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

Thomas Eiter and Wolfgang Faber and Michael Fink
Institut für Informationssysteme, TU Wien
 Favoritenstraße 9-11, A-1040 Wien, Austria
{eiter,faber,michael}@kr.tuwien.ac.at

Gerald Pfeifer
SUSE LINUX AG
Maxfeldstr. 5, D-90409 Nürnberg, Germany
gerald@pfeifer.com

Stefan Woltran
Institut für Informationssysteme, TU Wien
 Favoritenstraße 9-11, A-1040 Wien, Austria
stefan@kr.tuwien.ac.at

Abstract

Answer Set Programming has become a host for expressing knowledge representation problems, which reinforces the interest in efficient methods for computing answer sets of a logic program. While for propositional programs, the complexity of this task has been amply studied and is well-understood, less attention has been paid to the case of non-ground programs, which is much more important from a KR language perspective. Existing Answer Set Programming systems employ different representations of models, but the consequences of these representations for answer set computation and reasoning tasks have not been analyzed in detail. In this paper, we present novel complexity results on answer set checking for non-ground programs under two methods for representing answer sets and a variety of syntactic restrictions. In particular, we consider set-based and bitmap-based representations, which are popular in implementations of Answer Set Programming systems. Based on these results, we also derive new complexity results for the canonical reasoning tasks over answer sets, under the assumption that predicate arities are bounded by some constant. Our 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.

Introduction

After extensive theoretical research on non-monotonic logic programming, in the recent years several implemented systems have become available, e.g., (Leone et al. 2002; Simons, Niemelä, & Soininen 2002; Lin & Zhao 2002; Lierler & Maratea 2004). These systems provide the computational backbone for the Answer Set Programming (ASP) paradigm (Provetti & Son 2001), which has become a host for expressing knowledge representation problems. In ASP, problems are encoded as logic programs, such that the answer sets of such a program yield the solutions of the original problem. This approach is particularly interesting for logic-based KR formalisms which can be (efficiently) expressed by logic programs. So far, it has been successfully applied to planning, diagnosis, and inheritance reasoning, and is under investigation for other areas such as description logics and ontologies, e.g. (Swift 2004).

This, in turn, reinforces the interest in efficient algorithms and methods for computing answer sets of a logic program, cf. (Anger, Konczak, & Linke 2001; Lierler & Maratea 2004; Leone et al. 2002; Lin & Zhao 2002; Nicolas, Saubion, & Stéphan 2002; Simons, Niemelä, & Soininen 2002). While for propositional programs, the complexity of computing answer sets has been amply studied and is well-understood, less attention has been paid to non-ground programs. However, the latter are very important from a user perspective, since an expressive KR language should offer predicates allowing for natural problem representations. Indeed, all the ASP systems mentioned above support the use of predicates in some way (while functions symbols are usually disregarded or restricted).

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

Answer Set Existence: Given a program \( \mathcal{P} \), decide whether \( \mathcal{P} \) has some answer set.

Brave Reasoning: Given a program \( \mathcal{P} \), and a ground literal \( a \), decide whether \( a \) is true in some answer set of \( \mathcal{P} \).

Cautious Reasoning: Given a program \( \mathcal{P} \), and a ground literal \( a \), 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 \( \mathcal{P} \).

The complexity of the former three problems has been analyzed in depth for several classes of programs (stratified, normal, disjunctive, etc., (Ben-Eliyahu & Dechter 1994; Ben-Eliyahu-Zohary & Palopoli 1997; Eiter & Gottlob 1995; Eiter, Leone, & Saccà 1998; Marek & Truszczynski 1991); see (Dantsin et al. 2001) for a survey) and the respective results typically show an exponential shift be-
1. We show, analyzing a number of common syntactic fragments of ASP, that in most cases the complexity of ASC for non-ground programs is located within the polynomial hierarchy (PH), and thus does not follow the exponential shift which is incurred by the aforementioned grounding methods.

2. Furthermore, we show that the computational complexity of ASC depends on the representation of interpretations, i.e., how possible candidates for answer sets are provided. In practice, two concepts have proven useful:

   **SR:** An interpretation $I$ is represented as an explicit enumeration of the set of ground atoms which are true, i.e., an enumeration of all $a \in I$ (**set representation**). Commonly used instances of SR are binary trees and hash tables and variations thereof, like red-black trees.

   **BR:** An interpretation $I$ is represented as a bitmap, i.e., for each ground atom $a$, we have a bit $b_a$ which is 1 if $a \in I$ and 0 if $a \notin I$ (**bitmap representation**).

   Both forms have been used in ASP systems, and the DLV system for example, currently employs SR for grounding and BR for subsequent computations. It is thus of interest to know how the design choice for a particular representation affects (in theory) the computational properties of reasoning problems.

3. Furthermore, we present novel complexity results for ASP where the arity of predicates is bounded by a constant. We show that under this restriction, brave and cautious reasoning for non-ground programs fall back into PH; otherwise, these reasoning tasks are known to be complete for classes ranging from EXP to (co-)NEXP, 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 extend and complement previous results in the literature. More importantly, they alert to the fact that the grounding procedures used by current ASP systems are an inherent bottleneck which, as shown by our results, may be

---

### Table 1: Complexity of ASC for propositional fragments of DL. All entries are completeness results.

<table>
<thead>
<tr>
<th></th>
<th>{}</th>
<th>{w}</th>
<th>{not_a}</th>
<th>{not_a, w}</th>
<th>{not}</th>
<th>{not, w}</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>\text{co-NP}</td>
</tr>
<tr>
<td>{\forall_{\bar{a}}}</td>
<td>P</td>
<td>\text{co-NP}</td>
<td>P</td>
<td>\text{co-NP}</td>
<td>P</td>
<td>\text{co-NP}</td>
</tr>
<tr>
<td>{\forall}</td>
<td>\text{co-NP}</td>
<td>\Pi_2^p \setminus \text{co-NP}</td>
<td>\Pi_2^p \setminus \text{co-NP}</td>
<td>\Pi_2^p \setminus \text{co-NP}</td>
<td>\Pi_2^p \setminus \text{co-NP}</td>
<td>\Pi_2^p \setminus \text{co-NP}</td>
</tr>
</tbody>
</table>

### Table 2: Complexity of brave and cautious reasoning for propositional fragments of DL. All entries are completeness results.

