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 & Truszczyński 1991); see (Dantsin *et al.* 2001) for a survey) and the respective results typically show an exponential shift be-

*This work was partially supported by the Austrian Science Fund (FWF) under projects Z29-N04 and P15068-INF as well as the European Commission projects IST-2002-33570 INFOMIX, IST-2001-32429 ICONS, IST-2001-37004 WASP, and the IST-2001-33123 CologNeT Network of Excellence.
Copyright © 2004, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

	$\{\}$	$\{\mathbf{w}\}$	$\{\mathrm{not_s}\}$	$\{\mathrm{not_s}, \mathbf{w}\}$	$\{\mathrm{not}\}$	$\{\mathrm{not}, \mathbf{w}\}$
$\{\}$	P	P	P	P	P	co-NP
$\{\vee_\mathbf{h}\}$	P	co-NP	P	co-NP	P	co-NP
$\{\vee\}$	co-NP	Π_2^P	co-NP	Π_2^P	co-NP	Π_2^P

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

Brave / Cautious	$\{\}$	$\{\mathbf{w}\}$	$\{\mathrm{not_s}\}$	$\{\mathrm{not_s}, \mathbf{w}\}$	$\{\mathrm{not}\}$	$\{\mathrm{not}, \mathbf{w}\}$
$\{\}$	P	P	P	P	NP / co-NP	Δ_2^P
$\{\vee_\mathbf{h}\}$	NP / co-NP	Δ_2^P	NP / co-NP	Δ_2^P	NP / co-NP	Δ_2^P
$\{\vee\}$	Σ_2^P / co-NP	Δ_3^P	Σ_2^P / Π_2^P	Δ_3^P	Σ_2^P / Π_2^P	Δ_3^P

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

tween 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 be partly explained by the fact that, historically, ASC has not been considered as a standard reasoning task. There are, however, at least two reasons why ASC should be considered en par with the other problems mentioned above:

1. In ASP, solutions to a problem are encoded in the answer sets of a corresponding logic program, so it is of natural interest to test whether a given claimed-to-be solution is in fact a proper one.
2. Most ASP systems proceed by generating answer set candidates and subsequently checking these. This includes DLV and GnT (which solve problems above NP where such an approach is natural), as well as ASSAT and Cmodels (which employ a transformation to SAT with solutions not corresponding 1-1 with the answer sets of the original program in general) (Leone *et al.* 2002; Janhunen *et al.* 2000; Lin & Zhao 2002; Lierler & Maratea 2004). ASC emerges as a computational subtask in these systems.

For the propositional case, (Leone *et al.* 2002) gives an overview on the complexity of ASC, which is depicted in Table 1, but to our knowledge the non-ground case has not been studied in depth so far.

This may partly be due to the fact that, in line with the definition of the Answer Set semantics (Gelfond & Lifschitz 1991), 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, even though today's state of the art ASP systems try to keep the grounding as small as possible, it still may cause an exponential blow-up in the worst case.

To fill this gap, in this paper we consider ASC for non-ground programs, and apply our findings to derive novel results for reasoning over non-ground programs. Our main results can be summarized as follows:

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 aforecited 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-)$\mathrm{NEXP}^{\mathrm{NP}}$, 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 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

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

$$a_1 \vee \cdots \vee a_n \text{ :- } b_1, \ldots, b_k, \text{ not } b_{k+1}, \ldots, \text{ not } b_m.$$

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

A rule r is called *fact* if $m = 0$, $n > 0$, in which case the symbol :- 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

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

where $m \geq k \geq 0$ and $b_1, \ldots, b_m$ 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 sets $B^+(wc)$, and $B^-(wc)$, 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 $\mathcal{P}$. $w^{\mathcal{P}}_{max}$ and $l^{\mathcal{P}}_{max}$ denote the maximum weight and maximum level over $WC(\mathcal{P})$, respectively. Programs are normal (resp., definite, disjunctive, positive, Horn) if all of their rules enjoy this property. Horn programs without constraints and strong negation are *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 $r\sigma$ obtained by applying, to each rule and weak constraint $r \in \mathcal{P}$, all possible substitutions σ from the variables in $\mathcal{P}$ to elements of $U_{\mathcal{P}}$. $U_{\mathcal{P}}$ is usually called the *Herbrand Universe* of $\mathcal{P}$ and $B_{\mathcal{P}}$ the *Herbrand Literal Base* of $\mathcal{P}$.

