_2 = \{b,\ c,\ \neg a,\ \neg d\}$. In particular, M_2 is also L-stable and T-stable; while M_1 is not L-stable (and it is not T-stable obviously).

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

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

The P-stable models of $\mathcal{LP}_2$ are:

$$\begin{aligned} M_1 &= \{b,\ \neg a,\ \neg c,\ \neg d\}, \\ M_2 &= \{a,\ d,\ \neg b,\ \neg c\}, \\ M_3 &= \{\neg c\} \end{aligned}$$

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 $\mathcal{LP}_2$. 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. Consider the program $\mathcal{LP}_3$:

$$a \vee b \leftarrow$$

$$c \leftarrow \neg c$$

Both $\{a\}$ and $\{b\}$ are L-stable models; but neither the former nor the latter is T-stable. $\Box$

2.4 Regular Models

Regular models are similar in spirit to M-stable models, as they are maximal elements under set inclusion in a class of models. However, they are based on a weaker concept of model than the P-stable model which underlies M-stability.

Definition 10 [59] Let $\mathcal{LP}$ be a logic program and M an interpretation of it. Then, M is a regular model of $\mathcal{LP}$ iff

(a) M is a founded 3-valued model of $\mathcal{LP}$, and
(b) M is maximal w.r.t. inclusion, i.e., there exists no founded 3-valued model N of $\mathcal{LP}$ with $N \supset M$.

Notice that every program has a regular model [59]. Thus, in contrast to the semantics from above, the regular model semantics assigns a model to every program.

Example 11 [44] Let $\mathcal{LP}$ be the program below.

$$work \vee sleep \vee tired \leftarrow$$

$$work \leftarrow \neg tired$$

$$sleep \leftarrow \neg work$$

$$tired \leftarrow \neg sleep$$

It is easy to see that $\mathcal{LP}$ has no P-stable model (cf. [44]), and hence no M-stable, L-stable, and T-stable model. On the other hand, $\{work\}$, $\{sleep\}$, and $\{tired\}$ are (all the) regular models of $\mathcal{LP}$. $\Box$

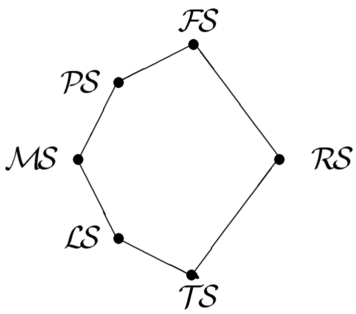

Fig. 1. Relation between partial model semantics

2.5 Relationships between Models

We briefly discuss some properties and relationships of the concepts of partial models from above; a more detailed account is given in [23]. We use $\mathcal{XS}_{\mathcal{LP}}$ as generic notation for the models of $\mathcal{LP}$, where $\mathcal{XS}_{\mathcal{LP}}$ is one of $\mathcal{PS}_{\mathcal{LP}}$, $\mathcal{MS}_{\mathcal{LP}}$, $\mathcal{LS}_{\mathcal{LP}}$, $\mathcal{TS}_{\mathcal{LP}}$, and $\mathcal{RS}_{\mathcal{LP}}$, and denotes the sets of P-stable, M-stable, L-stable, T-stable, and regular models of $\mathcal{LP}$, respectively. Whenever $\mathcal{LP}$ is understood from the context, we shall omit the subscript $\mathcal{LP}$.

The inclusion order diagram for the different sets of models $\mathcal{XS}$ is displayed in Figure 1. There, $\mathcal{FS}$ denotes the set of founded 3-valued models of $\mathcal{LP}$.

In general, all concepts of models are different. However, on the class of (even locally) stratified programs, with the exception of $\mathcal{FS}$ they all coincide. [5]

This is an immediate consequence of the fact that the P-stable models and T-stable models are the 3-valued stable and 2-valued stable models, which coincide for locally stratified DLPs [44].

Proposition 12 [23] *Let $\mathcal{LP}$ be a program. If $\mathcal{LP}$ is (locally) stratified, then* $\mathcal{PS}_{\mathcal{LP}} = \mathcal{MS}_{\mathcal{LP}} = \mathcal{LS}_{\mathcal{LP}} = \mathcal{TS}_{\mathcal{LP}} = \mathcal{RS}_{\mathcal{LP}}$.

Note that a (locally) stratified program has always P-stable models.

In addition to foundedness, P-stable models obey the principle of the closed world assumption (CWA), adapted to models of disjunctive logic programs [23]: "If every ground rule with an atom A in the head has a false body, or its head contains a true atom distinct from A w.r.t. an acceptable model, then A

[5] A program $\mathcal{LP}$ is *(locally) stratified* if each atom $p \in B_{\mathcal{LP}}$ can be associated a positive integer $l(p)$ such that, for each rule $r \in ground(\mathcal{LP})$ with $p \in H(r)$ and each ground atom q the following holds: (i) if $q \in H(r)$, then $l(p) = l(q)$; (ii) if $\neg q \in B^-(r)$, then $l(p) > l(q)$; (iii) if $q \in B^+(r)$, then $l(p) \geq l(q)$. In particular, if all ground atoms with the same predicate have the same number associated, then $\mathcal{LP}$ is globally stratified.

must be false in that model."

The concept of regular model is similar to M-stable model, but yet different; this is explained by lacking obedience to the CWA principle. However, if disjunction is not present, the concepts coincide.

Proposition 13 [60] *Let $\mathcal{LP}$ be a normal program. Then, $\mathcal{MS} = \mathcal{RS}$.*

$\mathcal{MS}$ and $\mathcal{LS}$ on the one hand and $\mathcal{RS}$ on the other provide approximations of $\mathcal{TS}$. In fact, $\mathcal{LS}$ is accurate for the programs that are consistent under stable semantics.

Proposition 14 [23] *Let $\mathcal{LP}$ be a program. Then, $\mathcal{TS} \neq \emptyset$ implies $\mathcal{TS} = \mathcal{LS}$.*

2.6 Deterministic Semantics

Each set $\mathcal{XS}$ of models can be considered as the *intended models* of a program $\mathcal{LP}$. Although $\mathcal{XS}$ contains multiple models, a deterministic semantics can be enforced using two main approaches cf. [1,47,53,22]: the *possibility* (or *credulous*, *brave*) semantics, and the *certainty* (or *skeptical*, *cautious*) semantics.

Definition 15 Let $\mathcal{LP}$ be a program and A be a ground literal. Then

(a) A is a $\exists_{\mathcal{XS}}$ (*possible*) inference of $\mathcal{LP}$, denoted $\mathcal{LP} \models^{\exists}_{\mathcal{XS}} A$, if $\exists M \in \mathcal{XS}$ such that $A \in M$ (i.e., A is true in some model in $\mathcal{XS}$);
(b) A is a $\forall_{\mathcal{XS}}$ (*certain*) inference of $\mathcal{LP}$, denoted $\mathcal{LP} \models^{\forall}_{\mathcal{XS}} A$, if $\forall M \in \mathcal{XS}$, $A \in M$ (i.e., A is true in every model in $\mathcal{XS}$). □

The next proposition points out some relationships among the various stable semantics.

Proposition 16 *Let $\mathcal{LP}$ be a program and A be a ground literal. Then*

(a) *(A is a $\forall_{\mathcal{PS}}$ inference) $\Rightarrow$ (A is a $\forall_{\mathcal{MS}}$ inference) $\Rightarrow$ (A is a $\forall_{\mathcal{LS}}$ inference)*
(b) *(A is a $\exists_{\mathcal{LS}}$ inference) $\Rightarrow$ (A is a $\exists_{\mathcal{MS}}$ inference) $\equiv$ (A is a $\exists_{\mathcal{PS}}$ inference).*

PROOF. Point (a) along with the first implication of Point (b) follow from Proposition 4 of [23] where it is proven that $\mathcal{PS} \subseteq \mathcal{MS} \subseteq \mathcal{LS}$.

Equivalence between $\exists_{\mathcal{MS}}$ and $\exists_{\mathcal{PS}}$ inference stems from the observation that the union of all M-stable models coincides with the union of all P-stable models. □

Example 17 Consider program $\mathcal{LP}_1$ of Example 1.

The set $\{M_1, M_2\}$ of P-stable models of $\mathcal{LP}_1$ coincides with the set of the M-stable models. Thus, we have that $\{a, \neg b, \neg c, b, c, \neg a, \neg d\}$ are both $\exists_{\mathcal{PS}}$ and $\exists_{\mathcal{MS}}$ inferences; while both sets of $\forall_{\mathcal{PS}}$ and $\forall_{\mathcal{MS}}$ inferences are empty.

Model M_2 is the only L-stable and T-stable model of $\mathcal{LP}_1$. Therefore, b, c, $\neg a$, and $\neg d$ are $\forall_{\mathcal{LS}}$, $\forall_{\mathcal{TS}}$, $\exists_{\mathcal{LS}}$ and $\exists_{\mathcal{TS}}$ inferences.

On program $\mathcal{LP}_2$ of Example 9 the various stable semantics coincide. Indeed, under P-stable, M-stable, L-stable and T-stable semantics, $\neg c$ is the only certain inference, and a, $\neg a$, b, $\neg b$, $\neg c$, d, and $\neg d$ are the possible inferences.

Finally, consider the program of Example 11. Since $\mathcal{PS} = \mathcal{MS} = \mathcal{LS} = \mathcal{TS} = \emptyset$, we have that: (i) the set of $\exists_{\mathcal{PS}}$, $\exists_{\mathcal{MS}}$, $\exists_{\mathcal{LS}}$ and $\exists_{\mathcal{TS}}$ inferences are empty, and (ii) under $\forall_{\mathcal{LS}}$, $\forall_{\mathcal{TS}}$, $\exists_{\mathcal{LS}}$ and $\exists_{\mathcal{TS}}$, every literal is inferred.

Moreover, concerning regular model semantics, we have that *work*, *sleep*, and *tired* are $\exists_{\mathcal{RS}}$ inferences; while the set of $\forall_{\mathcal{RS}}$ inferences is empty. □

3 Expressive Power

3.1 Preliminaries

In the context of deductive databases, programs as introduced above are known as disjunctive datalog ($DATALOG^{\vee,\neg}$) programs, i.e., the extension of datalog (with negation) [14,32,55] by disjunction [22]. Some of the predicate symbols correspond to database relations (the *extensional (EDB) predicates*), and are not allowed to occur in rule heads; the other predicate symbols are called *intensional (IDB) predicates.* Actual database relations are formed on a fixed countable domain U, from which also possible constants in a $DATALOG^{\vee,\neg}$ program are taken.

More formally, a $DATALOG^{\vee,\neg}$ program $\mathcal{LP}$ has associated a relational database scheme $\mathcal{DB}_{\mathcal{LP}} = \{r \mid r \text{ is an EDB predicate symbol of } \mathcal{LP}\}$; thus EDB predicate symbols are seen as relation symbols. A database D on $\mathcal{DB}_{\mathcal{LP}}$ is a set of finite relations on U, one for each r in $\mathcal{DB}_{\mathcal{LP}}$, denoted by $D(r)$; note that D can be seen as a first-order structure whose universe consists of the constants occurring in D (the *active domain* of D). [6] The set of all databases on $\mathcal{DB}_{\mathcal{LP}}$ is denoted by $\mathbf{D}_{\mathcal{LP}}$.

[6] We use here active domain semantics (cf. [2]), rather then a setting in which a (finite) universe of D is explicitly provided [25,16,57]. Note that Fagin's Theorem and all other results to which we refer remain valid in this (narrower) context; conversely, the results of this paper can be extended to that setting.

Given a database $D \in \mathbf{D}_{\mathcal{LP}}$, $\mathcal{LP}_D$ denotes the following program:

$$\mathcal{LP}_D = \mathcal{LP} \cup \{r(t) \leftarrow \ | r \in \mathcal{DB}_{\mathcal{LP}} \wedge t \in D(r)\}.$$

Definition 18 A (*bound DATALOG*$^{\vee,\neg}$) *query* $\mathcal{Q}$ is a pair $\langle \mathcal{LP}, G\rangle$, where $\mathcal{LP}$ is a *DATALOG*$^{\vee,\neg}$ program and G is a ground literal (the *query goal*). Given a database D in $\mathbf{D}_{\mathcal{LP}}$ and a class of models $\mathcal{XS}$, the $\exists_{\mathcal{XS}}$ (resp., $\forall_{\mathcal{XS}}$) *answer* of $\mathcal{Q}$ on D is *true* if G is a $\exists_{\mathcal{XS}}$ (resp., $\forall_{\mathcal{XS}}$) inference of $\mathcal{LP}_D$ and *false* otherwise.

The set of all queries is denoted by $\mathbf{Q}$. $\Box$

The constants occurring in $\mathcal{LP}_D$, G define the active domain of query $\mathcal{Q} = \langle \mathcal{LP}, G\rangle$ on the database D.

Observe that, in general, two queries $\langle \mathcal{LP}, G\rangle$ and $\langle \mathcal{LP}, \neg G\rangle$ on the same database need not give symmetric answers. That is, if e.g. $\langle \mathcal{LP}, G\rangle$ answers yes for D, it may be possible that also $\langle \mathcal{LP}, \neg G\rangle$ answers yes for D.

A bound query paired with an answer (i.e., inference) mode defines a Boolean C-generic query of [2], i.e., a map from $\mathbf{D}_{\mathcal{LP}}$ to $\{true, false\}$. As common, we focus in our analysis of the expressive power of a query language on generic queries, which are those mappings whose result is invariant under renaming the constants in D with constants from U. Genericity of a bound query $\langle \mathcal{LP}, G\rangle$ is assured by excluding constants in $\mathcal{LP}$ and G. As discussed in [2, p. 421], this issue is not central, since constants can be provided by designated input relations; moreover, any query goal $G = (\neg)p(\cdots)$ can be easily replaced by a new goal $G' = (\neg)q$ and the rule $q \leftarrow p(\cdots)$, where q is a propositional letter. In the rest of this paper, we thus implicitly assume that constants do not occur in queries.

