# Complexity of Answer Set Checking and Bounded Predicate Arities for Non-ground Answer Set Programming*

Thomas Eiter, Wolfgang Faber, Michael Fink,
Gerald Pfeifer, and Stefan Woltran

Institut für Informationssysteme, TU Wien
Favoritenstraße 9-11, A-1040 Wien, Austria
{eiter,faber,michael,stefan}@kr.tuwien.ac.at,
pfeifer@dbai.tuwien.ac.at

**Abstract.** We present new complexity results on answer set checking for non-ground programs under a variety of syntactic restrictions. For several of these problems, the kind of representation of the answer set to be checked is important. In particular, we consider set-based and bitmap-based representations, which are popular in implementations of Answer Set Programming systems. Furthermore, we present new complexity results for various reasoning tasks under the assumption that predicate arities are bounded by some constant. These results imply that in such a setting – which appears to be a reasonable assumption in practice – more efficient implementations than those currently available may be feasible.

## 1 Introduction

After a long period of theoretical research on non-monotonic logic programming, in the recent years several implemented systems have become available, including DLV [17], Smodels [24], ASSAT [18], and Cmodels [3]. These systems provide a computational backbone for the Answer Set Programming (ASP) paradigm [22], a promising approach to declarative problem solving which uses concepts from knowledge representation.

Similar as in other nonmonotonic formalisms, the following major problems have been identified for ASP:

**Answer Set Existence**: Given a program $\mathcal{P}$, decide whether it has some answer set.
**Brave Reasoning**: Given a program $\mathcal{P}$, and a ground atom $A$, decide whether $A$ is true in some answer set of $\mathcal{P}$.
**Cautious Reasoning**: Given a program $\mathcal{P}$, and a ground atom $A$, decide whether $A$ is true in all answer sets of $\mathcal{P}$.
**Answer Set Checking**: Given a program $\mathcal{P}$, and a set $M$ of ground literals, decide whether $M$ is an answer set of $\mathcal{P}$.

* This work was partially supported by the Austrian Science Fund (FWF) under project Z29-N04 as well as the European Commission projects IST-2002-33570 INFOMIX, IST-2001-32429 ICONS, and IST-2001-37004 WASP.

The complexity of the former three problems has been analyzed in depth for several classes of logic programs (stratified, normal, disjunctive, head-cycle free, etc., [4, 5, 9, 11, 19]; see [8] for a survey) and the respective results show the typical exponential shift between propositional and non-ground programs (from NP to NEXP, for instance). For the problem of answer set checking (ASC), until now the picture has not been so clear. This can partly be explained by the fact that, historically, ASC has not been considered as a standard reasoning task and thus received less attention.

There are, however, at least two reasons why ASC should be considered en par with the other problems mentioned above. First, in ASP solutions to a problem are encoded in the answer sets of a corresponding program, and it is of natural interest to test whether a given claimed-to-be solution is in fact a proper one.

Second, most ASP systems, including DLV and GnT [14] (which solve problems above NP where such an approach is natural), as well as ASSAT (which employs a transformation to SAT with solutions not corresponding 1-1 with the answer sets of the original program) proceed by generating answer set candidates and subsequently checking these. ASC emerges as a computational subtask in these systems.

For the propositional case, [17] provides a nice overview on the complexity of ASC, but to our knowledge the important case of non-ground programs has not received much attention so far. This may be due to the fact that, in line with the definition of the Answer Sets semantics, the computation in the non-ground case is commonly reduced to the ground case by instantiating the input program $\mathcal{P}$ and then running an algorithm for the propositional case. However, the grounding can grow very large, and while today's state of the art ASP systems try to keep it as small as possible, it still may cause an exponential blow up in the worst case.

In the following we show that in most cases, the complexity of ASC for non-ground programs is located within the polynomial hierarchy, and thus does not follow this exponential shift which is incurred by the above reduction method. Moreover, we show that the computational complexity of ASC depends on the representation of interpretations, i.e., how possible candidates for answer sets are provided. More precisely, we distinguish between the following two forms of representation:

- a *set-oriented enumeration* of the atoms being true in the interpretation at hand, and
- a *bitmap representation*, where for each atom from a fixed universe, the corresponding bit indicates whether the atom is true in the represented interpretation.

Both forms of representation have been considered in implementations of ASP systems. In particular, the DLV system currently uses the former to generate the ground instantiation of its input, and then switches to the latter. It is thus of interest to know how the design choice for a particular representation affects (in theory) the computational properties of reasoning problems, and also to compare these theoretical results with practical experience.

Starting from the results on ASC, we furthermore present some interesting results for the complexity of logic programming in ASP where the arity of predicates is bounded by a constant. In particular, we show that under this restriction, brave and cautious reasoning for non-ground programs falls back into polynomial hierarchy; otherwise, these reasoning tasks are known to be complete for classes ranging from EXP

to (co-)$\mathrm{NEXP}^{\mathrm{NP}}$, respectively, depending on the class of programs considered. We emphasize that this result is of high practical significance, since nearly all known applications for ASP are expressed by predicates with bounded arity.

Our results on bounded arities complement previous complexity results for database queries where the number of variables in the query language is bounded by a constant [25]. These two settings are orthogonal, since bounded predicate arity still allows for arbitrarily many variables in each rule of a program, while on the other hand a bounded number of variables does not restrict the arity of predicates up front, since any variable may occur in the same atom multiple times.

## 2 Background

### 2.1 Disjunctive Datalog

In this section, we give a brief overview of the syntax and semantics of disjunctive datalog under the answer sets semantics [12]; for further background, see [10, 17].

An *atom* is an expression $p(t_1, \ldots, t_n)$, where $p$ is a *predicate* of arity $n \geq 0$ and each $t_i$ is either a variable or a constant. A *(classical) literal* $l$ is an atom $p$ (in this case, it is *positive*), or a negated atom $\neg p$ (in this case, it is *negative*). Given a literal $l$, its *complement* $\neg l$ is defined as $\neg p$ if $l = p$ and $p$ if $l = \neg p$. A set $L$ of literals is said to be *consistent* if, for every literal $l \in L$, $\neg l \notin L$. A *rule* $r$ is a formula

$$a_1 \ \mathtt{v} \ \ldots \ \mathtt{v} \ a_n \ \mathtt{:-} \ b_1, \ldots, b_k, \ \mathrm{not}\ b_{k+1}, \ldots, \ \mathrm{not}\ b_m.$$

with $n \geq 0$, $m \geq k \geq 0$, $n + m > 0$, where $a_1, \ldots, a_n, b_1, \ldots, b_m$ are literals and not denotes *default negation* (in contrast to $\neg$ which is often called *strong negation*). The *head* of $r$ is the set $H(r) = \{a_1, \ldots, a_n\}$, and the *body* of $r$ is $B(r) = \{b_1, \ldots, b_k, \ \mathrm{not}\ b_{k+1}, \ldots, \ \mathrm{not}\ b_m\}$. Furthermore, $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$.