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

not_s:	negation; the program remains stratified;
not:	unrestricted negation;
$\vee_h$:	disjunction; the program remains HCF;
$\vee$:	unrestricted disjunction;
w:	weak constraints.

Recall that stratified negation, not_s, 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 ,$\vee_h$, (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[$\vee_h, \text{not}_s$] contains all HCF stratified programs without weak constraints, and DL $=$ DL[$\vee, \text{not}, w$] 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 $\mathcal{P}$, if each $r \in \mathcal{P}$ is satisfied by I. 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 $B^+(c) \subseteq I$ and $B^-(c) \cap I = \emptyset$; it is satisfied otherwise.

Recall that for $\mathcal{P} \in$ DL[$\vee$, not], a consistent set $I \subseteq B_{\mathcal{P}}$ is an *answer set*[1] iff it is a subset-minimal set satisfying the *Gelfond-Lifschitz reduct*

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

For $\mathcal{P} \in$ DL[$\vee$, not, w], a consistent set $I \subseteq B_{\mathcal{P}}$ 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 the penalization $H^{\mathcal{P}}(I)$ for weak constraint violation is defined as follows:

$$\begin{aligned} H^{\mathcal{P}}(I) &= \textstyle\sum_{i=1}^{l^{\mathcal{P}}_{max}} \left(f_{\mathcal{P}}(i) \cdot \sum_{w \in N^{\mathcal{P}}_i(I)} weight(w)\right) \\ f_{\mathcal{P}}(1) &= 1, \text{ and} \\ f_{\mathcal{P}}(n) &= f_{\mathcal{P}}(n-1) \cdot |WC(\mathcal{P})| \cdot w^{\mathcal{P}}_{max} + 1 \text{ for } n > 1. \end{aligned}$$

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

[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.

SR / BR	{}	{w}	{not$_s$}	{not$_s$, w}	{not}	{not, w}
{}	D^P	D^P	D^P	D^P	D^P	co-NEXP / Π_2^P
{$\vee_h$}	D^P	co-NEXP / Π_2^P	D^P	co-NEXP / Π_2^P	D^P	co-NEXP / Π_2^P
{$\vee$}	Π_2^P	co-$\mathrm{NEXP}^{\mathrm{NP}}$ / Π_3^P	Π_2^P	co-$\mathrm{NEXP}^{\mathrm{NP}}$ / Π_3^P	Π_2^P	co-$\mathrm{NEXP}^{\mathrm{NP}}$ / Π_3^P

Table 3: Complexity of ASC for DL. All entries are completeness results.

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

Hence:

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

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 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, Δ_2^P to $\mathrm{EXP}^{\mathrm{NP}}$, Σ_2^P to $\mathrm{NEXP}^{\mathrm{NP}}$, 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 $U_\mathcal{P}$, 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* 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 θ 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 θ 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 $\mathcal{P}_{exp}$:

$$bit(0). \quad bit(1).$$
$$w \vee number(X_1, \ldots, X_n) \text{:-} bit(X_1), \ldots, bit(X_n).$$

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

$$bit(0). \quad bit(1).$$
$$w \vee number(0, \ldots, 0) \text{:-} bit(0), \ldots, bit(0).$$
$$\vdots$$
$$w \vee number(1, \ldots, 1) \text{:-} bit(1), \ldots, bit(1).$$

Checking that an interpretation $I_0 = \{number(0, \ldots, 0)\}$ does not satisfy $\mathcal{P}_{exp}^{I_0}$ (instability of I_0) can be done by guessing a ground rule $r_{vio} = w \vee number(1, \ldots, 1) \text{:-} bit(1), \ldots, bit(1).$ (among exponentially many) such that I_0 does not satisfy $r_{vio}^{I_0}$. One can see that in general this task is in NP for both SR and BR, and therefore the complementary stability check is in co-NP.