Definition 19 Let $\mathcal{Q} = \langle \mathcal{LP}, G\rangle$ be a (constant-free) query. Then the *database collection* of $\mathcal{Q}$ w.r.t. the set of models $\mathcal{XS}$ is:

(a) *under the possibility ($\exists_{\mathcal{XS}}$) variant of semantics*, the set of all databases D in $\mathbf{D}_{\mathcal{LP}}$ for which the $\exists_{\mathcal{XS}}$-answer of $\mathcal{Q}$ is true, which is denoted by $\mathcal{EXP}^{\exists}_{\mathcal{XS}}(\mathcal{Q})$;
(b) *under the certainty ($\forall_{\mathcal{XS}}$) variant of semantics*, the set of all databases D in $\mathbf{D}_{\mathcal{LP}}$ for which the $\forall_{\mathcal{XS}}$-answer of $\mathcal{Q}$ is true, which is denoted by $\mathcal{EXP}^{\forall}_{\mathcal{XS}}(\mathcal{Q})$.

The *expressive power* of $\mathbf{Q}$ under a type of semantics w.r.t $\mathcal{XS}$ is the family of the database collections of all queries $\mathcal{Q}$, i.e., $\mathcal{EXP}^{\exists}_{\mathcal{XS}}[\mathbf{Q}] = \{\mathcal{EXP}^{\exists}_{\mathcal{XS}}(\mathcal{Q}) | \mathcal{Q} \in \mathbf{Q}, \mathcal{Q}$ constant-free$\}$ and $\mathcal{EXP}^{\forall}_{\mathcal{XS}}[\mathbf{Q}] = \{\mathcal{EXP}^{\forall}_{\mathcal{XS}}(\mathcal{Q}) | \mathcal{Q} \in \mathbf{Q}, \mathcal{Q}$ constant-free$\}$. $\Box$

The expressive power of each semantics will be related to database complexity classes, which are as follows. Let C be a Turing machine complexity class (e.g., P or NP), $\mathbf{R}$ be a relational database scheme, and $\mathbf{D}$ be a set of databases on

R.[7] Then, **D** is *C-recognizable* if the problem of deciding whether $D \in \mathbf{D}$ for a given database D on **R** is in C. The *database complexity class DB-C* is the family of all C-recognizable database collections. (For instance, *DB*-P is the family of all database collections that are recognizable in polynomial time). If the expressive power of a given semantics coincides with some class *DB-C*, we say that the given semantics *captures C*.

Recall that the classes Σ_k^P, Π_k^P of the polynomial hierarchy [54] are defined by $\Sigma_0^P = \mathrm{P}$, $\Sigma_{i+1}^P = \mathrm{NP}^{\Sigma_i^P}$, and $\Pi_i^P = \text{co-}\Sigma_i^P$, for all $i \geq 0$. In particular, $\Pi_0^P = \mathrm{P}$, $\Sigma_1^P = \mathrm{NP}$, and $\Pi_1^P = \mathrm{coNP}$. By Fagin's Theorem [25] and its generalization in [54], complexity and second-order definability are linked as follows.

Proposition 20 ([25,54]) *A database collection* **D** *over a scheme* **R** *is in DB-*Σ_k^P*,* $k \geq 1$*, iff it is definable by a second-order formula* $(\exists \mathbf{A}_1)(\forall \mathbf{A}_2)\cdots(Q_k\mathbf{A}_k)\phi$ *on* **R***, where the* $\mathbf{A}_i$ *are lists of predicate variables preceded by alternating quantifiers and* ϕ *is first-order.*

3.2 The Power of $\mathcal{PS}$ Semantics

In [21], it was shown that $\exists_{\mathcal{PS}}$ and $\forall_{\mathcal{PS}}$ inference from propositional programs is complete for the complexity classes Σ_2^P and Π_2^P, respectively. In this section, we complement these complexity results by capturing results for Σ_2^P and Π_2^P.

Lemma 21 *Let* $\mathcal{LP}$ *be a ground program, and let* M *be an interpretation. Deciding whether* $M \in \mathcal{PS}$ *is* coNP*-complete.*

PROOF. To show that M is a P-stable model, we have to show that (a) M is founded and (b) $\neg.M^-$ is a maximal unfounded set. Concerning (a), deciding if $M^+ = N^+$ for some $N \in MM^+(\mathcal{LP}(M))$ is in coNP, since the program $\mathcal{LP}(M)$ can be constructed efficiently and a guess for a total model $M1$ of $\mathcal{LP}(M)$ such that $M1 \prec M$ (where $M1$ is represented by its positive part $M1^+$) can be checked in polynomial time. Also (b) is in coNP: checking if $\neg.M^-$ is an unfounded set is easy, and if all ground atoms that do not occur in $\mathcal{LP}$ are false in M (otherwise $\neg.M^-$ is clearly not maximal) a guess for a larger consistent unfounded set has polynomial size and can be checked in polynomial time. Since coNP is closed under conjunctions, it follows that deciding P-stability of M is in coNP. For a proof of hardness where all atoms are propositional letters –which uses the equivalent definition of 3-valued stable model (Prop. 7)– cf. [21]. $\Box$

[7] As usual, adopting the data independence principle, it is assumed that **D** generic, i.e., it is closed under renamings of the constants in U.

Lemma 22 [22] *Every DB-Σ_2^P-recognizable database collection* $\mathbf{D}$ *can be defined by a (constant-free) query* $Q = \langle \mathcal{LP}, G \rangle$ *under* $\exists_{\mathcal{TS}}$ *semantics, where* $\mathcal{LP}$ *is stratified and* G *is an atom.*

By the coincidence of T-stable and P-stable models on stratified programs, we thus obtain the following results.

Theorem 23 $\mathcal{EXP}^{\exists}_{\mathcal{PS}}[\mathbf{Q}] = DB\text{-}\Sigma_2^P$.

PROOF. We first prove that for any query $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ in $\mathbf{Q}$, recognizing whether a database D is in $\mathcal{EXP}^{\exists}_{\mathcal{PS}}(\mathcal{Q})$ is in Σ_2^P. D is in $\mathcal{EXP}^{\exists}_{\mathcal{PS}}(\mathcal{Q})$ iff there exists a P-stable model M of $\mathcal{LP}_D$ such that $G \in M$. To check this, we may guess an interpretation M of $\mathcal{LP}_D$ and verify that: (i) M is a P-stable model of $\mathcal{LP}_D$, and (ii) $G \in M$. By virtue of Lemma 21, (i) is done by a single call to an NP oracle (note that, since $\mathcal{Q}$ is fixed, $ground(\mathcal{LP}_D)$ has size polynomial in D, and can be constructed in polynomial time), and (ii) is clearly polynomial. Hence, this problem is in Σ_2^P. Consequently, recognizing whether a database D is in $\mathcal{EXP}^{\exists}_{\mathcal{PS}}(\mathcal{Q})$ is in Σ_2^P.

By Proposition 12 and Lemma 22, it follows immediately that every Σ_2^P-recognizable database collection $\mathbf{D}$ is in $\mathcal{EXP}^{\exists}_{\mathcal{PS}}[\mathbf{Q}]$. This proves the result. □

Theorem 24 $\mathcal{EXP}^{\forall}_{\mathcal{PS}}[\mathbf{Q}] = DB\text{-}\Pi_2^P$.

PROOF. We first prove that for any query $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ in $\mathbf{Q}$, recognizing whether a database D is in $\mathcal{EXP}^{\forall}_{\mathcal{PS}}(\mathcal{Q})$ is in Π_2^P. To this end, let consider the complementary problem: is it true that D *is not* in $\mathcal{EXP}^{\forall}_{\mathcal{PS}}(\mathcal{Q})$? Now, D is not in $\mathcal{EXP}^{\forall}_{\mathcal{PS}}(\mathcal{Q})$ iff there exists a P-stable model M of $\mathcal{LP}_D$ such that $G \notin M$. Following the line of the proof of Theorem 23, we can easily see that the latter problem is in Σ_2^P. Hence, recognizing whether a database D is in $\mathcal{EXP}^{\forall}_{\mathcal{PS}}(\mathcal{Q})$ is in Π_2^P.

That every Π_2^P-recognizable database collection $\mathbf{D}$ is in $\mathcal{EXP}^{\forall}_{\mathcal{PS}}[\mathbf{Q}]$ is derived from Lemma 22 and Proposition 12. By these results, there exists a query $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ with stratified $\mathcal{LP}$, whose $\exists_{\mathcal{PS}}$ answer is true on a database D iff D *is not* in $\mathbf{D}$. Let $\mathcal{Q}' = \langle \mathcal{LP}', G' \rangle$ where G' is a new propositional atom and $\mathcal{LP}' = \mathcal{LP} \cup \{G' \leftarrow \neg G\}$. Note that for every D, $\mathcal{LP}'_D$ is stratified, and hence the P-stable models of $\mathcal{LP}'_D$ coincide with the T-stable models and $\mathcal{LP}'_D$ always has a P-stable model.

It is now easy to see that G' is a $\forall_{\mathcal{PS}}$ inference of $\mathcal{LP}'_D$ iff G is not a $\exists_{\mathcal{PS}}$ inference of $\mathcal{LP}_D$, i.e., $D \in \mathbf{D}$. Consequently, $\mathcal{EXP}^{\forall}_{\mathcal{PS}}(\mathcal{Q}') = \mathbf{D}$. This proves the result. □

It is worth noting that, under certainty semantics, the addition of disjunction increases the power of P-stable models of two levels in the polynomial hierarchy. Indeed, on normal programs $\forall_{\mathcal{PS}}$ semantics coincides with well-founded semantics, which expresses only a strict subset of the polynomial queries (cf. [2]).

3.3 The Power of $\mathcal{MS}$ Semantics

We next determine the expressive power of M-stable models, and start with the possibility semantics. As for normal programs [47,48], under the possibility semantics, the expressive power of M-stable models coincides with the expressive power of T-stable models.

Theorem 25 $\mathcal{EXP}^{\exists}_{\mathcal{MS}}[\mathbf{Q}] = DB\text{-}\Sigma^P_2$.

PROOF. By Proposition 16.(b), we have that for any query $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ and any given database D, G is an $\exists_{\mathcal{MS}}$ inference of $\mathcal{LP}_D$ iff G is a $\exists_{\mathcal{PS}}$ inference of $\mathcal{LP}_D$. Thus, by Theorem 23 the result is immediate. $\Box$

In order to determine the expressive power of M-stable models under certainty semantics, we need to establish the complexity of recognizing M-stable models.

Theorem 26 *Let $\mathcal{LP}$ be a fixed program. Given a database D on $\mathcal{DB}_{\mathcal{LP}}$ and an interpretation M for $\mathcal{LP}_D$, deciding whether M is an M-stable model for $\mathcal{LP}_D$ is Π^P_2-complete.*

PROOF. *Membership in Π^P_2.* Notice that, since $\mathcal{LP}$ is fixed, $ground(\mathcal{LP}_D)$ can be constructed from D in polynomial time. Let M be an interpretation and consider the complementary problem: is it true that M is not an M-stable model? This can be decided as follows. Guess an interpretation N and verify that (i) either M is not P-stable or (ii) N is P-stable and M is a proper subset of N. By Lemma 21, both (i) and (ii) can be decided by a single call to an NP oracle. Thus, this problem is in Σ^P_2. Consequently, deciding whether M is an M-stable model for $\mathcal{LP}_D$ lies in Π^P_2.

Hardness for Π^P_2. Let $\mathcal{LP}$ be stratified and $\neg q$ be a ground literal where q is an atom. Lemma 22 implies that deciding, given D, whether $\neg q$ is true in every T-stable model of $\mathcal{LP}_D$ is Π^P_2-hard (cf. also [22]). We reduce this problem in polynomial time to our problem. Let $\mathcal{LP}'$ be the program obtained from $\mathcal{LP}$

by the following steps:

- (i) insert $\neg p$ in the body of each rule in $\mathcal{LP}$, where p is a new propositional letter;
- (ii) add the rules $p \leftarrow \neg q, \neg p$ and $q \leftarrow p$;
- (iii) for each IDB predicate s of $\mathcal{LP}$, add the rules $s(X_1, ..., X_n) \leftarrow \neg s(X_1, ..., X_n), \neg q$ and $s(X_1, ..., X_n) \leftarrow p$.

Fix a database D over $\mathcal{DB}_{\mathcal{LP}}$. Program $\mathcal{LP}'_D$ has a P-stable model on D, which is obtained if all ground atoms over IDB predicates are undefined. More precisely, let $A = \{r(t) \mid r \in \mathcal{DB}_{\mathcal{LP}},\ t \in D(r)\}$ be the set of facts stored in the extensional database and B the Herbrand Base $B_{\mathcal{LP}'_D}$ of $\mathcal{LP}'_D$ restricted to the database predicates (i.e., the predicates in $\mathcal{DB}_{\mathcal{LP}}$). Intuitively (and as easy to prove), $M = A \cup \neg.(B - A)$ is contained in every P-stable model of $\mathcal{LP}'_D$. In fact, M itself is a P-stable model of $\mathcal{LP}'_D$. To show this, we have to prove that (a) M is founded and (b) $\neg.M^-$ is a maximal unfounded set w.r.t. M.