<table>
<thead>
<tr>
<th>Brave / Cautious</th>
<th>{}</th>
<th>{w}</th>
<th>{not_a}</th>
<th>{not, w}</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
</tr>
<tr>
<td>{\forall_{\bar{a}}}</td>
<td>\text{NP} / \text{co-NP}</td>
<td>\Delta_2^P \setminus \text{NP} / \text{co-NP}</td>
<td>\Delta_2^P \setminus \text{NP} / \text{co-NP}</td>
<td>\Delta_2^P \setminus \text{NP} / \text{co-NP}</td>
</tr>
<tr>
<td>{\forall}</td>
<td>\Sigma_2^P / \text{co-NP}</td>
<td>\Delta_2^P \setminus \Sigma_2^P / \Pi_2^P \setminus \text{NP}</td>
<td>\Delta_2^P \setminus \Sigma_2^P / \Pi_2^P \setminus \text{NP}</td>
<td>\Delta_2^P \setminus \Sigma_2^P / \Pi_2^P \setminus \text{NP}</td>
</tr>
</tbody>
</table>
overcome by a different system architecture in many relevant cases. The results on bounded arities complement previous complexity results for queries to a database where the number of variables in the query language is bounded by a constant (Vardi 1995). These two settings are orthogonal, since bounded predicate arity still allows for arbitrarily many variables in each rule of a program, and conversely 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.

Preliminaries and Previous Results

In this section, we first give a brief overview of the syntax and semantics of disjunctive datalog under the answer sets semantics (Gelfond & Lifschitz 1991); for further background, see (Eiter, Gottlob, & Mannila 1997; Leone et al. 2002).

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 (disjunctive) rule \( r \) is of the form

\[
a_1 \lor \cdots \lor a_n \leftarrow b_1, \ldots, b_k, \neg b_{k+1}, \ldots, \neg b_m,
\]

with \( n \geq 0, m \geq k \geq 0, n + m > 0 \), and where \( a_1, \ldots, a_n, b_1, \ldots, b_k, \neg b_{k+1}, \ldots, \neg b_m \) are literals. We refer to “\( \lor \)" as strong negation and to "\( \neg \)" as default negation. The head of \( r \) is the set \( H(r) = \{a_1, \ldots, a_n\} \), and the body of \( r \) is \( B(r) = \{b_1, \ldots, b_k, \neg b_{k+1}, \ldots, \neg b_m\} \). Furthermore, \( B^+(r) = \{b_1, \ldots, b_k\} \) and \( B^-(r) = \{b_{k+1}, \ldots, b_m\} \).

A rule \( r \) is called fact if \( m = 0, n > 0 \), in which case the symbol \( \leftarrow \) is usually omitted; (integrity) constraint if \( n = 0 \); \( r \) is normal if \( n \leq 1 \), definite if \( n = 1 \), disjunctive if \( n > 1 \), and positive if \( k = m \), Horn if \( k = m \) and \( n = 1 \).

A weak constraint (Buccafurri, Leone, & Rullo 2000) is an expression \( wc \) of the form

\[
\neg \sim b_1, \ldots, b_k, \not b_{k+1}, \ldots, \not b_m, [w : l]
\]

where \( m \geq k \geq 0 \) and \( b_1, \ldots, b_k, \not b_{k+1}, \ldots, \not b_m \) are literals, while \( weight(wc) = w \) (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 set \( B^+(wc) \) and \( B^-(wc) \) are defined as for rules.

A program \( P \) is a finite set of rules and weak constraints. \( Rules(P) \) denotes the set of rules and \( WC(P) \) the set of weak constraints in \( P \). \( w_{\text{max}}^P \) and \( l_{\text{max}}^P \) denote the maximum weight and maximum level over \( WC(P) \), respectively. Programs are normal (resp., defeq, disjunctive, positive, Horn) if all of their rules enjoy this property. Horn programs without constraints and strong negation are definite Horn.

For any program \( P \), let \( U_P \) be the set of all constants appearing in \( P \) (if no constant appears in \( P \), an arbitrary constant is added to \( U_P \)); let \( B_P \) be the set of all ground literals constructible from the predicate symbols appearing in \( P \) and the constants of \( U_P \); and let \( Ground(P) \) be the set of rules \( r \sigma \) obtained by applying, to each rule and weak constraint \( r \in P \), all possible substitutions \( \sigma \) from the variables in \( P \) to elements of \( U_P \). \( U_P \) is usually called the Herbrand Universe of \( P \) and \( B_P \) the Herbrand Literal Base of \( P \).

Classifying Logic Programs. Starting from Horn programs without weak constraints, we define classes \( DL[L] \) with \( L \subseteq \{\text{not}, \not, \lor, v, w\} \). This set is used to indicate the (possibly combined) admission of

- not: negation; the program remains stratified;
- not: unrestricted negation;
- \( \lor \): disjunction; the program remains HCF;
- \( v \): unrestricted disjunction;
- \( w \): weak constraints.

Recall that stratified negation, not, cf. (Apt, Blair, & Walker 1988; Przymusinski 1988) allows only a layered use of default negation not, such that negative literals of any rule instantiation are in a lower layer than the head literals, which must all be in the same layer, while positive body literals may occur in the same or lower layers than head literals. As well, in head-cycle-free disjunction, \( \lor \), (Ben-Eliyahu & Dechter 1994), for short HCF, no different head literals of any rule instance positively depend mutually on each other (a head literal \( a \in H(r) \) depends on a literal \( b \), if \( b \in B^+(r) \), or some literal \( c \in B^-(r) \) depends on \( b \)).

Thus, for instance, \( DL[\lor, \not] \) contains all HCF stratified programs without weak constraints, and \( DL = DL[\lor, \not, w] \) is the full language of all logic programs.