A rule $r$ is called a *fact* if $B(r) = \emptyset$ and $H(r) \neq \emptyset$. An (integrity) *constraint* is a rule $r$ with $H(r) = \emptyset$. A rule $r$ is *normal* if $n \leq 1$, *definite* if $n = 1$, *disjunctive* if $n > 1$, and *positive* if $k = m$; if it is both normal and positive, we call it *Horn*.

A weak constraint [7] is an expression $wc$ of the form

$$\mathtt{:\sim} \ b_1, \ldots, b_k, \ \mathrm{not}\ b_{k+1}, \ldots, \ \mathrm{not}\ b_m. \ [w : l]$$

where $m \geq k \geq 0$ and $b_1, \ldots, b_m$ are literals, while $\mathit{weight}(wc) = w$ (the *weight*) and $l$ (the *level*) are positive integer constants or variables. For convenience, $w$ and/or $l$ may be omitted and are set to 1 in this case. The sets $B(wc)$, $B^+(wc)$, and $B^-(wc)$, respectively, are defined as for rules.

A *program* $\mathcal{P}$ is a finite set of rules and weak constraints. $Rules(\mathcal{P})$ denotes the set of rules and $WC(\mathcal{P})$ the set of weak constraints in $\mathcal{P}$.

Moreover, let $w^{\mathcal{P}}_{max}$ and $l^{\mathcal{P}}_{max}$ denote the maximum weight and maximum level over $WC(\mathcal{P})$, respectively.

An atom, a rule, a program, etc. is called *ground*, if no variable appears in it, and we call Horn programs without constraints and strong negation *definite Horn*.

For any program $\mathcal{P}$, let $U_\mathcal{P}$ be the set of all constants appearing in $\mathcal{P}$ (if no constant appears in $\mathcal{P}$, an arbitrary constant is added to $U_\mathcal{P}$); let $B_\mathcal{P}$ be the set of all ground literals constructible from the predicate symbols appearing in $\mathcal{P}$ and the constants of $U_\mathcal{P}$; and let $Ground(\mathcal{P})$ be the set of rules obtained by applying, to each rule and weak constraint in $\mathcal{P}$, all possible substitutions $\sigma$ from the variables in $\mathcal{P}$ to elements of $U_\mathcal{P}$.[1]

A program $\mathcal{P}$ is normal/definite/disjunctive/positive/Horn if every rule in $\mathcal{P}$ is normal/definite/disjunctive/positive/Horn. A program $\mathcal{P}$ is *head-cycle free* (HCF, [4]), iff there exists a function $h : B_\mathcal{P} \to \{0, 1, \ldots\}$ such that, for every rule $r \in \mathcal{P}$, it holds that (i) for any $l \in B^+(r)$, and for any $l' \in H(r)$, $h(l) \leq h(l')$; and (ii) for any pair $l, l' \in H(r)$, $l \neq l'$ implies $h(l) \neq h(l')$. A program $\mathcal{P}$ is *stratified* [2, 23], iff there exists a function $s : B_\mathcal{P} \to \{0, 1, \ldots\}$ such that, for every rule $r \in \mathcal{P}$, it holds that (i) for any $l \in B^+(r)$, and for any $l' \in H(r)$, $s(l) \leq s(l')$; (ii) for any $l \in B^-(r)$, and for any $l' \in H(r)$, $s(l) < (l')$; and (iii) for any pair $l, l' \in H(r)$, $s(l) = s(l')$.

*Classifying Logic Programs.* Starting from normal positive logic programs without weak constraints, we define classes DL[$L$] with $L \subseteq \{\mathrm{not}_s, \mathrm{not}, \mathtt{v}_\mathtt{h}, \mathtt{v}, w\}$. This set is used to indicate the (possibly combined) admission of

| | |
|---|---|
| $\mathrm{not}_s$: | negation, such that the program remains stratified; |
| not: | unrestricted negation; |
| $\mathtt{v}_\mathtt{h}$: | disjunction, such that the program remains HCF; |
| v: | unrestricted disjunction; |
| $w$: | weak constraints. |

For instance, DL[$\mathtt{v}_\mathtt{h}, \mathrm{not}_s$] contains all HCF stratified logic programs without weak constraints, and DL = DL[v, not, $w$] is the full language of all logic programs.

*Semantics.* Let $I$ be a consistent set of literals. Then, $I$ is *closed* under a ground rule $r$, iff the rule is *satisfied* by $I$, i.e., $H(r) \cap I \neq \emptyset$ if $B^+(r) \subseteq I \wedge B^-(r) \cap I = \emptyset$. $I$ is closed under a ground program $\mathcal{P}$, respectively $I$ satisfies $\mathcal{P}$, if $I$ is closed under all $r \in \mathcal{P}$. For $\mathcal{P}$ non-ground, we say that $I$ satisfies $\mathcal{P}$ iff $I$ satisfies $Ground(\mathcal{P})$. A (weak) constraint $c$ is *violated* by $I$, iff $B^+(c) \subseteq I \wedge B^-(c) \cap I = \emptyset$, it is satisfied otherwise. Finally, the *reduct* $r^I$ of a ground rule $r$ *relative to* $I$ is the positive rule $r'$ with $H(r') = H(r)$ and $B(r') = B^+(r)$ if $I \cap B^-(r) = \emptyset$; it is void otherwise.

A consistent *interpretation* $I \subseteq B_\mathcal{P}$ is an *answer set* for a positive program $\mathcal{P}$ without weak constraints ($\mathcal{P} \in$ DL[v]), if it is minimal (under set inclusion) among all consistent interpretations which are closed under each $r \in Ground(\mathcal{P})$.[2] In the case of arbitrary programs without weak constraints ($\mathcal{P} \in$ DL[v, not]), $I$ is an answer set of $\mathcal{P}$ iff it is an answer set of the *Gelfond-Lifschitz reduct* given by $\mathcal{P}^I = \{r^I \mid r \in Ground(\mathcal{P})\}$. Finally, in the presence of weak constraints ($\mathcal{P} \in$ DL[v, not, $w$]) $I \subseteq B_\mathcal{P}$ is an *(optimal) answer set* of $\mathcal{P}$ iff $I$ is an answer set of $Rules(\mathcal{P})$ and $H^\mathcal{P}(I)$ is minimal among all the answer sets of $Rules(\mathcal{P})$, where $H^\mathcal{P}(I)$ is defined as follows,

[1] $U_\mathcal{P}$ is usually called the *Herbrand Universe* of $\mathcal{P}$ and $B_\mathcal{P}$ the *Herbrand Literal Base* of $\mathcal{P}$.

[2] Note that we only consider *consistent answer sets*, while in [12] also the inconsistent set of all possible literals can be a valid answer set.

using an auxiliary function $f_{\mathcal{P}}$ which maps leveled weights to weights without levels:

$$f_{\mathcal{P}}(1) \ = 1, \ \text{and} \ f_{\mathcal{P}}(n) = f_{\mathcal{P}}(n-1) \cdot |WC(\mathcal{P})| \cdot w^{\mathcal{P}}_{max} + 1 \ \text{for} \, n > 1,$$
$$H^{\mathcal{P}}(I) = \sum_{i=1}^{l^{\mathcal{P}}_{max}} \left( f_{\mathcal{P}}(i) \cdot \sum_{w \in N_i^{\mathcal{P}}(I)} weight(w) \right)$$

where $N_i^{\mathcal{P}}(I)$ denotes the set of the weak constraints in level $i$ that are violated by $I$.