(a) M is founded because (1) there is a fact in $\mathcal{LP}'_D(M)$ for every atom in $M^+ = A$, and (2) no other atom is derivable from $\mathcal{LP}'_D(M)$: all rules in (i)–(iii) that contain a negative literal are discarded in $\mathcal{LP}'_D(M)$, and the remaining rules $q \leftarrow p$, $s(\cdots) \leftarrow p$ can not fire because p is not derivable (no rule with p in the head exists in $\mathcal{LP}'_D(M)$).

(b) $\neg.M^- = (B - A)$ is an unfounded set for $\mathcal{LP}'_D$ w.r.t. M, as in $\mathcal{LP}'_D$ there is no rule with an atom from $(B - A)$ in the head. Finally, $\neg.M^-$ can not be extended to a larger consistent unfounded set (that is, no other undefined atom can be declared false): p does not belong to any unfounded set of M, since its unfoundness would require the truth of q because of rule $p \leftarrow \neg q, \neg p$; consequently, due to the rules $q \leftarrow p$ and $s(\cdots) \leftarrow p$, also q and all atoms $s(\cdots)$ does not belong to any unfounded set of M (Condition 3 of Definition 2 is not applicable to these rules since the they are non-disjunctive, Condition 1 and 2 do not apply as the body p is neither false nor unfounded).

Therefore, M is a P-stable model of $\mathcal{LP}'_D$. On the other hand, it is immediate that M is included in every P-stable model. It thus follows that M is M-stable iff there exists no other P-stable model of $\mathcal{LP}'_D$. We claim that this is the case iff $\neg q$ is true in every T-stable model of $\mathcal{LP}_D$. This implies that, given M and D, deciding whether M is an M-stable model of $\mathcal{LP}'_D$ is Π^P_2-hard.

We first show the following: In every P-stable model M' of $\mathcal{LP}'_D$ such that $M' \neq M$, atom p must be false.

Indeed, p can not be true in M', as then no rule with p in the head occurs in $\mathcal{LP}'_D(M')$ (if p is true rule $p \leftarrow \neg q, \neg p$ is not in $\mathcal{LP}'_D(M')$, as its negative body literal $\neg p$ is false). If p is undefined in M', then q must not be true due to the rule $p \leftarrow \neg q, \neg p$ (otherwise, $\neg.M'^-$ is not maximal unfounded), and q must not be false due to $q \leftarrow p$. Thus q must be undefined, as must be every $s(\cdots)$ due to the rules in (iii). This would imply $M' = M$, a contradiction. Thus, p must be false in M'.

To prove the claim, it suffices to show now that a P-stable model $M' \neq M$ of $\mathcal{LP}'_D$ exists iff $\mathcal{LP}_D$ has a P-stable model in which q is true, since $\mathcal{PS}_{\mathcal{LP}_D} = \mathcal{TS}_{\mathcal{LP}_D}$ and $\mathcal{LP}_D$ has some P-stable model as it is stratified.

Assume that an M' as described exists. Since p is false in M', q must be true due to the rule $p \leftarrow \neg q, \neg p$. It is easy to verify that $M' - \{\neg p\}$ is a P-stable model of $\mathcal{LP}_D$ in which q is true. Conversely, let N be a P-stable model of $\mathcal{LP}_D$ such that q is true. Then, it is straightforward to check that $M' = N \cup \{\neg p\}$ is a P-stable model of $\mathcal{LP}'_D$. Since $M' \neq M$, this proves our claim. We are done. $\Box$

Thus, while recognizing P-stable models is in coNP, recognizing M-stable models is one level higher up in the polynomial hierarchy.

From this result, it is intuitive that the expressive power of certainty semantics using M-stable models is precisely one level higher up in the polynomial hierarchy than certainty semantics using P-stable models. This is actually the case.

Theorem 27 $\mathcal{EXP}^{\forall}_{\mathcal{MS}}[\mathbf{Q}] = DB\text{-}\Pi^P_3$.

PROOF. Let us first prove that, for any fixed query $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ in $\mathbf{Q}$, recognizing whether a given database D is in $\mathcal{EXP}^{\forall}_{\mathcal{MS}}(\mathcal{Q})$ lies in Π^P_3. Consider the complementary problem: is there any M-stable model M of $\mathcal{LP}_D$ such that $G \notin \mathcal{LP}_D$? To solve this problem, we guess an interpretation M and use a Σ^P_2 oracle to check whether M is not M-stable (cf. Theorem 26); if not (i.e., M is M-stable), we check in polynomial time whether $G \in M$. Therefore, this problem is in Σ^P_3; hence, recognizing whether a database D is in $\mathcal{EXP}^{\forall}_{\mathcal{MS}}(\mathcal{Q})$ is in Π^P_3.

Let us now prove that every Π^P_3 recognizable database collection $\mathbf{D}$ on a database scheme $\mathcal{DB}$ is in $\mathcal{EXP}^{\forall}_{\mathcal{MS}}[\mathbf{Q}]$. By Proposition 20, $\mathbf{D}$ is definable by a second-order formula $(\forall \mathbf{A})\Psi$, where Ψ is a Σ^1_2 formula over $\mathcal{DB}$ and $\mathbf{A} = a_1, \ldots, a_k$ is a list of predicate variables. Let $\mathcal{DB}'$ be $\mathcal{DB}$ extended with $\mathbf{A}$, and let $\mathbf{D}'$ be the collection of all databases $D' = (D, \mathbf{A})$ over $\mathcal{DB}'$ such that D is from $\mathbf{D}$ and $D' \models \Psi$.[8] Clearly, a database D on $\mathcal{DB}$ belongs to $\mathbf{D}$ iff every database $D' = (D, \mathbf{A})$ over $\mathcal{DB}'$ whose the relations $\mathbf{A}$ are drawn from

[8] Here, the universe of D' as a structure consists of the constants occurring in its relations.

the active domain of D belongs to $\mathbf{D}'$. Note that by the results in [54], $\mathbf{D}'$ is Σ_2^P-recognizable. We use this to exhibit a query $\mathcal{Q}1$ such that $\mathcal{EXP}^{\forall}_{\mathcal{MS}}(\mathcal{Q}1)$ defines $\mathbf{D}$.

We proceed in two steps. First, we show that there is a query $Q0 = \langle \mathcal{LP}0, G0\rangle$ such that, given a database D' on $\mathcal{DB}'$, the $\exists_{\mathcal{MS}}$ answer of $Q0$ is yes iff D' belongs to $\mathbf{D}'$. The program $\mathcal{LP}0$ is the program $\mathcal{LP}'$ constructed from the stratified program $\mathcal{LP}$ in the proof of Theorem 26. Indeed, since $\mathbf{D}'$ is Σ_2^P-recognizable, by Lemma 22 there exists a (generic) query $\mathcal{Q} = \langle \mathcal{LP}, G\rangle$, where $\mathcal{LP}$ is stratified and G is an atom, such that for every database D' on $\mathcal{DB}'$, the program $\mathcal{LP}_{D'}$ has some T-stable model containing G iff D' belongs to $\mathbf{D}'$.

For the program $\mathcal{LP}'$ constructed from $\mathcal{LP}$ where $q = G$, the interpretation $M = A \cup \neg.(B - A)$ defined analogously for D' is a P-stable model of $\mathcal{LP}'_{D'}$, and it is M-stable iff $\neg q$ is true in every T-stable model of $\mathcal{LP}_{D'}$. Thus, M is not M-stable iff q is true in some T-stable model of $\mathcal{LP}_{D'}$. As argued in the proof of Theorem 26, M is contained in every P-stable model M' of $\mathcal{LP}'_{D'}$, and every such M' with $M' \neq M$ must contain $\neg p$. Thus, it follows that by choosing $\mathcal{LP}0 = \mathcal{LP}'$ and $G0 = \neg p$, the query $Q0 = \langle \mathcal{LP}0, G0\rangle$ under $\forall_{\mathcal{MS}}$ semantics defines $\mathbf{D}'$.

The second step in the proof uses the query $Q0$ to construct a query $Q1 = \langle \mathcal{LP}1, G1\rangle$ which defines $\mathbf{D}$. Recall that D belongs to $\mathbf{D}$ iff every $D' = (D, \mathbf{A})$ such that the relations in $\mathbf{A}$ are drawn from the active domain of D is in $\mathbf{D}'$. Now, construct the program $\mathcal{LP}1$ from $\mathcal{LP}0$ by adding the clause

$$a(X_1, \ldots, X_n) \vee \hat{a}(X_1, \ldots, X_n) \leftarrow$$

for each predicate a from $\mathbf{A}$, where $\hat{a}$ is a new predicate of the same arity. These clauses simulate all possible choices for the predicates in $\mathbf{A}$, and generate all possible extensions $D' = (D, \mathbf{A})$ of D.

It is easy to see that every P-stable model of $\mathcal{LP}1_D$ contains either $\{a(t), \neg\hat{a}(t)\}$ or $\{\neg a(t), \hat{a}(t)\}$, for each ground atom $a(t)$. Moreover, for each possible choice, some P-stable (and hence M-stable) model of $\mathcal{LP}1_D$ exists; notice that P-stable models $M1$ and $M2$ corresponding to different choices are incomparable, i.e., $M1 \not\subseteq M2 \not\subseteq M1$. It holds that $\neg p$ is true in every M-stable model iff for each possible extension $D' = (D, \mathbf{A})$ of D, the $\forall_{\mathcal{MS}}$ answer of $Q0$ for D' is yes. Thus, by choosing $G1 = \neg p$, the $\forall_{\mathcal{MS}}$ answer of $Q1 = \langle \mathcal{LP}1, G1\rangle$ is yes for database D on $\mathcal{DB}$ iff D belongs to $\mathbf{D}$. In other words, $\mathcal{EXP}^{\forall}_{\mathcal{MS}}(\mathcal{Q}1) = \mathbf{D}$. This proves the theorem. $\Box$

Therefore, while the certainty variant of P-stable semantics has dual complexity compared to the possibility semantics, the certainty semantics based on M-stable models has higher complexity than the possibility semantics.

Remark. The use of disjunctive clauses $a(X_1, \ldots, X_n) \vee \hat{a}(X_1, \ldots, X_n) \leftarrow$ in the program $\mathcal{LP}1$ is not essential for the expressiveness of the formalism. In fact, these clauses could be replaced by the clauses

$$\begin{aligned} a(X_1, \ldots, X_n) &\leftarrow \neg \hat{a}(X_1, \ldots, X_n) \\ \hat{a}(X_1, \ldots, X_n) &\leftarrow \neg a(X_1, \ldots, X_n) \\ p &\leftarrow \; a(X_1, \ldots, X_n), \hat{a}(X_1, \ldots, X_n) \end{aligned}$$

which also serve well for the purpose.

3.4 The Power of $\mathcal{LS}$ Semantics

We first consider certainty inference. The expressiveness of the $\forall_{\mathcal{LS}}$ semantics coincides with the expressiveness of the $\forall_{\mathcal{MS}}$ semantics. This is intuitive, since recognizing L-stable models has the same complexity as recognizing M-stable models.

Theorem 28 *Let $\mathcal{LP}$ be a fixed program. Give a database D on $\mathcal{DB}_{\mathcal{LP}}$ and an interpretation M for $\mathcal{LP}_D$, deciding whether M is an L-stable model for $\mathcal{LP}_D$ is Π_2^P-complete.*

PROOF. The proof of membership in Π_2^P follows the membership part in Theorem 26, with the only difference that the check $M \subset N$ is changed to $\overline{N} \subset \overline{M}$. Hardness for Π_2^P follows from the construction in the hardness part of Theorem 26: M is an L-stable model iff M is an M-stable model. $\square$

However, the query $Q1$ constructed in the proof of Theorem 27 does not serve to show that L-stable models can express all of DB-Π_3^P under the certainty semantics. The reason is that M-stable models M_1 and M_2 corresponding to different choices for the predicates $\mathbf{A}$, i.e., $M1 \not\subseteq M2 \not\subseteq M1$ may be comparable with respect to L-stability, e.g. $\overline{M_1} \subseteq \overline{M_2}$ is possible. A slight extension of the construction is sufficient to prove the expected result.

Theorem 29 $\mathcal{EXP}^{\forall}_{\mathcal{LS}}[\mathbf{Q}] = DB\text{-}\Pi_3^P$.

PROOF. The proof that, for any fixed query $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ in $\mathbf{Q}$, deciding whether a given database D is in $\mathcal{EXP}^{\forall}_{\mathcal{LS}}(\mathcal{Q})$ lies in Π_3^P is analogous to the proof that deciding if D is in $\mathcal{EXP}^{\forall}_{\mathcal{MS}}(\mathcal{Q})$ lies in Π_3^P (cf. proof of Theorem 27).