Now consider a different interpretation $I_2 = \{w, number(0, \ldots, 0), \ldots, number(1, \ldots, 1)\}$. 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 instability 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 that checking whether an interpretation is an answer set is in Π_2^P for a positive disjunctive program.

Now consider $I_3 = \{w\}$ and let us check whether it is an answer set of $\mathcal{P}_{wexp} = \mathcal{P}_{exp} \cup \{\text{:}\sim\ w.\ [1:1]\}$. 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 can also happen that the co-NP check succeeds for some $I_3'' \not\subseteq I_3$ and $H^{\mathcal{P}_{wexp}}(I_3'') < H^{\mathcal{P}_{wexp}}(I_3)$, invalidating I_3 as an answer set. Indeed, $I_3'' = \{number(0, \ldots, 0), \ldots, number(1, \ldots, 1)\}$ 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$^{\mathrm{NP}}$ 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 DL$[L_1] \subseteq$ DL$[L_2] \subseteq \ldots \subseteq$ DL$[L_k]$ to be complete for a complexity class C, it suffices to prove C-hardness for DL$[L_1]$ and C-membership for 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$^{\mathrm{NP}}$-completeness results in Table 3.

Lemma 3 *ASC under SR is in* co-NEXP$^{\mathrm{NP}}$ *for* DL *programs; it is in* co-NEXP *for* DL$[\vee_h, \mathrm{not}, w]$ *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$^{\mathrm{NP}}$*-hard for* DL$[\vee, w]$ *programs and* co-NEXP*-hard for* DL$[\vee_h, w]$ *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 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 $\text{:}\sim \mathrm{not}\ a.\ [1:1]$. 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 (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$^{\mathrm{NP}}$-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^P *for* DL$[\mathrm{not}]$ *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 $\mathcal{P}^I$, and (ii) I is minimal in satisfying $\mathcal{P}^I$. 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 $\mathcal{P}^I$ 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 have needed both a co-NP- and an NP-test, implying membership in D^P. □

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 $a \text{:-} B$ 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 $a_1 \text{:-} B_1$ and $a_2 \text{:-} B_2$, 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 $a_1 \text{:-} B_1$ evaluates to true under DB_1 and $a_2 \text{:-} B_2$ evaluates to false under DB_2; this implies D^P-hardness. □

Remaining are the Π_2^P-entries in the third row. Again, we have two results.

Lemma 7 *ASC for* DL[$\vee$, not] *programs is in* Π_2^P.

Proof. We show that the complementary problem is in Σ_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 $\mathrm{NP}^{\mathrm{NP}} = \Sigma_2^P$, 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 $Rules(\mathcal{P})$ 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. Recall that under BR, which is discussed in the next section, this problem does not occur.

Lemma 8 *ASC under SR is* Π_2^P*-hard for* DL[$\vee$] *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 Π_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. In particular, we construct a program $\mathcal{P}$ for each QBF Φ of above form, such that a dedicated set of atoms B^+ (see below) is an answer set of $\mathcal{P}$ iff Φ is true. The construction is as follows:

First, set up a disjunctive fact

$$t(x_i) \vee f(x_i). \qquad \text{for each } x_i \in X \tag{1}$$

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:

$$\begin{array}{ll} c_1(0) \text{:-} t(x_1). & c_1(1) \text{:-} t(x_1). \\ c_1(0) \text{:-} f(x_2). & c_1(1) \text{:-} f(x_2). \\ c_1(1) \text{:-} f(x_1), t(x_2). & \end{array}$$