Semantics. A ground rule \( r \) is satisfied by a consistent set of literals \( I \) iff \( H(r) \cap I \neq \emptyset \) whenever \( B^+(r) \subseteq I \) and \( B^-(r) \cap I = \emptyset \). \( I \) satisfies a ground program \( P \), if each \( r \in P \) is satisfied by \( I \). For \( P \) non-ground, we say that \( I \) satisfies \( P \) if \( I \) satisfies \( Ground(P) \). A (weak) constraint \( c \) is violated by \( I \), iff \( B^+(c) \subseteq I \) and \( B^-(c) \cap I = \emptyset \); it is satisfied otherwise.

Recall that for \( P \in DL[\lor, \not] \), a consistent set \( I \subseteq B_P \) is an answer set\(^1\) iff it is a subset-minimal set satisfying the Gelfond-Lifschitz reduct

\[
P^I = \{H(r) : B^+(r) \mid I \cap B^- (r) = \emptyset, r \in Ground(P)\}
\]

For \( P \in DL[\lor, \not, w] \), a consistent set \( I \subseteq B_P \) is an (optional) answer set of \( P \) iff \( I \) is an answer set of \( Rules(P) \) and \( H^P(I) \) is minimal among all the answer sets of \( Rules(P) \), where the penalization \( H^P(I) \) for weak constraint violation is defined as follows:

\[
H^P(I) = \sum_{i=1}^{N^P} \left( f_P(i) \cdot \sum_{w \in N^P(I)} weight(w) \right)
\]

\( f_P(1) = 1 \), and \( f_P(n) = f_P(n-1) \cdot |WC(P)| \cdot w_{\text{max}}^P + 1 \) for \( n > 1 \).

\(^{1}\)Note that we only consider consistent answer sets, while in (Gelfond & Lifschitz 1991) also the inconsistent set of all possible literals can be a valid answer set.
For any program $\mathcal{P}$, we denote the set of its answer sets by $\mathcal{AS}(\mathcal{P})$.

The following proposition is immediate from the well-known result that that any normal stratified program has at most one answer set:

**Proposition 1** For any $\mathcal{P} \in \text{DL}[L]$ with $\{L\} \subseteq \{w, \text{not}_s\}$, $|\mathcal{AS}(\mathcal{P})| \leq 1$.

Hence:

**Corollary 1** For $\mathcal{P} \in \text{DL}[L]$ with $\{L\} \subseteq \{w, \text{not}_s\}$, $\mathcal{AS}(\mathcal{P}) = \mathcal{AS}(\text{Rules}(\mathcal{P}))$.

**Previous Results.** We assume that the reader is acquainted with NP-completeness and basic notions of complexity theory, and refer to (Johnson 1990; Papadimitriou 1994) for further background.

As mentioned in the Introduction, previous work on the complexity of ASP mostly considered the case of propositional programs. Tables 1 and 2, which are taken from (Leone et al. 2002), provide a complete overview of the complexity of answer set checking and brave and cautious reasoning, respectively, for the propositional variants of the language fragments considered in this paper.

In these tables the rows specify the form of disjunction allowed (in particular, $\{\}$ = no disjunction), whereas the columns specify the support for negation and weak constraints. So the field in row $R$ and column $C$ indicates $\text{DL}[L]$, where $\{L\} = R \cup C$.

For the canonical reasoning problems in the general non-ground case, the complexity of brave and cautious reasoning in general increases by one exponential compared to the according results in the propositional case. In particular, the results shift from P to EXP, NP to NEXP, $\Delta_2^p$ to EXP$^\text{NP}$, $\Sigma_2^p$ to NEXP$^\text{NP}$, etc. For disjunctive programs and certain fragments, complexity results in the non-ground case have been derived e.g. in (Eiter, Gottlob, & Mannila 1997; Eiter, Leone, & Saccà 1998). For the other fragments, the results can be derived using complexity upgrading techniques (Eiter, Gottlob, & Mannila 1997; Gottlob, Leone, & Veith 1999).

### Complexity of Answer Set Checking

In what follows, we shall distinguish between two different representations for sets $I \subseteq B_\mathcal{P}$, for any program $\mathcal{P}$. To be more specific, we consider a set representation (SR) of $I$ as an explicit enumeration of the set of atoms $a \in I$, and a bit representation (BR) of $I$ which sets in a bitmap over all elements $a \in B_\mathcal{P}$, those bits $b_a$ to 1 where $a \in I$ holds, and the remaining bits to 0. Hence, in the case of BR the representation of a set $I$ may be exponential in the size of $I$, since $B_\mathcal{P}$ is responsible for the size of the representation, rather than $I$ itself.

In particular, we observe the following basic relations between BR and SR:

**Lemma 1** (i) If ASC under SR is in complexity class $C$ and $C$ is closed under polynomial-time transformations, then ASC under BR is also in $C$.

(ii) If ASC under BR is hard for class $C$ (under polynomial transformations), then ASC under SR is also hard for $C$.

(iii) If $B_\mathcal{P}$ is polynomial in the size of $\mathcal{P}$ and $\mathcal{U}_\mathcal{P}$, and if ASC under SR is hard for $C$ (under polynomial transformations), then ASC under BR is also hard for $C$.

Note that all complexity classes considered in this paper are closed under polynomial transformations. Items (i) and (ii) hold because SR can be produced from BR in polynomial time (and in logarithmic space). Concerning (iii), BR can be produced from SR only if $B_\mathcal{P}$ is small. We now state the main results for ASC.

**Theorem 1** The complexity of answer set checking in $\text{DL}$ under both the set representation SR and the bitmap representation BR is given by the respective entries in Table 3.

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. One key issue towards the complexity results is the following lemma, which holds for both BR and SR.

**Lemma 2** Given a program $\mathcal{P}$ and a consistent set $I \subseteq B_\mathcal{P}$ of literals, deciding whether $I$ satisfies $\mathcal{P}^I$ is in co-NP.