To prove that every Π_3^P-recognizable database collection $\mathbf{D}$ on a database scheme $\mathcal{DB}$ is in $\mathcal{EXP}^{\forall}_{\mathcal{MS}}[\mathbf{Q}]$, consider the query $Q1 = \langle \mathcal{LP}1, G1 \rangle$ constructed for $\mathbf{D}$ in the proof of Theorem 27. Construct $\mathcal{LP}2$ from $\mathcal{LP}1$ by adding for each of the clauses $a(X_1, \ldots, X_n) \vee \hat{a}(X_1, \ldots, X_n) \leftarrow$ implementing the possible choices for $\mathbf{A}$, the following clauses:

$$\begin{aligned}
\tilde{a}(X_1, \ldots, X_n) &\leftarrow a(X_1, \ldots, X_n) \\
\hat{a}(X_1, \ldots, X_n) &\leftarrow u \\
\tilde{a}(X_1, \ldots, X_n) &\leftarrow u \\
u &\leftarrow \neg u
\end{aligned}$$

These additional clauses serve to destroy the symmetry between the two possible choices for satisfying $a(X_1, \ldots, X_n) \vee \hat{a}(X_1, \ldots, X_n) \leftarrow$ in a P-stable model, namely $\{a(t), \neg\hat{a}(t)\}$ and $\{\neg a(t), \hat{a}(t)\}$, which both define $a(t)$ and $\hat{a}(t)$. After the addition of the above clauses, there are two P-stable models for the choice: $\{a(t), \tilde{a}(t)\}$ and $\{\neg a(t), \hat{a}(t)\}$. These models, which are not symmetric, are not comparable with respect to L-stability. It is easy to see that by choosing $G2 = G1$, for any database D the $\forall_{\mathcal{LS}}$ answer of the query $Q2 = \langle \mathcal{LP}2, G2 \rangle$ coincides with the $\forall_{\mathcal{MS}}$ answer of $Q1$. Consequently, $\mathbf{D} \in \mathcal{EXP}^{\forall}_{\mathcal{LS}}(\mathcal{Q}2)$. This proves the result. $\square$

Now let us consider possibility semantics. The expressiveness of $\exists_{\mathcal{MS}}$ semantics does not carry over to $\exists_{\mathcal{LS}}$ semantics. In fact, $\exists_{\mathcal{LS}}$ is more expressive than $\exists_{\mathcal{MS}}$. The reason is that it is not always possible to reach from an arbitrary P-stable model an L-stable model by successively adding literals. Thus, while in the case of M-stable models, possibility semantics only needs to inspect P-stability of a model, in the case of L-stable models the property of L-stability of P-stable models has to be taken into account.

Theorem 30 $\mathcal{EXP}^{\exists}_{\mathcal{LS}}[\mathbf{Q}] = DB\text{-}\Sigma_3^P$.

PROOF. We first verify that deciding whether a database D is in $\mathcal{EXP}^{\exists}_{\mathcal{LS}}(\mathcal{Q})$ is in Σ_3^P. D is in $\mathcal{EXP}^{\exists}_{\mathcal{LS}}(\mathcal{Q})$ iff there exists some L-stable model M of $\mathcal{LP}_D$ such that G belongs to M. A guess for M can be verified in polynomial time with a single call to an Σ_2^P oracle by Theorem 28. Thus, deciding whether $D \in \mathcal{EXP}^{\exists}_{\mathcal{LS}}(\mathcal{Q})$ is in Σ_3^P.

It remains to show that every Σ_3^P-recognizable database collection $\mathbf{D}$ on a database scheme $\mathcal{DB}$ is in $\mathcal{EXP}^{\exists}_{\mathcal{LS}}[\mathbf{Q}]$.

Let $\mathbf{D}' = \overline{\mathbf{D}}$ be the complementary database collection on $\mathcal{DB}$, and consider the query $Q2 = \langle \mathcal{LP}2, G2 \rangle$ from the proof of Theorem 29, constructed for $\mathbf{D}'$

(which is in Π_3^P). Then, for any database D on $\mathcal{DB}$, $D \notin \overline{\mathbf{D}}$ (i.e., $D \in \mathbf{D}$) iff $D \notin \mathcal{EXP}^{\forall}_{\mathcal{LS}}(\mathcal{Q}2)$ iff $G2$ does not belong to some L-stable model M of $\mathcal{LP}2_D$.

Recall that $G2 = \neg p$. Notice that, as easy to see, no P-stable (and hence also no L-stable) model of $\mathcal{LP}2_D$ contains p. Indeed, p occurs only in the head of a single rule, namely $p \leftarrow \neg G, \neg p$, which can however not be used to derive p.

Now construct $\mathcal{LP}3$ by adding the following clauses to $\mathcal{LP}2$:

$$a \vee b \leftarrow \neg p$$

$$b \leftarrow p$$

$$c \leftarrow \neg a, d$$

$$d \leftarrow \neg d$$

Here, a, b, c and d are new propositional atoms. [9]

Assuming that p is undefined, the clauses from above have a single P-stable model: $M = \{\neg a\}$; on the other hand, assuming that p is false, there are two P-stable models: $M1 = \{\neg a, b\}$ and $M2 = \{a, \neg b, \neg c\}$. Notice that $M1$ defines only a and b, while $M2$ defines a, b and also c. Thus, only $M2$ is L-stable.

Now let M be an L-stable model of $\mathcal{LP}2_D$. It is easy to see that, if $\neg p \notin M$, the set $M \cup \{\neg a\}$ is an L-stable model of $\mathcal{LP}3$; otherwise, the set $M \cup \{a, \neg b, \neg c\}$ is an L-stable model of $\mathcal{LP}3$.

On the other hand, every L-stable model M' of $\mathcal{LP}3$ must be of the form $M \cup \{\neg a\}$ or $M \cup \{a, \neg b, \neg c\}$, where M is an L-stable model of $\mathcal{LP}2_D$.

It holds that $\mathcal{LP}3_D$ has an L-stable model containing $\neg a$ iff $\mathcal{LP}2_D$ has some L-stable model not containing $\neg p$.

It follows that for $G3 = \neg a$, the query $Q3 = \langle \mathcal{LP}3, G3 \rangle$ satisfies $\mathcal{EXP}^{\exists}_{\mathcal{LS}}(\mathcal{Q}3) = \mathbf{D}$. Consequently, every Σ_3^P-recognizable database collection $\mathbf{D}$ is in $\mathcal{EXP}^{\exists}_{\mathcal{LS}}[\mathbf{Q}]$. $\square$

3.5 The Power of $\mathcal{RS}$ semantics

The definition of regular model is in the spirit of the definition of M-stable model (select the maximal models from a collection of models w.r.t. inclusion), but is based on a weaker notion of model. However, the expressive power of regular models is the same as of M-stable models.

[9] One could also drop the rule $d \leftarrow \neg d$ and use the atom u from $\mathcal{LP}2$ instead of d in the rule bodies.

Theorem 31 $\mathcal{EXP}^{\exists}_{\mathcal{RS}}[\mathbf{Q}] = DB\text{-}\Sigma^P_2$.

PROOF. We first have to show that for every query $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$, given a database D, deciding whether $D \in \mathcal{EXP}^{\exists}_{\mathcal{RS}}(\mathcal{Q})$ is in Σ^P_2. From the definition of regular model, it is easy to see that G is a $\exists_{\mathcal{RS}}$ inference of $\mathcal{LP}_D$ iff there exists a 3-valued model M of $\mathcal{LP}_D$ such that M^+ is founded and $G \in M$. A guess for M can be verified with an NP oracle in polynomial time: (i) check that M is a 3-valued model, and (ii) check using the NP oracle, that there exists no total minimal model N of $\mathcal{LP}(M)$ such that $N^+ \subset M^+$. Since $\mathcal{LP}_D$ is polynomial in D, this shows that deciding whether $D \in \mathcal{EXP}^{\exists}_{\mathcal{RS}}(\mathcal{Q})$ is in Σ^P_2.

On the other hand, by Proposition 12 and Theorem 23, it follows immediately that every Σ^P_2-recognizable database collection $\mathbf{D}$ is in $\mathcal{EXP}^{\forall}_{\mathcal{RS}}[\mathbf{Q}]$. $\square$

Prior to the result on certainty expressibility, we note the complexity of recognizing regular models.

Theorem 32 *Let $\mathcal{LP}$ be a fixed program. Given a database D on $\mathcal{DB}_{\mathcal{LP}}$, and an interpretation M for $\mathcal{LP}_D$, deciding whether M is a regular model for $\mathcal{LP}_D$ is Π^P_2-complete.*

PROOF. *Membership in* Π^P_2. An interpretation M is not a regular model of $\mathcal{LP}$ iff one of the following three conditions holds: (i) M is not a 3-valued model of $\mathcal{LP}$; (ii) M^+ is not a minimal model of $\mathcal{LP}(M)$; or (iii) there is some interpretation $N \supset M$ such that N is a 3-valued model of $\mathcal{LP}$ and N^+ is a minimal model of $LP(N)$.

Condition (i) can be efficiently checked, and (ii) can be easily checked with a call to an NP oracle. Finally, (iii) can be checked with a call to a Σ^P_2 oracle, since a guess for N can be verified with an NP oracle in polynomial time. Thus, deciding whether M is not regular is in Σ^P_2, which implies that deciding if M is regular is in Π^P_2.

Hardness for Π^P_2. This part is shown by a reduction from deciding whether a ground literal is true in every T-stable model of a stratified program $\mathcal{LP}_D$. The reduction is identical to the one in the proof of Theorem 26: The interpretation $M = A \cup \neg.(B - A)$ is contained in every regular model of $\mathcal{LP}'_D$, and is in fact a regular model iff $\neg q$ is true in every T-stable model of $\mathcal{LP}_D$. Note that M is founded and a 3-valued model, because M is a P-stable model. Along the lines in the proof of Theorem 26, one can show that p must be false and q true in every founded 3-valued model M' such that $M' \neq M$; the maximal such M' correspond to the T-stable models of $\mathcal{LP}_D$ in which q is true. The result follows. $\square$

Theorem 33 $\mathcal{EXP}^{\forall}_{\mathcal{RS}}[\mathbf{Q}] = DB\text{-}\Pi^P_3$.

PROOF. For any query in $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$, recognizing whether a database D belongs to $\mathcal{EXP}^{\forall}_{\mathcal{RS}}(\mathcal{Q})$ is in Π^P_3. In fact, $D \notin \mathcal{EXP}^{\forall}_{\mathcal{RS}}(\mathcal{Q})$ iff $\mathcal{LP}_D$ has some regular model M such that $G \notin M$. By the preceding Theorem 32, a guess for M can be verified in polynomial time with an oracle for Σ^P_2. Thus, it follows that $\mathcal{EXP}^{\forall}_{\mathcal{RS}}(\mathcal{Q})$ is a Π^P_3-recognizable database collection.

The proof the every Π^P_3-recognizable database collection $\mathbf{D}$ is in $\mathcal{EXP}^{\forall}_{\mathcal{RS}}(\mathbf{Q})$ is analogous to the respective part in the proof of Theorem 27. In fact, for the query $\mathcal{Q}1 = \langle \mathcal{LP}1, G1 \rangle$ constructed there the $\forall_{\mathcal{MS}}$ answer is identical to the $\forall_{\mathcal{RS}}$ answer. $\square$

4 Complexity

The complexity of a query language is usually measured according with two criteria, namely *data complexity* and *expression complexity* [16,57].

For the language $\mathbf{Q}$, data and expression complexity amount to the following. Let $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ be a query and D be a database in $\mathcal{DB}_{\mathcal{LP}}$.

- *Data complexity:* measure the complexity of evaluating $\mathcal{Q}$ as a function of the size of the database D, while $\mathcal{Q}$ is fixed.
- *Expression complexity:* measure the complexity of evaluating $\mathcal{Q}$ as a function of the size of $\mathcal{Q}$, while D is fixed.

Observe that, while measuring data complexity, since $\mathcal{LP}$ is fixed, the arity of the predicates is bounded by a constant and $ground(\mathcal{LP}_D)$ is polynomial in the size of the input D.

On the contrary, while measuring expression complexity, the size of $ground(\mathcal{LP}_D)$ is in general exponential in the size of the input $\mathcal{Q}$.

4.1 Data Complexity

The results on the data complexity of queries defined by partial model semantics are immediate consequences from the expressiveness results in the previous section.

Theorem 34 *(Possibility inference) Let $\mathcal{Q}$ be a fixed query. Deciding, given a database D, whether $\mathcal{Q}$ is true on D under $\exists_{\mathcal{XS}}$ semantics is (a) Σ^P_2-complete* *for $\mathcal{XS}$ from $\mathcal{PS}$, $\mathcal{MS}$, $\mathcal{RS}$, and (b) Σ_3^P-complete for $\mathcal{LS}$.*

Theorem 35 *(Certainty inference) Let $\mathcal{Q}$ be a fixed query. Deciding, given a database D, whether $\mathcal{Q}$ is true on D under $\forall_{\mathcal{XS}}$ semantics is (a) Π_2^P-complete for $\mathcal{XS} = \mathcal{PS}$, and (b) Π_3^P-complete for $\mathcal{XS}$ from $\mathcal{MS}$, $\mathcal{LS}$, and $\mathcal{RS}$.*

We complement these results with the complexity of deciding whether a P-stable model of $\mathcal{LP}_D$ exists. Recall that $\mathcal{LP}_D$ always has a regular model, and hence the same problem for regular semantics has trivial complexity.

Theorem 36 *Let $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ be a fixed query. Deciding, given database D, whether $\mathcal{LP}_D$ has a P-stable model, is Σ_2^P-complete.*

PROOF. Membership in Σ_2^P is already implicit from the membership part of the proof of Theorem 25.