**Proposition 1.** *Any program in* $\mathrm{DL}[L]$ *with* $L \subseteq \{w, \mathrm{not}_s\}$ *has at most one answer set.*

**Corollary 1.** *Let* $\mathcal{P} \in \mathrm{DL}[L]$ *with* $L \subseteq \{w, \mathrm{not}_s\}$. *Then, the unique answer sets of* $\mathcal{P}$ *and* $Rules(\mathcal{P})$, *respectively, coincide, or both* $\mathcal{P}$ *and* $Rules(\mathcal{P})$ *have no answer set.*

Without loss of generality, we tacitly assume in the rest of this paper that strong negation $\neg a(\bar{t})$ is emulated by a predicate -$a(\bar{t})$ and a constraint :- $a(\bar{t})$, -$a(\bar{t})$.

### 2.2 Complexity Theory

We assume that the reader is acquainted with NP-completeness and basic notions of complexity theory, and only briefly recall some complexity classes; see [16, 21] for further background.

The classes $\Sigma_k^P$, $\Pi_k^P$, and $\Delta_k^P$ of the *Polynomial Hierarchy* (PH) are given by

$$\Delta_0^P = \Sigma_0^P = \Pi_0^P = \mathrm{P}; \ \text{and for all} \ k \geq 1, \ \Delta_k^P = \mathrm{P}^{\Sigma_{k-1}^P}, \ \Sigma_k^P = \mathrm{NP}^{\Sigma_{k-1}^P}, \ \Pi_k^P = \text{co-}\Sigma_k^P,$$

where $\mathrm{NP}^C$ denotes the class of decision problems that are solvable in polynomial time on a nondeterministic Turing machine with an oracle for any decision problem $\pi$ in the class $C$. In particular, $\mathrm{NP} = \Sigma_1^P$, co-NP $= \Pi_1^P$, and $\Delta_2^P = \mathrm{P}^{\mathrm{NP}}$. The oracle replies to a query in unit time; loosely speaking, it models a subroutine for $\pi$ with unit cost.

Observe that $\Sigma_k^P \subseteq \Delta_{k+1}^P \subseteq \Sigma_{k+1}^P \subseteq \mathrm{PSPACE}$, and each inclusion is widely conjectured to be strict. The class $\mathrm{D}^P$ contains the decision problems whose *yes* instances are characterized by the conjunction of an NP property and an independent co-NP property. Note that $\mathrm{D}^P$ is contained in $\Delta_2^P$. Furthermore, NEXP denotes the class of decision problems that are solvable in (single-)exponential time on a nondeterministic Turing machine. Oracles for NEXP and its complementary class co-NEXP are defined as for the classes of PH.

We later use *Quantified Boolean Formulas* (QBFs), i.e. formulas $\mathsf{Q}_1 X_1 \mathsf{Q}_2 X_2 \ldots \phi$, where for $i \geq 1$, $\mathsf{Q}_i \in \{\forall, \exists\}$, $X_i$ are disjoint sets of Boolean variables, and $\phi$ is a Boolean expression in conjunctive normal form (CNF) over $\bigcup_i X_i$. As well known, depending on the quantifier nesting structure the satisfiability problem for a QBF is complete for a particular class of the PH, and complete for PSPACE in the general case.

Finally, we recall that to show that classes $\mathrm{DL}[L_1] \subseteq \mathrm{DL}[L_2] \subseteq \ldots \subseteq \mathrm{DL}[L_k]$ are complete for a complexity class $C$, it suffices to prove $C$-hardness for $\mathrm{DL}[L_1]$ and $C$-membership for $\mathrm{DL}[L_k]$. Furthermore, if we have $C$-hardness for HCF programs, then $C$-hardness for normal logic programs follows immediately by means of a faithful polynomial-time rewriting of HCF programs to equivalent normal programs, cf. [4].

| | $\{\}$ | $\{\mathbf{w}\}$ | $\{\mathrm{not_s}\}$ | $\{\mathrm{not_s}, \mathbf{w}\}$ | $\{\mathrm{not}\}$ | $\{\mathrm{not}, \mathbf{w}\}$ |
|---|---|---|---|---|---|---|
| $\{\}$ | P | P | P | P | P | co-NP |
| $\{\mathtt{v_h}\}$ | P | co-NP | P | co-NP | P | co-NP |
| $\{\mathtt{v}\}$ | co-NP | $\Pi_2^P$ | co-NP | $\Pi_2^P$ | co-NP | $\Pi_2^P$ |

**Table 1.** Complexity of answer set checking for various fragments of DL, prop. case.

### 2.3 Previous Results

As mentioned in the Introduction, previous work on the complexity of ASP mostly considered the case of propositional programs. Table 1, which is taken from [17], provides a complete overview for those fragments of the language we also consider in this paper.

The rows specify the form of disjunction allowed (in particular, $\{\}$ = no disjunction). The columns specify the support for negation and weak constraints. In other words, the union of the respective sets determining the row and the column of a specific entry yields a set $L$, indicating the considered fragment DL[$L$].

## 3 Representation of Interpretations

As already mentioned, answer set checking (ASC) for programs in the general, non-ground case depends on the assumption that we make on the representation of interpretations. In practice, the following two concepts have proven useful:

**SR** An interpretation $I \subseteq B_{\mathcal{P}}$ is represented as (an enumeration of) the set of atoms which are true in that interpretation, i.e., an enumeration of all $a \in I$;

**BR** An interpretation $I \subseteq B_{\mathcal{P}}$ is represented as a bitmap, i.e., for each atom $a \in B_{\mathcal{P}}$, we have a bit $b_a$ which is 1 if $a \in I$ and which is 0 if $a \notin I$.

**Lemma 1.** *SR is more compact than BR, but ASC under BR is not more complex than under SR, as SR can be produced from BR in polynomial time (and in log space). Any upper bound for ASC under SR is thus also an upper bound for ASC under BR modulo polynomial transformation. In turn, hardness results for BR carry over to SR.*

*On the other hand, if the Herbrand base is small (polynomial in the size of the Herbrand universe $U_{\mathcal{P}}$ and the program $\mathcal{P}$), BR can be produced from SR in polynomial time, and hardness results for SR carry over to BR.*

Note, that if we restrict programs to have the arities of predicates bounded by some constant, then representations SR and BR of an interpretation $I$ are polynomially intertranslatable. This remains true if we just restrict the arity of intensional predicates, i.e., predicates which are defined by rules and are not just given as (disjunctive) ground facts. In this case, interpretations (as sets) have size polynomial in the size of the problem instance. We will make use of this observation in Section 5, where we derive new complexity results for brave and cautious reasoning over such programs.