The result follows easily from the observation that for deciding the complementary problem it suffices to guess a ground substitution $\theta$ and a rule $r \in \mathcal{P}$ and check whether $I$ does not satisfy $(r \theta)^I$, where $r \theta$ denotes the standard way of applying the substitution $\theta$ to $r$. In fact, the problem is also co-NP-hard; and in the propositional case, the problem is polynomial.

In ASC, the above problem is a necessary subtask, and under BR, an interpretation $I'$ compromising a candidate answer set $I$ has always polynomial size, and can intuitively be guessed and checked for this property. However, for ASC under SR, $I'$ might be exponentially larger. Note that this can only occur if the language has weak constraints and if there is a choice for determining the optimal answer set (i.e., multiple regular answer sets may exist; thus Corollary 1 does not apply). This explains the drastic complexity increase by an exponential in these cases.
Example 1 Consider the program $\mathcal{P}_{exp}$:

$$
\text{bit}(0), \text{bit}(1),
$$

$$
\text{w} \lor \text{number}(X_1, \ldots, X_n) \Rightarrow \text{bit}(X_1), \ldots, \text{bit}(X_n).
$$

The ground program $\text{Ground}\mathcal{P}_{exp}$ is clearly exponential in the size of $\mathcal{P}_{exp}$:

$$
\text{bit}(0), \text{bit}(1),
$$

$$
\text{w} \lor \text{number}(0, \ldots, 0) \Rightarrow \text{bit}(0), \ldots, \text{bit}(0),
$$

$$
\vdots
$$

$$
\text{w} \lor \text{number}(1, \ldots, 1) \Rightarrow \text{bit}(1), \ldots, \text{bit}(1).
$$

Checking that an interpretation $I_0 = \{\text{number}(0, \ldots, 0)\}$ does not satisfy $\mathcal{P}_{exp}$ (instability of $I_0$) can be done by guessing a ground rule $\text{r}_{\text{exp}} = \text{w} \lor \text{number}(1, \ldots, 1) \Rightarrow \text{bit}(1), \ldots, \text{bit}(1)$.

Note that $I_2$ is exponential in $\mathcal{P}_{exp}$, but it is part of the problem input. In order to check whether $I_2$ is not an answer set of $\mathcal{P}_{exp}$, either the stability check (NP) succeeds on $I_2$ or the stability check (co-NP) succeeds for an $I_2 \subseteq I_2$. There are exponentially many $I_2$, but the size of each is bounded by the size of $I_2$ for both SR and BR. So in general an NP algorithm which uses an NP oracle can be employed. This justifies checking whether an interpretation is an answer set in $I_2$ for a positive disjunctive program.

Now consider $I_3 = \{w\}$ and let us check whether it is an answer set of $\mathcal{P}_{wexp} = \mathcal{P}_{exp} \cup \{\neg w. [1 : 1]\}$. In order to check the complementary problem, is not sufficient to check instability of $I_3$ (in NP) or instability (in co-NP) for some $I_3 \subseteq I_3$, as before. Now, it also cannot happen that the co-NP check succeeds for some $I_3' \subseteq I_3$ and $H_{\mathcal{P}_{wexp}}(I_3') < H_{\mathcal{P}_{wexp}}(I_3)$, invalidating $I_3$ as an answer set. Indeed, $I_3 = \{\text{number}(0, \ldots, 0), \ldots, \text{number}(1, \ldots, 1)\}$ is such a case (it does not violate any weak constraints, while $I_3$ does). But observe that the size of $I_3'$ is exponentially larger than $I_3$ (and hence exponentially larger than the input, consisting of $I_3$ and $\mathcal{P}_{wexp}$) when SR is used, while with BR both are of equal size. This is the reason for this problem to be hard for co-NEXP$^{NP}$ for SR, but to be located in a lower complexity class for BR.

In the following we shall discuss all results from Theorem 1 in detail. Note that for showing program classes $\text{DL}[L_1] \subseteq \text{DL}[L_2] \subseteq \ldots \subseteq \text{DL}[L_k]$ to be complete for a complexity class $C$, it suffices to prove $C$-hardness for $\text{DL}[L_1]$ and $C$-membership for $\text{DL}[L_k]$. Furthermore, $C$-hardness for normal programs is immediate from $C$-hardness for HCF programs, due to a faithful polynomial-time rewriting of HCF programs to equivalent normal programs (Ben-Eliyahu & Dechter 1994). We will implicitly employ this technique in the remainder of the paper.

ASC under the Set Representation (SR)

The first two results justify all co-NEXP$^{-}$ and co-NEXP$^{NP}$-completeness results in Table 3.

Lemma 3 ASC under SR is in co-NEXP$^{NP}$ for DL programs; it is in co-NEXP for $\text{DL}[\forall, w]$ programs.

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

Lemma 4 ASC under SR is co-NEXP$^{NP}$-hard for $\text{DL}[\forall, w]$ programs and co-NEXP-hard for $\text{DL}[\forall, w]$ 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 ground atom, and consider a program $\mathcal{P}'$, which results from adding $w$ to each head in $\mathcal{P}$, and adding a weak constraint $\neg a. [1 : 1]$. 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 (in the case the Herbrand base is exponential in the size of $\mathcal{P}$ and $U_\mathcal{P}$). 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 this shows co-NEXP$^{NP}$-hardness for positive disjunctive logic programs and co-NEXP-hardness for HCF programs.

We proceed with the $D^P$-entries; the class $D^P$ contains the decision problems whose yes instances are characterized by the conjunction of an NP property and an independent co-NP property.

The next two results, together with Corollary 1, cover all $D^P$-entries in Table 3.

Lemma 5 ASC is in $D^{NP}$ for $\text{DL}[\forall, w]$ programs.

Proof. Given a normal 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 $\mathcal{P}'$, and (ii) $I$ is minimal in satisfying $\mathcal{P}'$. From Lemma 2, we know that (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 $\mathcal{P}'$ 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 have needed both a co-NP- and an NP-test, implying membership in $D^{NP}$.