Hardness for Σ_2^P follows by a simple reduction of evaluating a fixed Σ_2^P-hard query $Q = \langle \mathcal{LP}, G \rangle$ under $\exists_{\mathcal{TS}}$ semantics, where $\mathcal{LP}$ is stratified and G is an atom. The existence of such $\mathcal{Q}$ follows from Lemma 22. Notice that for every D, by Proposition 12, $\mathcal{PS}_{\mathcal{LP}_D} = \mathcal{TS}_{\mathcal{LP}_D}$, and thus each P-stable model of $\mathcal{LP}_D$ is total. Let $\mathcal{LP}'$ be the program obtained by adding to $\mathcal{LP}$ the following clauses, where a, b and c are new propositional atoms:

$$\begin{aligned} a \vee b \vee c &\leftarrow \neg G \\ b &\leftarrow \neg a, \neg G \\ c &\leftarrow \neg b, \neg G \\ a &\leftarrow \neg c, \neg G \end{aligned}$$

These clauses do not have a P-stable model. Asserting that G is true, however, $\{\neg a, \neg b, \neg c\}$ is a P-stable model. Since every P-stable model of $\mathcal{LP}_D$ contains either G or $\neg G$, it is thus easy to see that $\mathcal{LP}'_D$ has a P-stable model iff $\mathcal{LP}_D$ has a P-stable model containing G. This shows Σ_2^P-hardness. □

Corollary 37 *Let $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ be a fixed query. Deciding, given database D, whether $\mathcal{LP}_D$ has an M-stable model, is Σ_2^P-complete. The same holds for L-stable models.*

PROOF. From Definition 8, M-stable and L-stable models exist for a program iff it admits some P-stable model. The statement therefore follows from Theorem 36. □

Notice that in [21] it was shown that deciding whether a given propositional program has a P-stable (equivalently, a 3-valued stable) model is Σ_2^P-complete. This result, which is on expression complexity of propositional $DATALOG^{\vee,\neg}$ programs, is an immediate consequence of the preceding theorem (which is on data complexity), while it needs a little extrawork to derive the latter result from the former.

4.2 Expression Complexity

Intuitively, the expression complexity is exponentially higher than the data complexity, since the size of $\mathcal{LP}_D$ is exponential in the the size of D. Thus, we expect that the program complexity of partial model semantics parallels the data complexity in the exponential analog of the polynomial hierarchy NP $= \Sigma_1^P$, $\Sigma_2^P = \mathrm{NP}^{\mathrm{NP}}$, $\Sigma_3^P = \mathrm{NP}^{\Sigma_2^P} \ldots$; the proper analog is the weak exponential hierarchy, which consists of the classes NEXP $= \Sigma_1^E$, $\Sigma_2^E = \mathrm{NEXP}^{\mathrm{NP}}$, $\Sigma_3^E = \mathrm{NEXP}^{\Sigma_2^P}, \ldots$. It turns out that this intuition is correct.

Theorem 38 *(Possibility inference) Let D be a fixed database. Deciding, given a query $\mathcal{Q}$, whether $\mathcal{Q}$ is true on D under $\exists_{\mathcal{XS}}$ semantics is (a) Σ_2^E-complete for $\mathcal{XS}$ from $\mathcal{PS}$, $\mathcal{MS}$, $\mathcal{RS}$, and (b) Σ_3^E-complete for $\mathcal{LS}$.*

Theorem 39 *(Certainty inference) Let D be a fixed database. Deciding, given a query $\mathcal{Q}$, whether $\mathcal{Q}$ is true on D under $\forall_{\mathcal{XS}}$ semantics is (a) Π_2^E-complete for $\mathcal{XS} = \mathcal{PS}$, and (b) Π_3^E-complete for $\mathcal{XS}$ from $\mathcal{MS}$, $\mathcal{LS}$, and $\mathcal{RS}$, where $\Pi_k^E = co\Sigma_k^E$, $k \geq 0$.*

PROOF. The membership parts of the theorems are easy to see from the proofs that for a fixed query $\mathcal{Q}$, $\mathcal{EXP}_{\mathcal{XS}}^{\forall}(\mathcal{Q})$ (resp. $\mathcal{EXP}_{\mathcal{XS}}^{\exists}(\mathcal{Q})$) define database collections in corresponding classes of the polynomial hierarchy.

The hardness parts follow from the facts that evaluating a given second order formula Φ from the Σ_k^1 (resp. Π_k^1) fragment on a fixed database D is complete for Σ_k^E (resp. Π_k^E) [30], and that the proofs of all expressiveness results in the previous section implicitly provide polynomial time transformations of evaluating a formula Φ on D into evaluating an equivalent query $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ on D. Indeed, the proof of Lemma 22 in [22] constructs a query $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ equivalent to a Σ_2^1 formula Ψ; tracing the proof, it is easy to see that $\mathcal{Q}$ can be effectively computed in polynomial time. Thus, each of the queries $\mathcal{Q}i = \langle \mathcal{LP}i, Gi \rangle$ in the expressibility proofs from above can be clearly constructed in polynomial time from the given formulas Φ. This gives polynomial time transformations of evaluating formulas Φ on D into evaluating queries $\mathcal{Q}$ on D, and proves the hardness parts. $\square$

5 The Impact of Disjunction and Negation

The previous section gave a complete description of the expressive power and complexity of evaluating queries $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ based on $DATALOG^{\vee,\neg}$ programs $\mathcal{LP}$ that resort to partial model semantics. An interesting issue is the impact of syntactical restrictions on the query programs $\mathcal{LP}$. In particular, comparing the power of disjunction with the power of negation is intriguing [22].

In an account of this issue, we focus here on the most powerful concepts of partial models, which are those that minimize undefinedness, and discuss queries whose programs make limited use of disjunction and negation. Starting from normal positive programs (i.e., pure datalog programs), we consider the effect of allowing the (combined) use of the following constructs:

- stratified negation ($\neg_s$)
- arbitrary negation ($\neg$)
- headcycle-free disjunction [6,7] ($\vee_h$)
- arbitrary disjunction ($\vee$)

Headcycle-free disjunction is a syntactical property of programs that gives no account to negation. A program $\mathcal{LP}$ is *headcycle-free* iff there are no two distinct atoms $a, b \in B_{\mathcal{LP}}$ such that a and b are on a negation-free cycle of the dependency graph of $ground(\mathcal{LP})$ (i.e., a and b mutually depend on each other without negation) and a, b occur together in the head of some clause of $ground(\mathcal{LP})$. It is known that admitting headcycle-free disjunction besides full negation does not increase the complexity of T-stable models. The following proposition, which is implicit in [6], is straightforward from the fact that checking whether a total model of a positive headcycle-free program is minimal is polynomial [8].

Proposition 40 *(cf. [6,8]) Let $\mathcal{LP}$ be a ground (i.e., propositional) headcycle-free program and let M be an interpretation of it. Then, deciding if M is a T-stable model of $\mathcal{LP}$ is polynomial.*

We say that a query $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ uses *headcycle-free disjunction* iff $\mathcal{LP}$ is disjunctive and $\mathcal{LP}_D$ is headcycle-free for each D, and similarly that $\mathcal{Q}$ is *(locally) stratified* iff $\mathcal{LP}_D$ is (locally) stratified for each D.

The expressive power of headcycle-free, stratified queries (that we denote by $\mathbf{Q}_{\vee_h,\neg_s}$) has been investigated in [8] in a different setting. There, it is shown that every polynomial time computable output-query (a query that computes an output relation) can be "weakly expressed" in $\mathbf{Q}_{\vee_h,\neg_s}$ using T-stable models, i.e., the query result is given by some T-stable model; if the input databases are ordered, it can be expressed in a stronger sense, i.e., the result is given by every T-stable model. Notice that such a notion of semantics is in general different from the usual notion of certainty or possibility inference. For bound (i.e., boolean) queries, weak expressibility coincides with possibility inference. Notice that the expressive power of headcycle-free stratified bound queries is higher than DB-P, and in fact DB-NP (see Figure 2).

By $\mathbf{Q}_{np}$ we denote the set of queries $\mathcal{Q} = \langle \mathcal{LP}, G \rangle$ such that $\mathcal{LP}$ is a normal positive program; for any combination X of constructs from above, we denote by $\mathbf{Q}_X$ the extension of $\mathbf{Q}_{np}$ in $\mathbf{Q}$ where the constructs in X may be used in query programs. Note that the programs in $\mathbf{Q}_\vee$ and $\mathbf{Q}_{\vee_h}$ are $\neg$-free, while the programs in $\mathbf{Q}_\neg$ and $\mathbf{Q}_{\neg_s}$ are $\vee$-free.

The inclusion order diagrams of the resulting different fragments of $\mathbf{Q}$, together with their expressive powers under $\exists_{\mathcal{XS}}$ and $\forall_{\mathcal{XS}}$ inference, where $\mathcal{XS}$ is from $\mathcal{LS}, \mathcal{MS}, \mathcal{RS}$, are displayed in Figure 2. Notice that different combinations X may give rise to the same fragment (e.g., $X = \vee, \vee_h, \neg, \neg_s$ and $X' = \vee, \neg$); we have chosen for each fragment the smallest X as a representative.

There are two main observations on the diagrams. First, the diagrams for $\exists_{\mathcal{LS}}$ inference differs from the diagram for $\exists_{\mathcal{MS}}$ and $\exists_{\mathcal{RS}}$ inference only on the three points along the right upper part of the outline (that is, on $\mathbf{Q}_{\vee,\neg}$, $\mathbf{Q}_{\vee_h,\neg}$, and $\mathbf{Q}_\neg$). Coincidence on the remaining part is an immediate consequence of the coincidence of L-stable, M-stable and regular models on locally stratified programs (see Proposition 12), since in the respective fragments of $\mathbf{Q}$ at most locally stratified negation can be used. Second, the diagram for $\forall_{\mathcal{LS}}$, $\forall_{\mathcal{MS}}$, and $\forall_{\mathcal{RS}}$ inference is the same and just symmetric to the diagram for $\exists_{\mathcal{LS}}$ inference. For the fragments that allow at most locally stratified negation, this is an easy consequence of the facts that the concepts of models coincide to T-stable models, that always a T-stable model exists and that on any database D, the $\exists_{\mathcal{TS}}$ answer of $\langle \mathcal{LP}, L \rangle$ is opposite to the $\forall_{\mathcal{TS}}$ answer of $\langle \mathcal{LP}, \neg L \rangle$.

Observe also that each of DB-Σ_i^P and DB-Π_i^P for $i = 1, 2, 3$ is expressed by at least one fragment. On the other hand, some fragments, namely $\mathbf{Q}_{\neg_s}$, $\mathbf{Q}_\vee$ and the more restrictive fragments, allow to express only a strict subclass of the respective classes DB-C; however, as shown below, they include queries that are hard for C (see below).

An example of a nonexpressible query for all these fragments is the Even-Constants query, which tells whether a given database D on a fixed scheme $\mathcal{DB}$ contains an even number of constants.

Proposition 41 *The Even-Constants query can not be expressed in* $\mathbf{Q}_\vee \cup \mathbf{Q}_{\neg_s}$ *using* $\exists_{\mathcal{XS}}$ *or* $\forall_{\mathcal{XS}}$ *inference, for any* $\mathcal{XS}$ *from* $\mathcal{PS}, \mathcal{TS}, \mathcal{MS}, \mathcal{LS}$, *and* $\mathcal{RS}$.

PROOF. Notice that each query in $\mathbf{Q}_\vee \cup \mathbf{Q}_{\neg_s}$ is stratified, and thus it suffices to consider $\mathcal{XS} = \mathcal{TS}$ by Proposition 12.

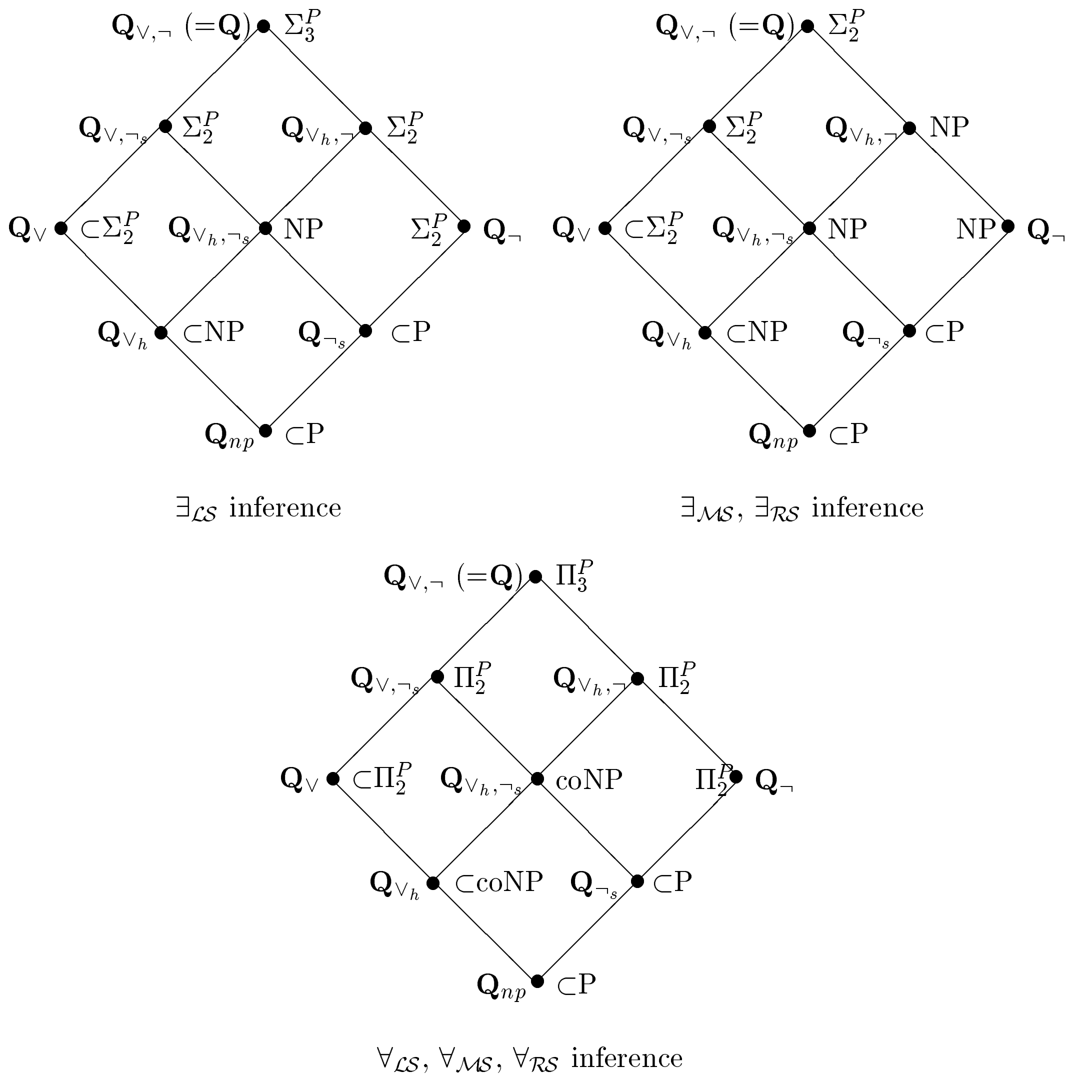

Fig. 2. Expressive power of queries based on partial models minimizing undefinedness (complexity class C stands for DB-C)

By the results in [16,33], it is immediate that the Even-Constants query can not be expressed in $\mathbf{Q}_{\neg_s}$ using $\exists_{\mathcal{TS}}$ inference: locally stratified queries are no more expressive than stratified queries [9], and stratified queries can only express fixpoint-queries [33], by which the Even-Constants query can not be defined [16]; notice that $\exists_{\mathcal{TS}}$ inference coincides with $\forall_{\mathcal{TS}}$ inference in $\mathbf{Q}_{\neg_s}$.