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:

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

Now set up a rule which corresponds to evaluating the formula $\exists Y c_1 \wedge \cdots \wedge c_k$ for a given assignment to X:

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

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

Let us call the program built so far $\mathcal{P}_{QBF}$; it will also be used in some of the subsequent proofs. 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, 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 $B_{\mathcal{P}_{QBF}}$, and add rules

$$p \text{:-} w. \quad \text{for each ground atom } p \in B^+ \setminus \{w\}, \tag{3}$$

to $\mathcal{P}_{QBF}$. Call the resulting program $\mathcal{P}$. Note that $\mathcal{P}$ is not HCF, and that $B_{\mathcal{P}} = B_{\mathcal{P}_{QBF}}$ has polynomial size, since the arity of each predicate is at most 3. If we derive w from $\mathcal{P}_{QBF}$, any element from B^+ can be derived in $\mathcal{P}$. Hence, if for each possible truth assignment to X a truth assignment to Y exists s.t. $c_1 \wedge \cdots \wedge c_k$ is true (i.e., Φ is true), B^+ is an answer set of $\mathcal{P}$. On the other hand, if a truth assignment to X exists such that no assignment to Y makes $c_1 \wedge \cdots \wedge c_k$ true (i.e., Φ is false), B^+ cannot be an answer set of $\mathcal{P}$, as there exists a proper subset (not containing w) of B^+ which is an answer set of $\mathcal{P}$. Hence, B^+ is an answer set of $\mathcal{P}$ iff Φ 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 Π_2^P-hardness proof of ASC under SR (Lemma 8) 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 4.

Brave / Cautious	$\{\}$	$\{\mathbf{w}\}$	$\{\mathrm{not_s}\}$	$\{\mathrm{not_s}, \mathbf{w}\}$	$\{\mathrm{not}\}$	$\{\mathrm{not}, \mathbf{w}\}$
$\{\}$	$\mathrm{D}^{P\ *}$ / NP	$\mathrm{D}^{P\ *}$ / NP	Δ_2^P	Δ_2^P	Σ_2^P / Π_2^P	Δ_3^P
$\{\vee_\mathbf{h}\}$	Σ_2^P / Π_2^P	Δ_3^P	Σ_2^P / Π_2^P	Δ_3^P	Σ_2^P / Π_2^P	Δ_3^P
$\{\vee\}$	Σ_3^P / Π_2^P	Δ_4^P	Σ_3^P / Π_3^P	Δ_4^P	Σ_3^P / Π_3^P	Δ_4^P

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

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

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. Π_2^P for HCF programs and Π_3^P in general.

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

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 Π_{k+1}^P is closed under conjunction, we can combine this into a single Π_{k+1}^P test. □

As an immediate consequence, ASC under BR is in Π_3^P for DL programs, and in Π_2^P for DL[$\vee_h, \mathrm{not}, w$] programs. The subsequent two results provide the matching lower bounds to complete the table entries for BR.

Lemma 9 *ASC is* Π_2^P*-hard for* DL[$\vee_h, \mathrm{not}, w$] *programs.*

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 8, 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 Φ 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 Φ is false. Hence, $\hat{I}$ is an (optimal) answer set of $\mathcal{P}$ iff Φ is true. □