KR 2004  381
Lemma 6 ASC under SR is $D^P$-hard for DL[] programs.

Proof. The result is easily shown by a reduction from conjunctive query evaluation, which is NP-complete (see (Abiteboul, Hull, & Vianu 1995)): 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 $P = DB_1 \cup DB_2 \cup \{a_1 \leftarrow B_1, a_2 \leftarrow B_2\}$ 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, $P$ 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 $P$ 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 $D^P$-hardness.

Remaining are the $\Pi^P_2$-entries in the third row. Again, we have two results.

Lemma 7 ASC for the $P$ programs is in $\Pi^P_2$.

Proof. We show that the complementary problem is in $\Sigma^P_2$. Let $P$ be a program without weak constraints and $I$ a consistent set of literals. Clearly, $I$ is not an answer set for $P$ iff (i) $I$ does not satisfy $P^I_2$ or (ii) there exists some $I' \subset I$ which satisfies $P$. 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 $NP^{NP} = \Sigma^P_2$, and so is the complementary problem of ASC.

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 $Rules(P)$ exists. But then, $I'$ is not necessarily polynomial in the size of the problem input (i.e., $P$ and $I$) if SR is used. Recall that under BR, which is discussed in the next section, this problem does not occur.

Lemma 8 ASC under SR is $\Pi^P_2$-hard for DL[$\lor$] programs.

Proof. The proof is via a polynomial reduction of the evaluation problem for QBPs of form $\Phi = \forall X \exists Y_1 \cdots \exists Y_k$, where the $c_i$ are clauses over $X \cup Y$. This problem is $\Pi^P_2$-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. In particular, we construct a program $P$ for each QBF $\Phi$ of above form, such that a dedicated set of atoms $B^+$ (see below) is an answer set of $P$ iff $\Phi$ is true. The construction is as follows:

First, set up a disjunctive fact

$$t(x_i) \lor f(x_i), \quad \text{for each } x_i \in X$$

(1)

using $x_i$ as a constant. For each clause $c_i = L_{i,1} \lor \cdots \lor L_{i,3}$, 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 $c_1 = x_1 \lor \neg x_2 \lor y_3$. Then, we introduce $c_1(V)$, where the argument $V$ is reserved for the truth assignments to $y_3$, and define:

$c_1(0) : t(x_1), \quad c_1(1) : t(x_1)$
$c_1(0) : f(x_2), \quad c_1(1) : f(x_2)$
$c_1(1) : f(x_1), t(x_2)$

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 $c_2 = x_2 \lor \neg y_1 \lor \neg y_3$. Here, we introduce $c_2(V_1, V_2)$, and define:

$c_2(0, 0) : t(x_2), \quad c_2(0, 1) : t(x_2)$
$c_2(1, 0) : t(x_2), \quad c_2(1, 1) : t(x_2)$
$c_2(0, 0) : f(x_2), \quad c_2(0, 1) : f(x_2), \quad c_2(1, 1) : f(x_2)$

Now set up a rule which corresponds to evaluating the formula $\exists Y_1 \cdots \exists Y_k$ for a given assignment to $X$:

$$w : c_1(Y_1) \land \cdots \land c_k(Y_k).$$

(2)

where $Y_i, 1 \leq i \leq k$, is a vector representing the variables from $Y$ occurring in $c_i$, put at proper position. In the case above, we have $c_1(Y_3)$ and $c_2(Y_1, Y_3)$.

Let us call the program built so far $P_{QBF}$; it will also be used in some of the subsequent proofs. Note that $P_{QBF}$ is positive, disjunctive, and HCF, as well as polynomial in the size of the underlying QBF. The functioning of $P_{QBF}$ is as follows: The disjunctive clauses (1) generate a truth assignment to $X$, and the remaining clauses check whether $\exists Y_1 \cdots \exists Y_k$ is true under this assignment, deriving $w$ if so.

For proving Lemma 8, we create a maximal interpretation if $w$ holds as follows. Let $B^+$ be the set of all positive literals in $BP_{QBF}$, and add rules

$$p : w, \quad \text{for each ground atom } p \in B^+ \setminus \{w\},$$

(3)

to $P_{QBF}$. Call the resulting program $P$. Note that $P$ is not HCF, and that $BP = BP_{QBF}$ has polynomial size, since the arity of each predicate is at most $3$. If we derive $w$ from $P_{QBF}$, any element from $B^+$ can be derived in $P$. Hence, if for each possible truth assignment to $X$ a truth assignment to $Y$ exists s.t. $c_1 \land \cdots \land c_k$ is true (i.e., $\Phi$ is true), $B^+$ is an answer set of $P$. On the other hand, if a truth assignment to $X$ exists such that no assignment to $Y$ makes $c_1 \land \cdots \land c_k$ true (i.e., $\Phi$ is false), $B^+$ cannot be an answer set of $P$, as there exists a proper subset (not containing $w$) of $B^+$ which is an answer set of $P$. Hence, $B^+$ is an answer set of $P$ iff $\Phi$ is true.

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. Moreover, The Herbrand literal bases of the programs used in the $D^P$-hardness proof of ASC under SR (Lemma 6) and the $\Pi^P_2$-hardness proof of ASC under SR (Lemma 8) have polynomial size in the program 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 4.
It thus remains to verify the results for those fragments where the set representation caused an exponential shift. The following result immediately clarifies the upper bounds for ASC under BR, viz. $\Pi^P_2$ for HCF programs and $\Pi^P_3$ in general.

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

**Proof.** Let $\mathcal{P} \in DL[L]$ 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. The bitmap representation guarantees that the respective guesses are polynomial in size of the problem instance. Since $\Pi^P_{k+1}$ is closed under conjunction, we can combine this into a single $\Pi^P_{k+1}$ test. □

As an immediate consequence, ASC under BR is in $\Pi^P_3$ for DL programs, and in $\Pi^P_2$ for DL[$\forall h, \not \ w$] programs. The subsequent two results provide the matching lower bounds to complete the table entries for BR.