It was reported in [22] that for a minor variant of the Even-Constants query, there is in $\mathbf{Q}_{\vee}$ no query $\mathcal{Q} = \langle \mathcal{LP}, G\rangle$ where G is a propositional atom, such that $\mathcal{Q}$ coincides under $\exists_{\mathcal{TS}}$ inference (or $\forall_{\mathcal{TS}}$ variant as well) with this variant. That result immediately implies the same for the Even-Constants query.

Since all queries in $\mathbf{Q}_{\vee}$ are $\neg$-free (and hence stratified), it remains by Proposition 12 to show that no query $\mathcal{Q} = \langle \mathcal{LP}, \neg a\rangle$ from $\mathbf{Q}_{\vee}$, where a is a propositional atom, can define the Even-Constants query using $\exists_{\mathcal{TS}}$ (resp. $\forall_{\mathcal{TS}}$) inference.

Assume such a $\mathcal{Q}$ using $\exists_{\mathcal{TS}}$ inference exists. Define $\mathcal{Q}' = \langle \mathcal{LP} \cup \{a \vee b \leftarrow\}, b\rangle$, where b is a new propositional atom. Then, for any D the $\exists_{\mathcal{TS}}$ answer for $\mathcal{Q}'$ coincides with the $\exists_{\mathcal{TS}}$ answer for $\mathcal{Q}$. Hence, $\mathcal{Q}'$ would embody the Even-Constants query, and we reached a contradiction.

Now assume such a $\mathcal{Q}$ under $\forall_{\mathcal{TS}}$ inference exists. Then, for any D the $\exists_{\mathcal{TS}}$ answer of $\mathcal{Q}' = \langle \mathcal{LP}, a\rangle$ is opposite to the $\forall_{\mathcal{TS}}$ answer of $\mathcal{Q}$. Hence, Q' defines the complement of the Even-Constants query under $\exists_{\mathcal{TS}}$ inference. Since the Even-Constants query is not in $\mathbf{Q}_{\vee}$, it is easy to verify that also its complement is not in $\mathbf{Q}_{\vee}$; hence, the existence of $\mathcal{Q}$ leads to a contradiction. $\Box$

Let us verify the upper bounds for the expressive powers of the fragments. We first consider the fragments in which no negation or only stratified negation $\neg_s$ is allowed. Note that T-stable, L-stable, M-stable, and regular models coincide on the programs of such queries. Since each of these fragment expresses under $\exists_{\mathcal{TS}}$ inference just the complement of the queries under $\forall_{\mathcal{TS}}$ inference, we consider only $\exists$ inference in the sequel.

From Lemma 22, on Σ_2^P-recognizability of $\exists_{\mathcal{TS}}$ inference, and Proposition 41, we have the following.

Theorem 42 *$\mathcal{EXP}^{\exists}_{\mathcal{XS}}[\mathbf{Q}_{\vee,\neg_s}] = DB\text{-}\Sigma_2^P$ and*
$\mathcal{EXP}^{\exists}_{\mathcal{XS}}[\mathbf{Q}_{\vee}] \subset DB\text{-}\Sigma_2^P$, for $\mathcal{XS} = \mathcal{TS}$, $\mathcal{LS}$, $\mathcal{MS}$, $\mathcal{RS}$.

Restricting $\vee$ to $\vee_h$ besides restricting $\neg$ to $\neg_s$ leads to a decrease in expressive power. In fact, only queries in DB-NP can be expressed in $\mathbf{Q}_{\vee_h}$ using $\exists_{\mathcal{TS}}$ inference. On the other hand, if $\neg_s$ is allowed, then all of DB-NP can be expressed.

Theorem 43 *$\mathcal{EXP}^{\exists}_{\mathcal{XS}}[\mathbf{Q}_{\vee_h,\neg_s}] = DB\text{-NP}$ and*
$\mathcal{EXP}^{\exists}_{\mathcal{XS}}[\mathbf{Q}_{\vee_h}] \subset DB\text{-NP}$, for $\mathcal{XS}$ from $\mathcal{TS}$, $\mathcal{LS}$, $\mathcal{MS}$, $\mathcal{RS}$.

PROOF. By Proposition 12, it suffices to consider $\exists_{\mathcal{TS}}$ inference.

It is straightforward to derive from Proposition 40 that, for any query $\mathcal{Q} \in \mathbf{Q}_{\vee_h,\neg_s}$, deciding whether a given database D belongs to $\mathcal{EXP}^{\exists}_{\mathcal{TS}}(\mathcal{Q})$ is in NP. Thus, it follows from Proposition 41 that $\mathcal{EXP}^{\exists}_{\mathcal{XS}}[\mathbf{Q}_{\vee_h}] \subset DB$-NP.

It remains to show that every NP-recognizable database collection $\mathbf{D}$ is in $\mathcal{EXP}^{\exists}_{\mathcal{TS}}[\mathbf{Q}_{\vee_h,\neg_s}]$. We use Fagin's Theorem [25] and construct for any existential second order formula Φ a headcycle-free stratified query $\mathcal{Q} = \langle \mathcal{LP}, G\rangle$ such that $\mathcal{Q}$ is true on D under $\exists_{\mathcal{TS}}$ inference iff Φ is true on D. We assume that Φ is in second-order Skolem normal form [35], i.e.

$$\Phi = (\exists \mathbf{A})(\forall \mathbf{X})(\exists \mathbf{Y})(\theta_1(\mathbf{X}, \mathbf{Y}) \vee \cdots \vee \theta_k(\mathbf{X}, \mathbf{Y}))$$

where $\mathbf{A}$ is a list of predicate variables, $\mathbf{X}, \mathbf{Y}$ are lists of individual variables, and the θ_i are conjunctions of literals.

$\mathcal{LP}$ consists of the following clauses (cf. [52,35] for similar programs that are nondisjunctive but use unstratified negation):

(*i*) $a(X_1, \ldots, X_n) \vee \hat{a}(X_1, \ldots, X_n) \leftarrow$, for every a occurring in $\mathbf{A}$, where $\hat{a}$ is a new predicate;
(*ii*) $q(\mathbf{X}) \leftarrow \theta'_i(\mathbf{X}, \mathbf{Y})$, for $i = 1, \ldots, k$, where θ'_i denotes the literals of θ_i, among which equality (resp. inequality) atoms $X = Y$ (resp. $X \neq Y$) are replaced with $eq(X, Y)$ and $\neg eq(X, Y)$, respectively, where q and eq are new predicates;
(*iii*) $f \leftarrow \neg q(\mathbf{X})$, where f is a new propositional atom; and
(*iv*) $eq(X, X) \leftarrow$

Now let $\mathcal{Q} = \langle \mathcal{LP}, \neg f \rangle$. Clearly, $\mathcal{Q}$ is headcycle-free and stratified. It is easily seen that $\mathcal{Q}$ answers true for D under $\exists_{\mathcal{TS}}$ inference iff Φ is true on D. This proves the theorem. $\square$

The expressive power of $\mathbf{Q}_{np}$ and $\mathbf{Q}_{\neg_s}$ has been studied extensively (see [32,2]). Notice that on these fragments, $\exists_{\mathcal{TS}}$ coincides with $\forall_{\mathcal{TS}}$, since $\mathcal{TS}$ contains a single model. It is well-known that $\mathbf{Q}_{\neg_s}$ expresses a strict subset of DB-P (in fact, even a strict subset of the fixpoint-queries [33]), and that $\mathbf{Q}_{np}$ can express only monotonic queries (see [2] for monotonic queries). Thus, $\mathbf{Q}_{\vee_h}$ and $\mathbf{Q}_{\neg_s}$ allow for queries of different complexity. [10]

It remains to consider the fragments in which full negation $\neg$ is allowed. The results for the top elements in the diagrams ($\mathbf{Q}_{\vee,\neg}$) have been established in Section 3, while the results on the expressive power of $\mathbf{Q}_{\neg}$ for $\mathcal{LS}, \mathcal{MS}$ have been derived in [47,48]; by Proposition 13, they carry over to $\mathcal{RS}$.

In order to justify the results for the last fragment $\mathbf{Q}_{\vee_h,\neg}$ that we have to consider, it suffices to show that for each mode of inference, query-recognizability

[10] In fact the expressive powers of $\mathbf{Q}_{\vee_h}$ and $\mathbf{Q}_{\neg_s}$ are incomparable: the collection of 3-colorable graphs can be defined in $\mathbf{Q}_{\vee_h}$ using $\exists$-inference (as demonstrated below), but not in $\mathbf{Q}_{\neg_s}$ (since $\mathbf{Q}_{\neg_s}$ resorts to a fragment of $L^{\omega}_{\infty,\omega}$, in which 3-colorability can not be defined [18]). On the other hand, e.g. a query whether two relations R_1, R_2 contain the same tuples is simple in $\mathbf{Q}_{\neg_s}$, while it is impossible in $\mathbf{Q}_{\vee_h}$ (the proof is easy with a weak monotonicity property of *DATALOG*$^{\vee}$ query programs derived in [22]).

has the same upper bound as on $\mathbf{Q}_{\neg}$. The key result is that recognizing P-stable models of a program does not become harder if besides negation also headcycle-free disjunction is allowed.

To show this, we provide a helpful characterization of P-stable models, which is proven in the appendix. For any program $\mathcal{LP}$ and interpretation I of it, denote by $red(\mathcal{LP}, I)$ the program obtained from $ground(\mathcal{LP})$ as follows: (1) Remove every rule r such that $H(r) \cap M^+ \neq \emptyset$ or $B^-(r)$ is false in M; (2) Remove from the remaining rules all negative literals and all atoms from M^+.

Lemma 44 *Let $\mathcal{LP}$ be a program and M be a founded model of P. Then, M is a P-stable model of $\mathcal{LP}$ iff $\overline{M} = N^+$ for some $N \in \mathrm{MM}^+(red(\mathcal{LP}, M))$.*

PROOF. See appendix. □

Using this lemma, it is straightforward to derive a generalization of Proposition 40 from T-stable to P-stable models. The lower complexity of recognizing T-stable and P-stable models entails also lower complexity for recognizing M-stable and L-stable models, as well as for recognizing regular models (due to lower complexity of recognizing minimal models for headcycle-free, positive programs).

Proposition 45 *Let $\mathcal{LP}$ be a ground (i.e., propositional) headcycle-free program and let M be an interpretation of it. Then, deciding if (i) $M \in \mathcal{PS}$ is polynomial; and (ii) $M \in \mathcal{XS}$ is in* coNP*, for $\mathcal{XS}$ from $\mathcal{LS}, \mathcal{MS}$, and $\mathcal{RS}$.*

PROOF. (i): By Lemma 44, M is P-stable iff (1) M is a model of $\mathcal{LP}$, (2) M is founded (i.e., $M^+ = N_1^+$ for some $N_1 \in \mathrm{MM}^+(\mathcal{LP}(M))$), and (3) $\overline{M}^+ = N_2^+$ for some $N_2 \in \mathrm{MM}^+(red(\mathcal{LP}, M))$.

Notice that the programs $\mathcal{LP}(M)$ and $red(\mathcal{LP}, M)$ are headcycle-free, positive, and polynomial-time constructible. Since $\mathrm{MM}^+(\mathcal{LP}') = \mathcal{TS}_{\mathcal{LP}'}$ for any positive program $\mathcal{LP}'$ (this is a simple consequence of Proposition 12 and the definition of P-stable model), it follows from Proposition 40 that the tests (2) and (3) can be performed in polynomial time.

(ii): For $\mathcal{MS}$ and $\mathcal{LS}$, this follows easily from (i), and for $\mathcal{RS}$, from Proposition 40 and the fact that $\mathcal{LP}(M)$ is positive (thus minimal total models coincide with T-stable models) and headcycle-free. □

Using this properties, we derive the expressibility results for $\mathbf{Q}_{\vee_h,\neg}$.