## 4 Complexity of Answer Set Checking

**Theorem 1.** *The complexity of answer set checking in* DL *under both the set representation SR and the bitmap representation BR is given by the respective entries in the following table in terms of a completeness result.*

| *SR / BR* | $\{\}$ | $\{\mathbf{w}\}$ | $\{\mathrm{not_s}\}$ | $\{\mathrm{not_s}, \mathbf{w}\}$ | $\{\mathrm{not}\}$ | $\{\mathrm{not}, \mathbf{w}\}$ |
|---|---|---|---|---|---|---|
| $\{\}$ | $\mathrm{D}^P$ | $\mathrm{D}^P$ | $\mathrm{D}^P$ | $\mathrm{D}^P$ | $\mathrm{D}^P$ | co-NEXP / $\Pi_2^P$ |
| $\{\mathtt{v_h}\}$ | $\mathrm{D}^P$ | co-NEXP / $\Pi_2^P$ | $\mathrm{D}^P$ | co-NEXP / $\Pi_2^P$ | $\mathrm{D}^P$ | co-NEXP / $\Pi_2^P$ |
| $\{\mathtt{v}\}$ | $\Pi_2^P$ | co-NEXP$^{\mathrm{NP}}$ / $\Pi_3^P$ | $\Pi_2^P$ | co-NEXP$^{\mathrm{NP}}$ / $\Pi_3^P$ | $\Pi_2^P$ | co-NEXP$^{\mathrm{NP}}$ / $\Pi_3^P$ |

Compared to propositional answer set checking, we observe that we move up only one level in the polynomial hierarchy, provided that weak constraints are not in the considered fragment, or that answer sets are represented as bitmaps.

In the following two subsections, we prove all results from Theorem 1 in detail.

### 4.1 ASC under the Set Representation (SR)

The first two results justify all co-NEXP- and co-NEXP$^{\mathrm{NP}}$-completeness results in Theorem 1.

**Lemma 2.** *ASC is in* co-NEXP$^{\mathrm{NP}}$ *for arbitrary programs; it is in* co-NEXP *for HCF programs.*

The lemma holds by a simple exponential blowup of the respective results for the ground case after a preliminary exponential grounding step.

**Lemma 3.** *ASC under SR is* co-NEXP$^{\mathrm{NP}}$*-hard for positive disjunctive logic programs and* co-NEXP*-hard for positive HCF programs.*

*Proof.* To show the lemma, we first give the following result: Let $\mathcal{P}$ be a (non-ground) positive program without weak constraints and w.l.o.g. assume $\mathcal{P}$ contains at least one (possibly disjunctive) fact, to avoid that $\mathcal{P}$ has an empty answer set. Moreover, let $a$ be a ground atom, $w$ a fresh ground atom, and consider a program $\mathcal{P}'$, which results from adding $w$ to each head in $\mathcal{P}$, and adding a weak constraint $:\sim \mathrm{not}\, a.\, [1:1]$. Then, $\{w\}$ is an answer set for $\mathcal{P}'$ iff $\mathcal{P}$ has no answer set containing $a$ (including the case that $\mathcal{P}$ has no answer set at all).

Hence, we reduced the complement of brave reasoning (i.e., given a program $\mathcal{P}$ without weak constraints and an atom $a$, is there no answer set of $\mathcal{P}$ containing $a$?) to ASC (i.e., given a program $\mathcal{P}$' and a consistent set of literals $I$, is $I$ an answer set of $\mathcal{P}$'?) in polynomial time. Note that a polynomial reduction is only guaranteed in the case of SR, since the interpretation $I$ where $w$ is true and everything else is false can be compactly represented in SR, but not in BR, whenever the Herbrand base is exponential in the size of $\mathcal{P}$. Moreover, note that $\mathcal{P}'$ is positive whenever $\mathcal{P}$ is positive, and that $\mathcal{P}'$ is HCF whenever $\mathcal{P}$ is HCF. Combined with the known complexity results for brave reasoning in the non-ground case [10], this shows co-NEXP$^{\mathrm{NP}}$-hardness for positive disjunctive logic programs and co-NEXP-hardness for HCF programs. □

Clearly, these lemmas imply several completeness results, in particular co-NEXP-completeness for normal logic programs is easily obtained from a polynomial rewriting of HCF programs to equivalent normal programs in polynomial time, such that the programs have the same answer sets [4]. We sometimes use this technique implicitly in the remainder of the paper.

The next two results, together with Corollary 1, cover all $\mathrm{D}^P$-entries in Theorem 1.

**Lemma 4.** *ASC is in* $\mathrm{D}^P$ *for HCF programs without weak constraints.*

*Proof.* By the usual rewriting technique to normal logic programs, it suffices to show the claimed membership for normal logic programs. This can be done as follows. Given a HCF program $\mathcal{P}$ without weak constraints and a consistent set $I$ of literals. $I$ is an answer set of $\mathcal{P}$, iff (i) $I$ satisfies the reduct $\mathcal{P}^I$, and (ii) $I$ is minimal in satisfying $\mathcal{P}^I$. We can check (i) as follows. Guess a suitable ground substitution $\theta$ of a rule $r$ such that $I$ does not satisfy $(r\theta)^I$. If such a substitution exists, $I$ does not satisfy $\mathcal{P}^I$. Hence, the test for (i) is in co-NP. Second, we can check the minimality of $I$ by providing, for each atom $a \in I$, a founded proof $Pr_a$ which is a sequence of rule applications $r_1\theta_1, \ldots, r_k\theta_k$ which derives $a$ starting from scratch, where default negation is evaluated w.r.t. $I$. Since $\mathcal{P}^I$ is Horn, the number of steps required to derive $a$ is at most the the number of atoms in $I$, which is obviously linear in the size of the problem. Hence, we can guess such proofs $Pr_a$ for all $a \in I$ at once and check them in polynomial time. To conclude, we need both a co-NP- and an NP-test, implying membership in $\mathrm{D}^P$. □

**Lemma 5.** *ASC under SR is* $\mathrm{D}^P$*-hard for Horn programs.*

*Proof.* The result is easily shown by a reduction from conjunctive query evaluation, which is NP-complete (see [1]): Given a query $a \leftarrow B$ and a database $DB$, deciding whether the query fires and derives atom $a$ is NP-complete. This holds even if all involved predicates have arity bounded by a constant. Consider $\mathcal{P} = DB_1 \cup DB_2 \cup \{a_1 \text{ :- } B_1.\ \ a_2 \text{ :- } B_2\}$ for two conjunctive queries $a_1 \leftarrow B_1$ and $a_2 \leftarrow B_2$, where $a_1 \neq a_2$, and $DB_1$ and $DB_2$ are over disjoint alphabets not containing $a_1$ and $a_2$. Obviously, $\mathcal{P}$ is Horn and polynomial in size of the databases and queries involved. It is easily seen that $DB_1 \cup DB_2 \cup \{a_1\}$ is an answer set of $\mathcal{P}$ iff $a_1 \leftarrow B_1$ evaluates to true under $DB_1$ and $a_2 \leftarrow B_2$ evaluates to false under $DB_2$; this implies $\mathrm{D}^P$-hardness. □

Remaining are the $\Pi_2^P$-entries in the third row. Again, we have two results.