**Lemma 9** ASC is $\Pi^P_2$-hard for DL[$\forall h, \not \ w$] programs.

**Proof.** The proof is by reduction of a QBF of the form $\Phi = \forall X \exists Y (c_1 \land \cdots \land c_k)$. Recall the program $\mathcal{P}_{QBF}$ as defined in the proof of Lemma 8, add a fresh atom $q$ in the head of each rule of $\mathcal{P}_{QBF}$, and finally add the weak constraints $\vdash q. [1 : 1]$ and $\vdash \not \ w. [2 : 1]$. The resulting program $\mathcal{P}$ is HCF (in fact, it is acyclic). We claim that $\tilde{I} = \{q\}$ is the optimal answer set of $\mathcal{P}$ iff $\Phi$ is true. This can be seen as follows. First, $\tilde{I}$ 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, $\tilde{I}$ is the only answer set of Rules($\mathcal{P}$) which contains $q$. The cost of $\tilde{I}$ for $\mathcal{P}$ is 1. By the weak constraints in $\mathcal{P}$, any other answer set $I$ has smaller cost than $\tilde{I}$ iff $w \notin I$. This, however, amounts to the existence of a truth assignment to the variables $X$ such that $\exists Y (c_1 \land \cdots \land c_k)$ is false, i.e., formula $\Phi$ is false. Hence, $\tilde{I}$ is an (optimal) answer set of $\mathcal{P}$ iff $\Phi$ is true. □

**Lemma 10** ASC is $\Pi^P_3$-hard for DL[$\forall, \not \ w$] programs.

**Proof.** Consider an existential QBF $\Phi = \exists X_1 \forall X_2 \exists Y c_1 \land \cdots \land c_k$, take $\mathcal{P}_{QBF}$ from the proof of Lemma 8, but now with $X = X_1 \cup X_2$, and add rules $p \vdash \not \ w$. For each ground atom $p \in B^+ \setminus \{w, t(x_1), f(x_1) \mid x_1 \in X_1\}$, making the program non-HCF, where $B^+$ is defined as in Lemma 8, as well. The resulting program intuitively guesses a truth assignment $\sigma$ for the atoms $X_1$. Then, for each of these truth assignments, the program has a corresponding answer set and exactly behaves like the program in Lemma 8 for $\Phi' = \forall X_2 \exists Y (c_1 \land \cdots \land c_k)\sigma$. 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 $\vdash q. [1 : 1]$ and $\vdash \not \ w. [2 : 1]$. 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. □ We show that $\tilde{I} = \{q\}$ 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 $\Pi^P_3$-complete. Clearly, $\Phi$ is false iff there exists no truth assignment $\sigma$ to $X_1$ such that $\Phi'$ is true. $\tilde{I}$ is the only answer set of Rules($\mathcal{P}$) containing $q$, and it has cost 1. Thus, $\tilde{I}$ 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. □

**Complexity of Bounded Predicate Arities**

If we constrain the programs to have the arities of predicates bounded by some constant, then representations SR and BR of an interpretation $I$ are polynomially intertranslatable. In this case, interpretations (as sets) have size polynomial in the size of the program instance. The following result is obtained from Theorem 1 and the derivation of the results it summarizes.

**Theorem 2** The complexity of ASC under both SR and BR for predicate arities bounded by a constant coincides with the complexity of ASC under BR for arbitrary predicate arities.

The complexity results for brave and cautious reasoning under bounded intensional predicate arities are summarized in Theorem 3.

**Theorem 3** The complexity of brave and cautious reasoning under bounded predicate arities is given by the respective entries in Table 4.

These results show, that if we move from ground (i.e., propositional) programs to non-ground programs but allow only predicates with small arity, the complexity of the language moves up only one level in the polynomial hierarchy.

---

**Table 4:** Complexity of brave and cautious reasoning under bounded predicate arities. All entries are completeness results.

<table>
<thead>
<tr>
<th>Brave / Cautious</th>
<th>{}</th>
<th>{w}</th>
<th>{not}</th>
<th>{not, w}</th>
<th>{not}</th>
<th>{not, w}</th>
</tr>
</thead>
<tbody>
<tr>
<td>{}</td>
<td>$\Delta^P_2$ / $\Pi^P_2$</td>
<td>$\Delta^P_2$</td>
<td>$\Delta^P_2$</td>
<td>$\Delta^P_2$</td>
<td>$\Delta^P_2$</td>
<td>$\Delta^P_2$</td>
</tr>
<tr>
<td>$\forall h$</td>
<td>$\Sigma^P_2 / \Pi^P_2$</td>
<td>$\Sigma^P_2 / \Pi^P_2$</td>
<td>$\Sigma^P_2 / \Pi^P_2$</td>
<td>$\Sigma^P_2 / \Pi^P_2$</td>
<td>$\Sigma^P_2 / \Pi^P_2$</td>
<td>$\Sigma^P_2 / \Pi^P_2$</td>
</tr>
<tr>
<td>$\forall$</td>
<td>$\Sigma^P_3 / \Pi^P_2$</td>
<td>$\Delta^P_3$</td>
<td>$\Sigma^P_3 / \Pi^P_3$</td>
<td>$\Delta^P_3$</td>
<td>$\Sigma^P_3 / \Pi^P_3$</td>
<td>$\Delta^P_3$</td>
</tr>
</tbody>
</table>

* Without constraints and strong negation (= definite Horn) the complexity is NP.
(PH), but not more. Thus, unless we use growing predicates arities, we (most likely) can not encode problems above PH, e.g. PSPACE-complete problems. On the other hand, it means that an exponential-size grounding-at-once can be avoided. Furthermore, a number of the problems can be polynomially mapped to ASP with disjunctive propositional programs (harboring $\Sigma^p_2 / \Pi^p_2$ complexity), avoiding grounding.

We note that the results remain valid if we just restrict the arities of the intensional predicates, i.e., those occurring in the heads of non-facts, and predicates of non-ground atoms in disjunctive facts. Intuitively, any answer set $S$ has then polynomial size modulo a fixed part, while checking rule compliance of a candidate answer set $S$ is co-NP-complete rather than polynomial as in the ground case.