Theorem 46 $\mathcal{EXP}^{\exists}_{\mathcal{LS}}[\mathbf{Q}_{\vee_h,\neg}] = DB\text{-}\Sigma_2^P$,

$$\mathcal{EXP}^{\exists}_{\mathcal{MS}}[\mathbf{Q}_{\vee_h,\neg}] = \mathcal{EXP}^{\exists}_{\mathcal{RS}}[\mathbf{Q}_{\vee_h,\neg}] = DB\text{-NP}, \textit{ and}$$
$$\mathcal{EXP}^{\forall}_{\mathcal{XS}}[\mathbf{Q}_{\vee_h,\neg}] = DB\text{-}\Pi^P_2, \textit{ for } \mathcal{XS} \textit{ from } \mathcal{LS}, \mathcal{MS}, \mathcal{RS}.$$

PROOF. By the expressiveness results for $\mathbf{Q}_\neg$ in [47,48] and Proposition 13, it remains to prove the upper bounds on expressiveness. For $\mathcal{EXP}^{\exists}_{\mathcal{LS}}[\mathbf{Q}_{\vee_h,\neg}]$ and $\mathcal{EXP}^{\forall}_{\mathcal{XS}}[\mathbf{Q}_{\vee_h,\neg}]$, where $\mathcal{XS}$ is from $\mathcal{LS}, \mathcal{MS}, \mathcal{RS}$, they are easily derived from Proposition 45.

From Proposition 45 it is straightforward that the database collections in $\mathcal{EXP}^{\exists}_{\mathcal{PS}}[\mathbf{Q}_{\vee_h,\neg}]$ are NP-recognizable; since $\exists_{\mathcal{MS}}$ inference coincides with $\exists_{\mathcal{PS}}$ inference, the upper bound of DB-NP for $\mathcal{EXP}^{\exists}_{\mathcal{MS}}[\mathbf{Q}_{\vee_h,\neg}]$ is justified.

Finally, note that a literal L is a $\exists_{\mathcal{RS}}$ inference of a program $\mathcal{LP}$ iff there is some founded 3-valued model of $\mathcal{LP}$ such that $L \in M$. Since $\mathcal{LP}(M)$ is positive (which means that minimal total models and T-stable models coincide) and headcycle-free, it follows from Proposition 40 that deciding if L is a $\exists_{\mathcal{RS}}$ inference of a given ground program $\mathcal{LP}$ is in NP; the DB-NP upper bound for $\mathcal{EXP}^{\exists}_{\mathcal{RS}}[\mathbf{Q}_{\vee_h,\neg}]$ follows immediately. This proves the theorem. $\Box$

Thus, we have verified all expressiveness results in Figure 2. As mentioned above, some fragments fail to express even simple queries (in particular, the Even-Constants query), but they include queries whose complexity matches with the upper bound on recognizability for the whole fragment.

To verify this, note that it is well-known that P-time hard queries can be expressed in $\mathbf{Q}_{np}$ (i.e., in standard datalog), e.g. the path systems query (cf. kola-vard-90); a Σ^P_2-hard query that uses only disjunction but no negation has been presented in [22]. Finally, 3-colorability of a finite graph $G = (V, E)$, which is well-known to be NP-complete, is easily expressed by a headcycle-free query under $\exists$-inference without the use of negation:

$$\begin{aligned}
&red(x) \vee green(x) \vee blue(x) \leftarrow \\
&\qquad\qquad f \leftarrow e(x,y), red(x), red(y) \\
&\qquad\qquad f \leftarrow e(x,y), green(x), green(y) \\
&\qquad\qquad f \leftarrow e(x,y), blue(x), blue(y)
\end{aligned}$$

Here e is the the edge predicate, respectively. It holds that for a given graph G, the $\exists_{\mathcal{XS}}$ answer of $\mathcal{Q} = \langle \mathcal{LP}, \neg f \rangle$ is true iff G is 3-colorable, where $\mathcal{XS}$ is from $\mathcal{TS}, \mathcal{LS}, \mathcal{MS}, \mathcal{RS}$.

Since the fragments of $\mathbf{Q}$ that express a database collection DB-C contain database collections that are hard for C, it follows that the data complexity of each fragment of $\mathbf{Q}$ under any considered inference mode is complete for the complexity class C, where $DB\text{-}C$ is the upper bound on the query complexity (in case of $C = \mathrm{P}$, completeness is via logspace-reductions). We formulate this here as a result for $\exists_{\mathcal{CS}}$ inference; for the other modes of inference, the results are analogous according with Figure 2.

Theorem 47 *Let $\mathcal{Q}$ be a fixed query from $\mathbf{Q}_X$. Deciding, given a database D, whether $\mathcal{Q}$ is true on D under $\exists_{\mathcal{CS}}$ semantics is (a) P-time complete for $X = np$ and $X = \neg_s$; (b)* NP*-complete for $X = \vee_h$ and $X = \vee_h, \neg_s$; (c) Σ_2^P-complete for $X = \vee$, $X = \vee, \neg_s$, $X = \neg$ and $X = \vee_h, \neg$; and (d) Σ_3^P-complete for $X = \vee, \neg$.*

The expression complexity of each fragment parallels the data complexity in the exponential analogues (EXPTIME, NEXP $= \Sigma_1^E$, $\mathrm{NEXP}^{\mathrm{NP}} = \Sigma_2^E$,...) of the classes for the data complexity (P, NP $= \Sigma_1^P$, $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^P$,...).

The hardness results for the fragments capturing a class Σ_i^P are immediate from the expressibility proofs above and [47] and the results in [30] (cf. Section 4.2 for a discussion). It remains to justify the hardness results for the other, noncapturing fragments.

It is known that inference of a ground atom from a given datalog program over a fixed database is EXPTIME-complete;[11] hence, the expression complexity of $\mathbf{Q}_{np}$ and $\mathbf{Q}_{\neg_s}$ is EXPTIME-hard. Σ_2^E-hardness of the expression complexity of $\mathbf{Q}_\vee$ queries using $\exists_{\mathcal{TS}}$ inference was shown in [22]. The proof gave a reduction of solving a generalized 3-coloring problem on graphs (coCERT3COL), given in its succinct representation, to evaluating a $\mathbf{Q}_\vee$ query using $\exists_{\mathcal{TS}}$ inference over a fixed database. The succinct problem representation consists of a boolean circuit, by which the bits of the normal representation can be uniformly computed [35,42]; this boolean circuit is simulated by a positive disjunctive program.

A straightforward simplification and minor modification of the reduction in [22] (see full paper) yields a polynomial-time reduction of deciding 3-colorability of a graph G (which is a subproblem of coCERT3COL), given in succinct representation, to evaluating a query $\mathcal{Q}$ from $\mathbf{Q}_{\vee_h}$ over a fixed database. In particular, the boolean circuit is simulated by a headcycle-free positive program. Since succinct graph 3-colorability is NEXP-hard [35], it follows that the expression complexity of $\mathbf{Q}_{\vee_h}$ is NEXP-hard. This shows the results on the expression complexity. For $\exists_{\mathcal{CS}}$ inference, they are summarized as follows. (Again, for the other inference modes, they are analogous according with Fig-

[11] This is folklore and implicit in the work of [57,16]; it can be also derived using results on relational data dependencies (G. Gottlob, personal communication).

	P-Stable	T-Stable	M-Stable	L-stable	Regular
(a) Recognition and Existence for propositional (ground) programs					
Recognition	coNP	coNP	Π_2^P	Π_2^P	Π_2^P
Existence	Σ_2^P	Σ_2^P	Σ_2^P	Σ_2^P	guaranteed
(b) Expressive Power					
Possibility	DB-Σ_2^P	DB-Σ_2^P	DB-Σ_2^P	DB-Σ_3^P	DB-Σ_2^P
Certainty	DB-Π_2^P	DB-Π_2^P	DB-Π_3^P	DB-Π_3^P	DB-Π_3^P
(c) Data Complexity					
Possibility	Σ_2^P	Σ_2^P	Σ_2^P	Σ_3^P	Σ_2^P
Certainty	Π_2^P	Π_2^P	Π_3^P	Π_3^P	Π_3^P
(d) Program Complexity					
Possibility	$\mathrm{NEXP}^{\mathrm{NP}}$	$\mathrm{NEXP}^{\mathrm{NP}}$	$\mathrm{NEXP}^{\mathrm{NP}}$	$\mathrm{NEXP}^{\Sigma_2^P}$	$\mathrm{NEXP}^{\mathrm{NP}}$
Certainty	$\mathrm{coNEXP}^{\mathrm{NP}}$	$\mathrm{coNEXP}^{\mathrm{NP}}$	$\mathrm{coNEXP}^{\Sigma_2^P}$	$\mathrm{coNEXP}^{\Sigma_2^P}$	$\mathrm{coNEXP}^{\Sigma_2^P}$

Table 1
Expressibility and complexity results on partial models

ure 2).

Theorem 48 *Let D be a fixed database. Deciding, given a query $\mathcal{Q}$ from $\mathcal{Q}_X$, whether $\mathcal{Q}$ is true on D under $\exists_{\mathcal{LS}}$ semantics is (a) EXPTIME-complete for $X = np$ and $X = \neg_s$; (b)* NEXP*-complete for $X = \vee_h$ and $X = \vee_h, \neg_s$; (c) Σ_2^E-complete for $X = \vee$, $X = \vee, \neg_s$, $X = \neg$ and $X = \vee_h, \neg$; and (d) Σ_3^E-complete for $X = \vee, \neg$.*

6 Conclusion

The results on the expressibility and complexity of partial model semantics in the general case, complemented by previously known results for total stable (T-stable) models [22,21], are compactly represented in Table 1. There, each entry of a complexity class C symbolizes C-completeness, and each entry *DB-C* that C is captured (i.e., precisely the database collections in *DB-C* are expressible).

The results in (a) have been actually derived for the data complexity of bound queries, except for T-stable models and recognition of P-stable models. (Using the results in [21,22], they can be easily extended to data complexity in these cases, too.)

It appears that using P-stable models as a substitute for T-stable models has no impact on the expressive power and the complexity of query evaluation. In fact, the full expressive power of P-stable models is available already with stratified disjunctive programs, on which all concepts of models in Table 1 coincide. Minimization of undefinedness imposes another level of data complexity in the polynomial hierarchy (except for possibility inference with M-stable or regular models), but also increases the expressive power in a balanced way, such that all queries within this complexity can be expressed.

From the point of complexity, there is no difference between M-stable, L-stable and regular models under certainty inference, while under possibility inference, L-stable models are more complex than M-stable and regular models. However, L-stable models coincide with T-stable models if some T-stable model exists, while this is not guaranteed for M-stable or regular models. In particular, the complexity of certainty inference under M-stable or regular models is not diminished by the assertion that the query program has T-stable models on every input database; this can be easily derived from the proofs of Theorems 27 and 33, which we leave to the interested reader. (Hint: use the observation that each P-stable (resp. regular) model M' of the program $\mathcal{LP}'_D$ from the proof of Theorem 26 such that $M' \neq M$ is total, and hence T-stable.)

These observations candidate L-stable models as an attractive, natural candidate among L-stable, M-stable and regular models for approximating T-stable models.

The high complexity of inference using L-stable models (Σ_3^P resp. Π_3^P) is intuitively due to three sources of complexity that interact, in a sense, orthogonally: (1) the (possibly exponential) number of L-stable models; (2) the P-stability condition for a model (foundedness and maximal unfounded set condition); and, (3) the condition of minimal undefinedness. Similar intuitions underly the other kinds of models.

We have also investigated the impact of limited disjunction and negation on expressive power and complexity of partial models that minimize undefinedness. We found that fragments of the languages capture the classes Σ_i^P, Π_i^P, $1 \leq i \leq 3$ of the polynomial hierarchy. An interesting result in this course is that negation is more powerful than disjunction (except for $\exists_{\mathcal{MS}}$, $\exists_{\mathcal{RS}}$) inference, which is not the case for T-stable models. Moreover, we found that the combination of headcycle-free disjunction and stratified negation allows to capture NP (or coNP) for every considered kind of model.

The results complement and extend previous results on disjunctive datalog and partial models for normal programs. They show that syntactic restrictions on query programs result in a broad spectrum of complexity classes that can be captured. In this context, interesting research issues remain to be addressed. One such issue are further syntactic restrictions and their effect on expressiveness and complexity. Another issue is to identify fragments of the language on which concepts of models coincide; in particular, fragments that extend locally stratified programs are interesting.

A Appendix

Lemma 44 *Let $\mathcal{LP}$ be a program and M be a founded model of P. Then, M is a P-stable model of $\mathcal{LP}$ iff $\overline{M} = N^+$ for some $N \in \mathrm{MM}^+(red(\mathcal{LP}, M))$.*

PROOF. From Definition 5, we have to prove that (a) $\neg.M^-$ is a maximal unfounded set for $\mathcal{LP}$ w.r.t. M iff (b) $\overline{M} = N^+$ for some $N \in \mathrm{MM}^+(red(\mathcal{LP}, M))$.

For convenience, let $\mathcal{LP}' = red(\mathcal{LP}, M)$. We first observe that the total interpretation I_M of $\mathcal{LP}'$ such that $I_M^+ = \overline{M}$ is a model of $\mathcal{LP}'$. Indeed, consider a rule r from $\mathcal{LP}'$ such that $H(r)$ is false in I_M. Then, by the definition of $\mathcal{LP}'$, $H(r)$ is false in M. Hence, there exists a rule r' in $ground(\mathcal{LP})$ such that $B(r) = B^+(r') - M^+ \neq \emptyset$ and $B^-(r')$ is not false in M; since r' is satisfied in M, it follows that $B^+(r')$ is false in M. Consequently, $B(r)$ is false in M, and thus also false in I_M. Hence, r is satisfied in I_M. Observe also that, since M is a model, $\neg.M^-$ is an unfounded set for $\mathcal{LP}$ w.r.t. M.