**Lemma 6.** *ASC for programs without weak constraints is in* $\Pi_2^P$.

*Proof.* We show that the complementary problem is in $\Sigma_2^P$. Let $\mathcal{P}$ be a program without weak constraints and $I$ a consistent set of literals. Clearly, $I$ is not an answer set for $\mathcal{P}$ iff (i) $I$ does not satisfy $\mathcal{P}^I$ or (ii) there exists some $I' \subset I$ which satisfies $\mathcal{P}^I$ Obviously, (i) is in NP. For (ii), we have to guess $I' \subset I$ and use an NP oracle for the check. Hence, (ii) is in $\mathrm{NP}^{\mathrm{NP}}$. □

Note that for programs with weak constraints this argumentation does not hold since we have to guess an arbitrary set of literals $I' \neq I$ rather than a proper subset, in order to check whether a "cheaper" answer set of $Rules(\mathcal{P})$ exists. But then, $I'$ is not necessarily polynomial in the size of the problem input (i.e., $\mathcal{P}$ and $I$) if SR is used. (Under BR, which is discussed in the next section, this problem does not occur.)

**Lemma 7.** *ASC under SR is $\Pi_2^P$-hard for positive disjunctive programs.*

*Proof.* The proof is via a polynomial reduction of the evaluation problem for QBFs of form $\Phi = \forall X \exists Y c_1 \wedge \cdots \wedge c_k$, where the $c_i$ are clauses over $X \cup Y$. This problem is $\Pi_2^P$-hard, even if all clauses have size 3. The reduction presented here is similar to the "classic" reduction of such formulas to the problem of brave reasoning over disjunctive programs. The idea is to set up a disjunctive fact

$$t(x_i) \vee f(x_i) \text{ :- .}, \quad \text{for each } x_i \in X \tag{1}$$

using $x_i$ as a constant. For each clause $c_i = L_{i,1} \vee L_{i,2} \vee L_{i,3}$, we introduce a predicate whose arity is the number of variables from $Y$. We then define, by rules, which truth assignments to these variables make the clause true, given the truth of the variables from $X$ in $c_i$, This is best illustrated by examples. Suppose we have $c_1 = x_1 \vee \neg x_2 \vee y_3$. Then, we introduce $c_1(V)$, where the argument $V$ is reserved for the truth assignments to $y_3$, and define:

$$c_1(0) \text{ :- } t(x_1).\ \ c_1(1) \text{ :- } t(x_1).\ \ c_1(0) \text{ :- } f(x_2).\ \ c_1(1) \text{ :- } f(x_2).\ \ c_1(1) \text{ :- } f(x_1), t(x_2).$$

Informally, this states that clause $c_1$ is satisfied, if either $x_1$ is true or $x_2$ is false, and in both cases the value of the $Y$-variable is irrelevant. Or, $x_1$ is false and $x_2$ is true and the $Y$-variable is true as well. As another example, consider $c_2 = x_2 \vee \neg y_1 \vee y_5$. Here, we introduce $c_2(V_1, V_2)$, and define:

$$\begin{array}{llll} c_2(0,0) \text{ :- } t(x_2). & c_2(0,1) \text{ :- } t(x_2). & c_2(1,0) \text{ :- } t(x_2). & c_2(1,1) \text{ :- } t(x_2). \\ c_2(0,0) \text{ :- } f(x_2). & c_2(0,1) \text{ :- } f(x_2). & c_2(1,1) \text{ :- } f(x_2). & \end{array}$$

Now we set up a rule which corresponds to evaluating $\exists Y c_1 \wedge \cdots \wedge c_k$ for a given $X$ as follows:

$$w \text{ :- } c_1(\bar{Y}_1) \wedge \cdots \wedge c_k(\bar{Y}_k). \tag{2}$$

where $\bar{Y}_i$, $1 \leq i \leq k$, is a vector of variables which represent the variables from $Y$ occurring in $c_i$, put at proper position. In the case above, we have $c_1(Y_3)$ and $c_2(Y_1, Y_5)$.

Call the program built so far $\mathcal{P}_{QBF}$. It will be used in many of the subsequent proofs, as well. Note that $\mathcal{P}_{QBF}$ is positive, disjunctive, and HCF, as well as polynomial in the size of the underlying QBF. The functioning of $\mathcal{P}_{QBF}$ is as follows: The disjunctive clauses (1) generate a truth assignment to $X$, and the remaining clauses check whether $\exists Y c_1 \wedge \cdots \wedge c_k$ is true under this assignment. This holds iff $w$ can be derived from (2).

For the current lemma, we add further rules which create the maximal interpretation if $w$ is true:

$$p \text{ :- } w., \quad \text{for each ground atom } p \in B_{\mathcal{P}_{QBF}} - \{w\}. \tag{3}$$

(we could do with a much smaller interpretation, but this is not important here). Call the resulting program $\mathcal{P}$ and note that $B_{\mathcal{P}} = B_{\mathcal{P}_{QBF}}$ has polynomial size, since the arity of each predicate is at most 3. Moreover, $\mathcal{P}$ is no longer HCF, due to the rules of form 3.

The functioning of $\mathcal{P}$ is as follows. If we derive $w$ from $\mathcal{P}_{QBF}$, any element from $B_{\mathcal{P}}$ can be derived. Hence, if we have, for each possible truth assignment to $X$, a truth assignment to $Y$, such that $c_1 \wedge \cdots \wedge c_k$ is true (i.e., $\Phi$ is true), $B_{\mathcal{P}}$ is answer set of $\mathcal{P}$.

On the other hand, if there exists a truth assignment to $X$, such that no assignment to $Y$ makes $c_1 \wedge \cdots \wedge c_k$ true (i.e., $\Phi$ is false), $B_{\mathcal{P}}$ cannot be an answer set of $\mathcal{P}$, since then there exists a proper subset (not containing $w$) of $B_{\mathcal{P}}$ which is an answer set of $\mathcal{P}$. Consequently, $B_{\mathcal{P}}$ is an answer set of $\mathcal{P}$ iff $\Phi$ is true. This shows $\Pi_2^P$-hardness. □

### 4.2 ASC under the Bitmap Representation (BR)

From the discussion at the beginning of the problem description, all upper bounds for SR carry over to BR, since the classes appearing in the characterization of SR are closed under polynomial time transformations. On the other hand, the Herbrand literal bases of the programs in the $\mathrm{D}^P$-hardness proof of ASC under SR (Lemma 5) and the $\Pi_2^P$-hardness proof of ASC under SR (Lemma 7) have polynomial size in the problem input. Therefore, also these hardness results carry over to BR. Recall that this is not the case for the program used in the proof of Lemma 3. It thus remains to verify the results for those fragments where the set representation caused an exponential shift, i.e., for those classes of programs from Theorem 1 with co-NEXP-hardness, respectively $\text{co-NEXP}^{\text{NP}}$-hardness under SR.