In what follows, we informally summarize some remarks on the results in Table 4, and afterwards give the formal proofs.

The D$^P$ results are explained similarly as those in the case of ASC. The co-NP part is needed to show that no contradiction is derivable (which vanishes for definite Horn programs), while the NP part stems from a foundedness (minimality) check.

For stratified normal programs, we have slightly higher complexity since we must evaluate a sequence of NP problems according to the layers of the program.

In the presence of weak constraints, the upper bounds easily follow from the complexity of ASC, first computing the cost of an optimal answer set in a binary search, and then deciding the problem with a single oracle call.

The only peculiarity in Theorem 3 is for DL[$\forall$], for which brave reasoning is one level higher than cautious reasoning. However, also this is carried over from the propositional case in which a similar gap exists, see Table 2. This gap can explained by the fact that minimalilty is not important for cautious reasoning in this case, while it is for brave reasoning. These results (also $\Pi^P_2$-hardness when negation is involved) can be proved similar to Lemma 10, where $\Pi^P_2$-hardness of ASC for positive disjunctive programs using weak constraints was shown.

We proceed with a more formal elaboration of the results, starting with the D$^P$ and NP entries in Table 4.

**Lemma 11** Brave reasoning is in 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 5. 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 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 D$^P$-hard.

**Proof.** The results are inherited from (bounded) conjunctive queries as used in the proof of Lemma 6. Indeed, consider a conjunctive query $a \leftarrow B$ over a database $DB$. Then $a \leftarrow B$ evaluates to true under $DB$ iff the unique answer set of the definite Horn program $DB \cup \{a \leftarrow B\}$ contains a. For the D$^P$-hardness result, consider two conjunctive queries $a_1 \leftarrow B_1, a_2 \leftarrow B_2$ with $a_1 \neq a_2$, and two databases $DB_1, DB_2$ over disjoint alphabets not containing $a_1$ or $a_2$. Then, $a_1 \leftarrow B_1$ evaluates to false under $DB_1$ and $a_2 \leftarrow B_2$ evaluates to false under $DB_2$ iff $a_1$ is a brave consequence of the (non-definite) Horn program $DB_1 \cup DB_2 \cup \{a_1 \leftarrow B_1, \neg a_2, B_2\}$.

Without weak constraints, complexity of brave (resp. cautious) reasoning has obvious upper bounds of $\Sigma^P_{k+1}$ (resp. $\Pi^P_{k+1}$), if answer set checking is in $\Delta^P_k$. The following results give the matching lower bounds.

**Lemma 13** For DL[$\forall_h$] programs brave reasoning is $\Sigma^P_2$-hard, and cautious reasoning is $\Pi^P_2$-hard.

**Proof.** $\Pi^P_2$-hardness immediately follows from the reduction in the $\Pi^P_2$-hardness proof of Lemma 8: $w$ is a cautious consequence of the program $P$ used iff the formula $\Phi = \forall X \exists Y c_1 \land \cdots \land c_k$ is true. We obtain the dual $\Sigma^P_2$-hardness result for brave reasoning by adding the disjunctive fact $u \lor w$, to $P$, where $u$ is a fresh atom, and asking whether $u$ is a brave consequence of the resulting program; this is the case iff $w$ is not a cautious consequence of the original program.

**Lemma 14** For DL[$\forall$] programs, brave reasoning is $\Sigma^P_3$-hard, and cautious reasoning is $\Pi^P_3$-hard. For DL[$\forall, \not a_k$] programs, cautious reasoning is $\Pi^P_3$-hard.

**Proof.** $\Sigma^P_3$-hardness of brave reasoning follows from the construction in the proof of Lemma 10, where $\Pi^P_3$-hardness of ASC for positive disjunctive programs using weak constraints was shown. In fact, $w$ is a brave consequence of the program there iff $\Phi = \exists X_1 \forall X_2 \exists Y c_1 \land \cdots \land c_k$ is true. Cautious reasoning for this fragment, however, is in $\Pi^P_2$, since to disprove a cautious consequence it is sufficient to find some (not necessarily subset-minimal) interpretation $I$ which satisfies $P$ and does not contain the query; such $I$ can be guessed and checked with an NP oracle in polynomial time.

If negation is involved, we obtain $\Pi^P_3$-hardness of cautious inference by a simple reduction of the complement of brave reasoning of the atom $w$ as above, by adding the stratified rule $w' \leftarrow \neg w$, where $w'$ is a fresh atom, and asking whether $w'$ is a cautious consequence.

**Lemma 15** For DL[$\not a_k, w$] programs, both brave and cautious inference are $\Delta^P_2$-complete, where hardness holds also for DL[$\not a_k$].

**Proof.** Membership holds, since the number of strata is polynomially bounded. We show hardness by a reduction from deciding the last bit of the lexicographic maximum satisfying truth assignment for a CNF $C = c_1 \land \cdots \land c_k$ over atoms $X = \{x_1, \ldots, x_n\}$,
which is \( \Delta^P_3 \)-complete, cf. (Papadimitriou 1994). W.l.o.g., each \( c_i = L_{i,1} \lor L_{i,2} \lor L_{i,3} \) contains three literals and \( C \) is known to be satisfiable.