$(a) \Longrightarrow (b)$: Assume then that (b) is false. Then, some total model N of $\mathcal{LP}'$ exists such that $N \prec I_M$. We show that $X = \neg.M^- \cup (I_M^+ - N^+)$ is a consistent unfounded set for $\mathcal{LP}'$ w.r.t. M demonstrating that $\neg.M^-$ is not a maximal unfounded set. (This intuitively means that the undefined atoms in $\overline{M} - N^+$ can be switched to false without violating any rule of $\mathcal{LP}$.)

Let r be a rule in $ground(\mathcal{LP})$ with an atom from X in the head. If r has been discarded by the $red(\cdot, \cdot)$ tranformation, then it necessarily satisfies either Condition 1 or Condition 2 of Definition 2. Otherwise, let r' be the rule corresponding to r in $\mathcal{LP}'$. Then, by definition of $red(\cdot, \cdot)$ tranformation, we have: (i) $H(r) \cap M^+ = \emptyset$, (ii) $H(r) = H(r')$, (iii) $B(r') = B^+(r) - M^+$, (iv) $B(r') \subseteq \overline{M}$. Since N is a total model of $\mathcal{LP}'$, either (1) $B(r') \not\subseteq N^+$ or (2) $H(r') \cap N^+ \neq \emptyset$. If (1) is the case, then (by (iii) and (iv) above) the body of r', and consequently also the body of r, contains an atom in $\overline{M} - N^+$; thus, r satisfies Condition 1 of Definition 2, as $B(r) \cap X \neq \emptyset$. If (2) is the case, then r satisfies Condition 2 of Definition 2, as $H(r)$ (which coincides with $H(r')$) contains some element which is not in $\neg.M^- \cup X$, since N^+ is disjoint from both X and $\neg.M^-$. X is therefore a consistent unfounded set which contains properly $\neg.M^-$ ($(a) \Longrightarrow (b)$ is proven).

$(a) \Longleftarrow (b)$. Let $N \in \mathrm{MM}^+(red(\mathcal{LP}, M))$ such that $N^+ = \overline{M}$. By contradiction, assume that the unfounded set $\neg.M^-$ is not maximal (i.e., (a) does not hold). Let X be a consistent unfounded set for $\mathcal{LP}$ w.r.t. M containing $\neg.M^-$ properly. We prove that the total interpretation J such that $J^+ = N^+ - X$ is a total model for $\mathcal{LP}'$ (note that this contradicts the minimality of N, as $J \prec N$).

Let r be a rule in $\mathcal{LP}'$ whose body is true w.r.t. J. Then, the body of r is true also w.r.t. N and, as N is a model for $\mathcal{LP}'$, the head of r contains some atom in N^+, say a. Now, if a is in $N^+ - X$, then r is satisfied in J. Otherwise, a is necessarily in X. To see that r is satisfied also in the latter case, consider the rule $r' \in ground(\mathcal{LP})$ whose image (under $red(\cdot,\cdot)$ tranformation) is r. Since $H(r')$ contains some element of the unfounded set X, r' must verify either Condition 1 or Condition 2 of Definition 2 for $\mathcal{LP}$ w.r.t. M. Condition 1 can not be verified, however: $B(r') \cap \neg.M \neq \emptyset$ is not possible, since $B^-(r') \cap \neg.M \neq \emptyset$ would mean that r is not in $red(\mathcal{LP}, M)$, and $B^+(r') \cap \neg.M \neq \emptyset$ would imply that $B(r)$ is false in J. Further, $B(r') \cap X \neq \emptyset$ is not possible, since $B^+(r') \subseteq B(r) \cup M^+$, $X \cap M^+ = \emptyset$, and $B(r) \subseteq J^+ = N^+ - X$. Therefore, r' must satisfy Condition 2, that is, $H(r') \not\subseteq (\neg.M^- \cup X)$. The head of r' must hence contain some atom, say b, in $(M^+ \cup \overline{M}) - X$. b cannot be in M^+, otherwise r would not be in $red(\mathcal{LP}, M)$. Therefore b is in $\overline{M} - X$; consequently, $b \in J$. Thus, r' (whose head coincides with $H(r)$) is satisfied w.r.t. J. We arrived hence at the desired contradiction that a total interpretation $J \prec N$ is a model for $\mathcal{LP}'$. We are done. $\Box$

References

[1] A. Abiteboul, E. Simon, and V. Vianu. Non-Deterministic Languages to Express Deterministic Transformations. In *Proc. PODS-90*, pp. 218–229, 1990.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[3] K.R. Apt, R.N. Bol. Logic Programming and Negation: a Survey. *Journal of Logic Programming*, Vol. 19/20, May/July 1994, pp. 9–71.

[4] C. Baral, J. Lobo and J. Minker. Generalized Disjunctive Well-Founded Semantics for Logic Programs: Declarative Semantics. In Z. Ras, M. Zemenkova and M.L. Emrich, ed., *Proc. Fourth International Symposium on Methods for Intelligent Systems (ISMIS-90)*, pp. 465–473, North Holland, 1990.

[5] C. Baral and V. Subrahmanian. Stable and Extension Class Theory for Logic Programs and Default Logic. *Journal of Automated Reasoning*, 8:345–366, 1992.

[6] R. Ben-Eliyahu and R. Dechter. Propositional Semantics for Disjunctive Logic Programs. In *Proc. Joint International Conference and Symposium on Logic Programming (JICSLP-92)*, pp. 813–827, Washington DC, November 1992.

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

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

[9] N. Bidoit. Negation in Rule-Based Database Systems: A Survey. *Theoretical Computer Science*, 78:3–83, 1991.

[10] N. Bidoit and C. Froidevaux. General Logic Databases and Programs: Default Semantics and Stratification. *Information and Computation*, 19:15–54, 1991.

[11] M. Cadoli and M. Schaerf. A Survey of Complexity Results for Non-monotonic Logics. *Journal of Logic Programming*, 17:127–160, 1993.

[12] M. Cadoli, L. Palopoli. Circumscribing DATALOG: Expressive Power and Complexity. *Theoretical Computer Science*, to appear.

[13] F. Cacace, S. Ceri, S. Crespi-Reghizzi, L. Tanca, R. Zicari. Integrating Object-Oriented Data Modeling with a Rule-Based Programming Paradigm. *Proc. of 1990 ACM-SIGMOD Intern. Conf.*, Atlantic City, NJ, May, 23-25, 1990, pp. 225-236.

[14] S. Ceri, G. Gottlob, and L. Tanca. *Logical Programming and Databases.* Springer, 1990.

[15] D. Chimenti, R. Gamboa, R. Krishnamurthy, S. Naqvi, S. Tsur, C. Zaniolo. The LDL System Prototype. *IEEE TKDE*, Vol. 2, no. 1, 1990.

[16] A. Chandra and D. Harel. Structure and Complexity of Relational Queries. *Journal of Computer and System Sciences*, 25:99–128, 1982.

[17] Jan Chomicki and V.S. Subrahmanian. Generalized Closed World Assumption is Π^0_2-Complete. *Information Processing Letters*, 34:289–291, 1990.

[18] A. Dawar. A Restricted Second Order Logic for Finite Structures. In *Proc. Workshop on Logic and Computational Complexity*, D. Leivant (ed), Springer LNCS 960, 1995.

[19] J. Dix. Semantics of Logic Programs: Their Intuitions and Formal Properties. An Overview. *Logic, Action and Information. Proc. of the Konstanz Colloquium in Logic and Information (LogIn'92)*, pp. 241-329, DeGruyter, 1995.

[20] P. Dung. Negation as Hypotheses: An Abductive Foundation for Logic Programming. In *Proc. ICLP-91*, pp. 3–17, MIT Press, 1991.

[21] T. Eiter and G. Gottlob. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Annals of Mathematics and Artificial Intelligence*, 1995. (Abstract under different title in Proc. PODS 93.)

[22] T. Eiter, G. Gottlob, and H. Mannila. Adding Disjunction to Datalog. In *Proc. PODS-94*, pp. 267–278, 1994. Full paper *Disjunctive Datalog, ACM Transactions on Database Systems*, Sept. 1997.

[23] T. Eiter, N. Leone, and D. Saccà. On the Partial Semantics for Disjunctive Deductive Databases. *Annals of Mathematics and Artificial Intelligence*, special issue on advances in logic-based database languages, 1997.

[24] A. Kakas, P. Mancarella. Preferred Extensions are Partial Stable Models. *Journal of Logic Programming*, 14, pp. 341-348, 1992.

[25] R. Fagin. Generalized First-Order Spectra and Polynomial-Time Recognizable Sets. In R. M. Karp, editor, *Complexity of Computation*, pp. 43–74. AMS, 1974.

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

[27] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proc. Fifth Logic Programming Symposium (ILPS '88)* , pp. 1070–1080, 1988. MIT Press.

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

[29] G. Gottlob. Complexity and Expressive Power of Disjunctive Logic Programming. In M. Bruynooghe, editor, *Proc. of the International Logic Programming Symposium (ILPS)*, pp. 23–42, Ithaca NY, 1994. MIT Press.

[30] G. Gottlob, N. Leone, and H. Veith. Second-Order Logic and the Weak Exponential Hierarchies. In *Proc. Conference on Mathematical Foundations of Computer Science (MFCS-95)*, J. Wiedermann and P. Hajek (eds), Springer LNCS 969, pp. 66–81, Prague, 1995. Full paper available as CD/TR 95/80, Christian Doppler Lab for Expert Systems, Information Systems Department, TU Wien.

[31] S. Greco, N. Leone, P. Rullo. COMPLEX: An Object-Oriented Logic Programming System. *IEEE TKDE*, vol. 4, no. 4, August 1992.

[32] P. Kanellakis. Elements of Relational Database Theory. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science* B, 1990.

[33] Ph. Kolaitis. The Expressive Power of Stratified Logic Programs. *Information and Computation*, 90:50–66, 1991.

[34] Ph. Kolaitis and M. Vardi, On the Expressive Power of Datalog: Tools and a Case Study. *Journal of Computer and System Sciences*, 51:110–134, 1995.

[35] Ph. Kolaitis and C. H. Papadimitriou. Why Not Negation By Fixpoint ? *Journal of Computer and System Sciences*, 43:125–144, 1991.

[36] N. Leone, P. Rullo, A. Mecchia, G. Rossi. A Deductive Environment for Dealing with Objects and Non-Monotonic Reasoning. *IEEE Transactions on Knowledge and Data Engineering*, 1997.

[37] J. LLoyd. *Foundations of Logic Programming.* Springer, Berlin, 1984.

[38] J. Lobo, J. Minker, and A. Rajasekar. *Foundations of Disjunctive Logic Programming.* MIT Press, Cambridge, MA, 1992.

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

[40] J. Minker. On Indefinite Data Bases and the Closed World Assumption. In *Proc.* 6^{th} *Conference on Automated Deduction (CADE-82)*, D.W. Loveland (ed), Springer LNCS 138, pp. 292–308, 1982.

[41] J. Minker. Overview of Disjunctive Logic Programming. *Annals of Mathematics and Artificial Intelligence*, 12:1–24, 1994.

[42] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[43] G. Phipps, M. A. Derr, and K. A. Ross. Glue-NAIL!: A Deductive Database System. *Proc. ACM-SIGMOD Conf. on Management of Data*, pp. 308-317, 1991.

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

[45] R. Ramakrishnan, D. Srivastava, S. Sudarshan. CORAL - Control, Relations and Logic. *Proc. of the 18th VLDB Conf.*, Vancouver, British Columbia, Canada, 1992.

[46] K.A. Ross. The Well Founded Semantics for Disjunctive Logic Programs. In W. Kim, J.–M. Nicolas and S. Nishio, ed., *Proc. First International Conference on Deductive and Object-Oriented Databases (DOOD-89)*, pp. 385–402, Elsevier Science Pub., 1990.

[47] D. Saccà. The Expressive Powers of Stable Models for Bound and Unbound DATALOG Queries. *Journal of Computer and System Sciences*, 1997, to appear.

[48] D. Saccà. Deterministic and Nondeterministic Stable Model Semantics for Unbound DATALOG Queries. In *Proc. 5th Intl. Conference on Database Theory (ICDT-95)*, G. Gottlob and M.Y. Vardi (eds), LNCS 893, pp. 353–367, January 1995.

[49] D. Saccà and C. Zaniolo. Stable Models and Non-Determinism in Logic Programs with Negation. In *Proc. PODS-90*, pp. 205–218, 1990.

[50] D. Saccá, C. Zaniolo. Deterministic and Non-Deterministic Stable Models. *Journal of Logic and Computation*, 1997, to appear.

[51] C. Sakama. Possible Model Semantics for Disjunctive Databases. In W. Kim, J.–M. Nicolas and S. Nishio, ed., *Proc. First International Conference on Deductive and Object-Oriented Databases (DOOD-89)*, pp. 1055–1060, 1990.

[52] J. Schlipf. The Expressive Powers of Logic Programming Semantics. *Journal of Computer and System Sciences*, 51(1): 64–86, 1995.

[53] J. Schlipf. Complexity and Undecidability Results in Logic Programming. *Annals of Mathematics and Artificial Intelligence*, 15(3/4):257–288, 1995.

[54] L. J. Stockmeyer. The Polynomial-Time Hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[55] J. D. Ullman. *Principles of Database and Knowledge Base Systems.* Computer Science Press, 1989.

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

[57] M. Vardi. Complexity of relational query languages. In *Proc. 14th ACM STOC*, San Francisco, pp. 137–146, 1982.

[58] J.-H. You and L. Yuan. Three-Valued Formalization of Logic Programming: Is it Needed? In *Proc. PODS-90*, pp. 172–182, 1990.

[59] J.-H. You and L. Yuan. A Three-Valued Semantics for Deductive Databases and Logic Programs. *Journal of Computer and System Sciences*, 49:334–361, 1994.

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