The following result immediately clarifies the upper bounds for ASC under BR, namely $\Pi_2^P$-membership in the case of HCF programs and $\Pi_3^P$-membership in general.

**Proposition 2.** *Suppose that, for a fragment* $\mathrm{DL}[L]$*, ASC under BR is feasible in* $\Delta_{k+1}^P$*. Then, for the fragment* $L' = L \cup \{w\}$*, it is feasible in* $\Pi_{k+1}^P$*.*

*Proof.* Let $\mathcal{P} \in \mathrm{DL}[L]$ and $I$ a consistent set of literals. We have to check that $I$ is an answer set of $Rules(\mathcal{P})$ and, using the oracle, that no other answer set of $Rules(\mathcal{P})$ exists which has smaller cost. Note that the usage of the bitmap representation guarantees that the respective guesses are all polynomial in size of the problem instance. Since $\Pi_{k+1}^P$ is closed under conjunction, we can combine this into a single $\Pi_{k+1}^P$ test. □

As an immediate consequence, we get the following upper bounds.

**Lemma 8.** *ASC under BR is in* $\Pi_3^P$ *for arbitrary programs, and in* $\Pi_2^P$ *for HCF programs.*

The subsequent two results provide the matching lower bounds to complete the table entries for BR. As before, the result for positive HCF programs directly leads to the corresponding result for normal programs via the usual techniques.

**Lemma 9.** *ASC for HCF programs is* $\Pi_2^P$*-hard.*

*Proof.* The proof is by reduction of a QBF of the form $\Phi = \forall X \exists Y c_1 \wedge \cdots \wedge c_k$. Recall the program $\mathcal{P}_{QBF}$ as defined in the proof of Lemma 7, add a fresh atom $q$ in the head of each rule of $\mathcal{P}_{QBF}$, and finally add the weak constraints $:\sim q.\ [1:1]$ and $:\sim w.\ [2:1]$. The resulting program $\mathcal{P}$ is HCF (in fact, it is acyclic). We claim that $\hat{I} = \{q\}$ is the optimal answer set of $\mathcal{P}$ iff $\Phi$ is true. This can be seen as follows. First, $\hat{I}$ is an answer set of $Rules(\mathcal{P})$. This follows from the fact that $q$ occurs in the head of each rule in $\mathcal{P}$, and among them we have (disjunctive) facts – in particular those resulting from the rules (1). Due to minimality, $\hat{I}$ is the only answer set of $Rules(\mathcal{P})$ which contains $q$.

The cost of $\hat{I}$ for $\mathcal{P}$ is 1. By the weak constraints in $\mathcal{P}$, any other answer set $I$ has smaller cost than $\hat{I}$ iff $w \notin I$. This, however, amounts to the existence of a truth assignment to the variables $X$ such that $\exists Y c_1 \wedge \cdots \wedge c_k$ is false, i.e., formula $\Phi$ is false. Hence, $\hat{I}$ is an (optimal) answer set of $\mathcal{P}$ iff $\Phi$ is true. □

**Lemma 10.** *ASC is $\Pi_3^P$-hard for positive disjunctive programs.*

*Proof.* Consider an existential QBF $\Phi = \exists X_1 \forall X_2 \exists Y c_1 \wedge \cdots \wedge c_k$, take again $\mathcal{P}_{QBF}$ from the proof of Lemma 7, but now with $X = X_1 \cup X_2$, and add rules $p$ :- $w$. for each ground atom $p \in B_{\mathcal{P}_{QBF}} - \{w, t(x_i), f(x_i) \mid x_i \in X_1\}$. In contrast to Lemma 9, the resulting program is not HCF. Moreover, it is quite similar to the program used in Lemma 7, and intuitively works as follows. We guess a truth assignment $\sigma$ for the atoms $X_1$. For each of these truth assignments, the program has a corresponding answer set and exactly behaves like the program in Lemma 7 for $\Phi' = \forall X_2 \exists Y (c_1 \wedge \cdots \wedge c_k)\sigma$. In particular, $w$ is in an answer set iff $\Phi'$ is true.

Now extend the program as follows. Add a fresh atom $q$ to the head of all rules and add the two weak constraints $:\sim q.\ [1:1]$ and $:\sim \mathrm{not}\ w.\ [2:1]$. Let $\mathcal{P}$ be the resulting program, which again is obviously polynomial in the size of $\Phi$. We remark that $\mathcal{P}$ is a positive program, since negation occurs only in the weak constraints.[3] We show that $\hat{I} = \{q\}$ is an answer set of $\mathcal{P}$ iff $\Phi$ is false. This proves the claim since the evaluation problem for QBFs of form $\Phi$ is $\Sigma_3^P$-complete. Clearly, $\Phi$ is false iff there exists no truth assignment $\sigma$ to $X_1$ such that $\Phi'$ is true. By the same arguments as in the proof of Lemma 9, $\hat{I}$ is the only answer set of $Rules(\mathcal{P})$ containing $q$, and it has cost 1. Thus, $\hat{I}$ is an answer set of $\mathcal{P}$ iff $Rules(\mathcal{P})$ has no answer set $I$ containing $w$. But as already shown above, such an answer set $I$ exists iff there is a truth assignment $\sigma$ to $X_1$ such that $\Phi'$ is true, i.e. iff $\Phi$ is true. □

## 5 Complexity of Reasoning with Bounded Predicate Arities

By the results above, we obtain some interesting consequences for the complexity of answer set programming if we impose restrictions on the arities of predicates.

The complexity results for brave and cautious reasoning under bounded intensional predicate arities are summarized in Theorem 2. Observe that in the first two entries of the first line, the results for brave reasoning differ if we disallow integrity constraints and strong negation, i.e., if we restrict ourselves to definite Horn programs.

**Theorem 2.** . *The complexity of brave and cautious reasoning under bounded predicate arities is given by the respective entries in the following table.*

| *Brave / Cautious* | {} | {w} | {not$_s$} | {not$_s$, w} | {not} | {not, w} |
|---|---|---|---|---|---|---|
| {} | D$^{P\,*}$ / NP | D$^{P\,*}$ / NP | $\Delta_2^P$ | $\Delta_2^P$ | $\Sigma_2^P$ / $\Pi_2^P$ | $\Delta_3^P$ |
| {v$_h$} | $\Sigma_2^P$ / $\Pi_2^P$ | $\Delta_3^P$ | $\Sigma_2^P$ / $\Pi_2^P$ | $\Delta_3^P$ | $\Sigma_2^P$ / $\Pi_2^P$ | $\Delta_3^P$ |
| {v} | $\Sigma_3^P$ / $\Pi_2^P$ | $\Delta_4^P$ | $\Sigma_3^P$ / $\Pi_3^P$ | $\Delta_4^P$ | $\Sigma_3^P$ / $\Pi_3^P$ | $\Delta_4^P$ |

* *Without constraints and strong negation (= definite Horn) the complexity is* NP.

[3] However, the negation in the weak constraint body is essential to obtain the hardness result.