Let \( \mathcal{P} \) contain facts of ternary predicates describing the satisfying truth assignments for each clause \( c_i \). For example, if \( c_1 = x_1 \lor \neg x_2 \lor x_3 \), we add

\[
\begin{align*}
&c_1(0,0,0), \quad c_1(0,0,1), \quad c_1(0,1,1), \quad c_1(1,0,0), \quad c_1(1,0,1), \quad c_1(1,1,1).
\end{align*}
\]

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

\[
\begin{align*}
&\text{val}_{x_i}(1) \leftarrow c_1(\bar{t}_1), \ldots, c_k(\bar{t}_k), \text{true}(V),
&\text{val}_{x_{i-1}}(V_{i-1}), \ldots, \text{val}_{x_i}(V_1).
\end{align*}
\]

\[
\text{val}_{x_i}(0) \leftarrow \neg \text{val}_{x_i}(1).
\]

where \( \bar{t}_j = V_{i_j}, V_{j_2}, V_{j_3}, 1 \leq j \leq k \), given that the atoms of literal \( L_{i,1} \), \( L_{i,2} \), and \( L_{i,3} \) 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 (locally) stratified. The maximum satisfying truth assignment for \( C \) is computed in the layers of \( \mathcal{P} \), and by \( \text{val}_{x_i}(b_i) \) in the unique answer set \( I \) of \( \mathcal{P} \). At the bottom \( \text{val}_{x_i}(1) \) is derived iff \( C \theta \) for \( \theta = \{ x_i/1 \} \) is satisfiable. Otherwise, \( \text{val}_{x_i}(0) \) is derived. Next, depending on the value of \( \text{val}_{x_i}(1) \), \( \text{val}_{x_i}(1) \) is derived iff \( C \theta \) for \( \theta = \{ x_i/b_1, x_2/1 \} \) is satisfiable, otherwise \( \text{val}_{x_i}(0) \) is derived, and so on.

Thus, \( \text{val}_{x_i}(1) \) is in \( I \) iff the last bit of the maximum satisfying assignment is 1, and \( \text{val}_{x_i}(0) \) is in \( I \) otherwise.

**Lemma 16** For DL\([v, w]\) programs, both inference tasks are \( \Delta^P_3 \)-hard.

**Proof.** Consider the open QBF \( \Phi[X] = \exists y c_1 \land \cdots \land c_k \) with \( X = \{ x_1, \ldots, x_n \} \). Deciding the last bit of the lexicographic maximum assignment to \( x_1, \ldots, x_n \) making \( \Phi[X] \) false is \( \Delta^P_3 \)-complete.

Consider now the program \( \mathcal{P} \) which extends \( \mathcal{P}_{\text{QBF}} \) 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 \( \text{Rules}(\mathcal{P}) \) 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 \( \text{Rules}(\mathcal{P}) \) 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.

**Lemma 17** For DL\([v, w]\) programs, both inference tasks are \( \Delta^P_3 \)-hard.

**Proof.** The proof is similar to the one of Lemma 16; the differences mirror the lifting between the proofs of Lemmas 9 and 10, respectively. In fact, consider the open QBF \( \Phi[X] = \forall x_1 \exists y c_1 \land \cdots \land c_k \) with \( X_1 = \{ x_1, \ldots, x_n \} \). Deciding the last bit of the lexicographic maximum satisfying truth assignment to the atoms \( x_1, \ldots, x_n \) for \( \Phi[X] \) is \( \Delta^P_3 \)-complete. Let \( Q = \text{Rules}(\mathcal{P}) \), where \( \mathcal{P} \) is as in Lemma 10, which is positive and disjunctive, but not HCF. \( Q \) guesses a truth assignment \( \sigma \) for \( X_1 \), and \( w \) is in the corresponding answer set iff \( \Phi[X_1] \sigma \) is true. We then add weak constraints \( \sim w, [n+1] \) and \( \sim f(x_i), [n-i+1] \) for each \( i \in \{1, \ldots, n\} \), giving the highest penalty if \( \Phi[X_1] \sigma \) is false. By a similar argumentation as in the proof of Lemma 16, we get that the optimal answer set of the resulting program corresponds to the maximal truth assignment to variables \( X_1 \) such that \( \Phi[X_1] \) is true. Both brave and cautious reasoning therefore allow to decide the last bit of this assignment. Hence, we derive \( \Delta^P_3 \)-hardness.

**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 representation for interpretations is crucial in terms of ASC complexity. If a set-oriented, explicit enumeration (SR) is chosen, an exponential blowup can be witnessed for programs containing weak constraints and disjunctions or unstratified negation, while with a 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 \( \text{P} \) to \( \Delta^P_4 \) and from \( \text{co-NP} \) to \( \Pi^P_4 \) for program classes without weak constraints but with disjunctions and unstratified negation, respectively, under both SR and BR. For other classes, complexity shifts from \( \text{co-NP} \) to \( \text{co-NEXP} \) and from \( \Pi^P_4 \) to \( \Pi^P_5 \) if \( \text{SR} \) is chosen, which moves from \( \text{co-NP} \) to \( \Pi^P_4 \) and from \( \Pi^P_4 \) to \( \Pi^P_5 \) (and thus remains in the Polynomial Hierarchy) 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 \( \text{EXP}^{	ext{co-NP}} \) to an area from NP to \( \Delta^P_3 \). Since bounding arities is a natural restriction, these results are of high practical interest.

In particular, the results in Table 4 imply that it should be feasible to find methods for non-ground programs 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/Girit, ASSAT, Cmodels), cannot guarantee these resource restrictions, as the ground program may consume exponential space in the worst case.

Top-down algorithms appear to be good candidates for fulfilling these requirements, but so far there is relatively little work on this topic: In (Bonatti 2001) a resolution method for cautious reasoning with DL[not] programs has been presented. Several approaches to top-down derivation for DL[not] programs have been proposed, see e.g. (Yahya 2002) and references therein. Very recently, a method for top-down cautious query answering for DL[not, v] pro-
programs has been described (Johnson 2003). Unfortunately, it is not clear whether the space and time complexities of these approaches stay in polynomial space and time, respectively, whenever predicate arities are bounded. We are not aware of any top-down methods for full DL_[not, ∨] programs or programs containing weak constraints.

Another approach to overcome exponential space requirements could be to perform a focused grounding using the query, in principle “emulating” a top-down derivation. In (Greco 2003) a generalization of the magic sets technique to DL_[∨] 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 the rewritten program.

We believe that our results carry over to other nonmonotonic formalisms, such as default logic, autoepistemic logic, or circumscription, as they are closely related to ASP. However, we leave this issue for future work.

References