These results show, that if we move from ground (i.e., propositional) programs to non-ground programs but allow only intensional predicates with small arity, the complexity of the language moves up one level in the polynomial hierarchy (PH). Thus, unless predicates of growing arities are used, we (most likely) can not encode problems above PH, such as PSPACE-complete problems.

Subsequently, we prove all completeness results summarized in Theorem 2, starting with those fragments, where programs have at most one answer set. For space reasons, some of the more straightforward proofs are omitted.

**Lemma 11.** *Brave reasoning is in* $\mathrm{D}^P$ *for Horn programs and in* NP *for definite Horn programs. Cautious reasoning is in* NP *for Horn programs in general.*

*Proof.* For brave reasoning, we do not need to guess an interpretation $I$, but instead can guess a polynomial-size founded proof $Pr_a$ for the query literal $a$, as described in Lemma 4. If the program is definite, we do not need to take care of a violation, and thus the test is in NP. If constraints or strong negation are present, we need an additional, independent co-NP-check to ensure that no constraint is violated and obtain $\mathrm{D}^P$-membership in this case. Concerning cautious reasoning, it is sufficient to guess and check a polynomial-size founded proof for either the query $a$ or a constraint violation in order to witness cautious consequence of $a$. □

**Lemma 12.** *For definite Horn programs without weak constraints, both brave and cautious reasoning are* NP*-hard. For Horn programs without weak constraints, brave reasoning is* $\mathrm{D}^P$*-hard.*

For stratified normal programs, we have slightly higher complexity since we have to evaluate a sequence of NP problems according to the layers (i.e., the *strata* described by the function $s$ as defined in Section 2.1) of the program.

**Lemma 13.** *For stratified normal logic programs, both brave and cautious inference are* $\Delta_2^P$*-complete, where hardness holds even if no weak constraints occur.*

*Proof.* Membership follows from the fact that the number of the layers mentioned above is polynomially bounded.

We show hardness by a simple reduction from deciding the last bit of the lexicographic maximum satisfying truth assignment for a CNF $C = c_1 \wedge \cdots \wedge c_k$ over atoms $X = \{x_1, \ldots, x_n\}$, which is $\Delta_2^P$-complete, cf. [21]. Without loss of generality, each $c_i = L_{i,1} \vee L_{i,2} \vee L_{i,3}$ contains three literals and $C$ is known to be satisfiable.

We build a program $\mathcal{P}$ as follows. For every clause $c_i$ we introduce a ternary predicate and describe the truth assignments to the variables in $c_i$ which satisfy that clause by facts. For example, if $c_1 = x_1 \vee \neg x_2 \vee x_3$, we add

$$c_1(0,0,0).\ c_1(0,0,1).\ c_1(0,1,1).\ c_1(1,0,0).\ c_1(1,0,1).\ c_1(1,1,0).\ c_1(1,1,1).$$

Furthermore, we introduce a fact $true(1).$, and for each atom $x_i \in X$, we add a predicate $val_{x_i}(V)$ and rules

$$val_{x_i}(1) \text{ :- } c_1(\bar{t}_1), \ldots, c_k(\bar{t}_k), true(V_i), val_{x_{i-1}}(V_{i-1}), \ldots, val_{x_1}(V_1).,$$
$$val_{x_i}(0) \text{ :- not } val_{x_i}(1).$$

where $\bar{t}_j = V_{i_1}, V_{i_2}, V_{i_3}$, $1 \leq j \leq k$, given that the atoms of literal $L_{j,1}$, $L_{j,2}$, and $L_{j,3}$ are $x_{i_1}$, $x_{i_2}$, and $x_{i_3}$, $1 \leq i \leq k$, respectively. This completes the program.

Note that $\mathcal{P}$ is definite and stratified. The maximum satisfying truth assignment for $C$ is computed in the layers of $\mathcal{P}$, and encoded by $val_{x_i}(b_i)$., $1 \leq i \leq n$, $b_i \in \{0, 1\}$, in the unique answer set $I$ of $\mathcal{P}$. At the bottom $val_{x_1}(1)$ is derived iff $C\theta$ for $\theta = \{x_1/1\}$ is satisfiable. Otherwise, $val_{x_1}(0)$ is derived. Next, depending on the value of $val_{x_1}(b_1)$, $val_{x_2}(1)$ is derived iff $C\theta$ for $\theta = \{x_1/b_1, x_2/1\}$ is satisfiable, otherwise $val_{x_2}(0)$ is derived, and so on. Thus, $val_{x_n}(1)$ is in $I$ iff the last bit of the maximum satisfying assignment is 1, and $val_{x_n}(0)$ is in $I$ otherwise. Since $\mathcal{P}$ is constructible from $C$ in polynomial time, and the arities of the predicates in $\mathcal{P}$ are bounded by a constant, $\Delta_2^P$-hardness of brave / cautious reasoning for stratified normal programs without weak constraints follows. □

For the remaining fragments without weak constraints, complexity of brave (resp. cautious) reasoning has an obvious upper bound of $\Sigma_{k+1}^P$ (resp. $\Pi_{k+1}^P$), if answer set checking is in $\Delta_{k+1}^P$. The following results give the matching lower bounds.

**Lemma 14.** *For positive HCF programs without weak constraints, brave reasoning is $\Sigma_2^P$-hard, and cautious reasoning is $\Pi_2^P$-hard.*

**Lemma 15.** *For positive disjunctive programs without weak constraints, brave reasoning is $\Sigma_3^P$-hard, and cautious reasoning is $\Pi_2^P$-hard. If (stratified) negation is added cautious reasoning is $\Pi_3^P$-hard.*

Finally, for the remaining fragments with weak constraints, the upper bounds easily follow from the complexity of ASC, employing the usual schema to first compute the cost of an optimal answer set in a binary search, and then decide the problem with a single oracle call.

For the matching hardness results, we use the following canonical problems on QBFs with free variables, which generalize the lexicographic maximum satisfying assignment problem.

**Proposition 3.** *Given a QBF $\Phi[X] = \mathsf{Q}_1 Y_1 \mathsf{Q}_2 Y_2 \cdots \mathsf{Q}_j Y_j c_1 \wedge \cdots \wedge c_k$ with alternating quantifies $\mathsf{Q}_1, \mathsf{Q}_2, \ldots, \mathsf{Q}_j = \exists$, and where the $c_i$ are clauses over $X \cup Y_1 \cup \cdots \cup Y_j$, deciding the last bit of the lexicographic maximum assignment to the free variables $X$ which makes (i) $\Phi[X]$ true for $j$ even; (ii) $\Phi[X]$ false for $j$ odd is $\Delta_{j+2}^P$-complete, $j \geq 0$, even if it is guaranteed that at least one such assignment exists.*

**Lemma 16.** *For positive HCF programs, brave and cautious inference are $\Delta_3^P$-hard.*

*Proof.* Consider $\Phi[X] = \exists Y c_1 \wedge \cdots \wedge c_k$ and the program $\mathcal{P}$ which extends $\mathcal{P}_{QBF}$ by the weak constraints $:\sim w.\ [: n+1]$ and $:\sim f(x_i).\ [: n-i+1]$ for each $i \in \{1, \ldots, n\}$. As in previous proofs, $\mathcal{P}$ is positive and HCF. The answer sets of $Rules(\mathcal{P})$ correspond to all possible truth assignments to $X$ and contain $w$ iff $\Phi[X]$ evaluates to true under the corresponding guess for $X$. Now we are interested in those assignments making $\Phi[X]$ false and w.l.o.g. we assume that at least one such assignment exists. The intuition of the weak constraints then is as follows: If $w$ is in an answer set of $Rules(\mathcal{P})$ then the highest penalty is given. For the remaining ones, we first eliminate those where $x_1$ is

set to false, then those where $x_2$ is set to false, and so on. The unique optimal answer set of $\mathcal{P}$ thus corresponds to the lexicographic maximum assignment to $X$ which makes $\Phi[X]$ false. Hence, via both brave and cautious reasoning, we can decide the last bit of this assignment. By Proposition 3, this shows $\Delta_3^P$-hardness. □

**Lemma 17.** *For positive disjunctive programs, both inference tasks are $\Delta_4^P$-hard.*

## 6 Conclusions and Implications

We have provided new complexity results on answer set checking (ASC) for non-ground programs under various syntactic restrictions. We have demonstrated that the choice of the representation for interpretations is crucial in terms of ASC complexity. If set-oriented enumeration (SR) is chosen, an exponential blowup can be witnessed for programs containing weak constraints and disjunctions or unstratified negation, while for the choice of bitmap representation (BR), these problems just move up one level within the polynomial hierarchy.

In general, comparing ASC for propositional programs to ASC for non-ground programs, the complexity moves from P to $\mathrm{D}^P$ and from co-NP to $\Pi_2^P$ for program classes without weak constraints or with weak constraints but without disjunctions and unstratified negation under both SR and BR. For other classes however, complexity shifts from co-NP to co-NEXP and from $\Pi_2^P$ to $\mathrm{co\text{-}NEXP}^{\mathrm{NP}}$ if SR is chosen, while it moves from co-NP to $\Pi_2^P$ and from $\Pi_2^P$ to $\Pi_3^P$ for BR.

Furthermore, we have demonstrated that bounding predicate arities moves the complexity of both brave and cautious reasoning over non-ground programs from an area ranging from EXP to $\mathrm{co\text{-}NEXP}^{\mathrm{NP}}$ to an area ranging from NP to $\Delta_4^P$. Since bounding arities is a natural restriction, these results are of high practical interest.

In particular, the results in Section 5 imply that it should be feasible to find methods for non-ground query answering that operate in polynomial space and exponential time if the predicate arities are bounded. The classical approach of computing the (more or less) full ground program as a first step, which is employed in virtually all competitive answer set programming systems (DLV, Smodels, ASSAT, Cmodels), cannot meet those resource restrictions, as the ground program may in general consume exponential space.

Top-down algorithms appear to be good candidates for fulfilling these requirements, but so far there is only little work on this topic: In [6] a resolution method is presented for cautious reasoning with DL[not] programs. Several approaches to top-down derivation for DL[v] programs have been proposed, see e.g. [26] and references therein. Very recently, in [15] a method for top-down cautious query answering for $\mathrm{DL}[\mathrm{not}_s, \mathtt{v}]$ programs has been described. Unfortunately, it is not clear whether the space and time complexities of these approaches stay in polynomial space and time, respectively. We are not aware of any top-down methods for full $\mathrm{DL}[\mathrm{not}, \mathtt{v}]$ programs or programs containing weak constraints.

Another approach to overcome exponential space requirements could be to perform a focused grounding using the query, thus in principle emulating a top-down derivation. In [13] a generalization of the magic sets technique to DL[v] has been described, but it is highly unclear to what extent such an optimization technique can reduce grounding

size, and in particular whether exponential space consumption can always be avoided, given that standard grounding techniques are employed on a rewritten program.

## References


1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.
2. K. R. Apt, H. A. Blair, and A. Walker. Towards a theory of declarative knowledge. In [20].
3. Y. Babovich. Cmodels homepage, since 2002. `http://www.cs.utexas.edu/users/tag/cmodels.html`.
4. R. Ben-Eliyahu and R. Dechter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics & Artifi cial Intelligence*, 12:53–87, 1994.
5. R. Ben-Eliyahu-Zohary and L. Palopoli. Reasoning with minimal models: Efficient algorithms and applications. *Artifi cial Intelligence*, 96:421–449, 1997.
6. P. A. Bonatti. Resolution for skeptical stable model semantics. *Journal of Automated Reasoning*, 27(4):391–421, 2001.
7. F. Buccafurri, N. Leone, and P. Rullo. Enhancing disjunctive datalog by constraints. *IEEE TKDE*, 12(5):845–860, 2000.
8. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.
9. T. Eiter and G. Gottlob. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics & Artifi cial Intelligence*, 15(3/4):289–323, 1995.
10. T. Eiter, G. Gottlob, and H. Mannila. Disjunctive Datalog. *ACM TODS*, 22(3):364–418, 1997.
11. T. Eiter, N. Leone, and D. Saccá. Expressive power and complexity of partial models for disjunctive deductive databases. *Theoretical Computer Science*, 206(1–2):181–218, 1998.
12. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
13. S. Greco. Binding propagation techniques for the optimization of bound disjunctive queries. *IEEE TKDE*, 15(2):368–385, 2003.
14. T. Janhunen, I. Niemelä, P. Simons, and J.-H. You. Partiality and disjunctions in stable model semantics. In *Proc. KR 2000*, pp. 411–419. Morgan Kaufmann, 2000.
15. C. A. Johnson. Computing only minimal answers in disjunctive deductive databases. Technical Report cs.LO/0305007, arXiv.org, May 2003.
16. D. S. Johnson. A catalog of complexity classes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A, chapter 2. Elsevier Science Pub., 1990.
17. N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. cs.AI/0211004, arXiv.org, Nov. 2002.
18. F. Lin and Y. Zhao. ASSAT: Computing answer sets of a logic program by SAT solvers. In *Proc. AAAI-2002*, 2002. AAAI Press / MIT Press.
19. V. W. Marek and M. Truszczyński. Autoepistemic logic. *J. ACM*, 38(3):588–619, 1991.
20. J. Minker, editor. *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufman, Washington DC, 1988.
21. C. H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.
22. A. Provetti and S. T. Cao (eds). *Proc. AAAI 2001 Spring Symposium on Answer Set Programming* (Workshop Technical Report SS-01-01). AAAI Press, 2001.
23. T. C. Przymusinski. On the declarative semantics of deductive databases and logic programs. In [20].
24. P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artifi cial Intelligence*, 138:181–234, June 2002.
25. M. Vardi. On the complexity of bounded-variable queries. In *Proc. PODS-95*, 1995.
26. A. H. Yahya. Duality for goal-driven query processing in disjunctive deductive databases. *Journal of Automated Reasoning*, 28(1):1–34, 2002.