Lemma 10 *ASC is* Π_3^P*-hard for* DL[$\vee, w$] *programs.*

Proof. Consider an existential QBF $\Phi = \exists X_1 \forall X_2 \exists Y c_1 \wedge \cdots \wedge c_k$, take $\mathcal{P}_{QBF}$ from the proof of Lemma 8, but now with $X = X_1 \cup X_2$, and add rules $p \text{:-} w.$ for each ground atom $p \in B^+ \setminus \{w, t(x_i), f(x_i) \mid x_i \in X_1\}$, making the program non-HCF, where B^+ is defined as in Lemma 8, as well. The resulting program intuitively guesses a truth assignment σ 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 $\Phi' = \forall X_2 \exists Y (c_1 \wedge \cdots \wedge c_k)\sigma$. In particular, w is in an answer set iff Φ' 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 Φ. We remark that $\mathcal{P}$ is a positive program, since negation occurs only in the weak constraints.[2] We show that $\hat{I} = \{q\}$ is an answer set of $\mathcal{P}$ iff Φ is false. This proves the claim since the evaluation problem for QBFs of form Φ is Σ_3^P-complete. Clearly, Φ is false iff there exists no truth assignment σ to X_1 such that Φ' is true. $\hat{I}$ 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 σ to X_1 such that Φ' is true, i.e. iff Φ 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 problem 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*

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

(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 Σ_2^P / Π_2^P 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[$\vee$], 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 minimality is not important for cautious reasoning in this case, while it is for brave reasoning. These results (also Π_3^P-hardness when negation is involved) can be proved similar to Lemma 10, where Π_3^P-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 $a \text{:-} B$ over a database DB. Then $a \text{:-} B$ evaluates to true under DB iff the *unique* answer set of the definite Horn program $DB \cup \{a \text{:-} B.\}$ contains a. For the D^P-hardness result, consider two conjunctive queries $a_1 \text{:-} B_1$, $a_2 \text{:-} B_2$ with $a_1 \neq a_2$, and two databases DB_1, DB_2 over disjoint alphabets not containing a_1 or a_2. Then, $a_1 \text{:-} B_1$ evaluates to true under DB_1 and $a_2 \text{:-} B_2$ evaluates to false under DB_2 iff a_1 is a brave consequence of the (non-definite) Horn program $DB_1 \cup DB_2 \cup \{a_1 \text{:-} B_1.\ \ \text{:-} a_2, B_2.\}$. □

Without weak constraints, complexity of brave (resp. cautious) reasoning has obvious upper bounds of Σ_{k+1}^P (resp. Π_{k+1}^P), if answer set checking is in Δ_{k+1}^P. The following results give the matching lower bounds.

Lemma 13 *For* DL[$\vee_h$] *programs brave reasoning is* Σ_2^P*-hard, and cautious reasoning is* Π_2^P*-hard.*

Proof. Π_2^P-hardness immediately follows from the reduction in the Π_2^P-hardness proof of Lemma 8: w is a cautious consequence of the program $\mathcal{P}$ used iff the formula $\Phi = \forall X \exists Y\, c_1 \wedge \cdots \wedge c_k$ is true. We obtain the dual Σ_2^P-hardness result for brave reasoning by adding the disjunctive fact $u \vee w.$ to $\mathcal{P}$, 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[$\vee$] *programs, brave reasoning is* Σ_3^P*-hard, and cautious reasoning is* Π_2^P*-hard. For* DL[$\vee, \mathrm{not}_s$] *programs, cautious reasoning is* Π_3^P*-hard.*

Proof. Σ_3^P-hardness of brave reasoning follows from the construction in the proof of Lemma 10, where Π_3^P-hardness of ASC for positive disjunctive programs using weak constraints was shown. In fact, w is a brave consequence of the program there iff $\Phi = \exists X_1 \forall X_2 \exists Y\, c_1 \wedge \cdots \wedge c_k$ is true. Cautious reasoning for this fragment, however, is in Π_2^P, since to disprove a cautious consequence it is sufficient to find some (not necessarily subset-minimal) interpretation I which satisfies $\mathcal{P}$ 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 Π_3^P-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 $w' \text{:-} \mathrm{not}\ w.$, where w' is a fresh atom, and asking whether w' is a cautious consequence. □

Lemma 15 *For* DL[not_s, w] *programs, both brave and cautious inference are* Δ_2^P*-complete, where hardness holds also for* DL[not_s].

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 $C = c_1 \wedge \cdots \wedge c_k$ over atoms $X = \{x_1, \ldots, x_n\}$,

which is Δ_2^P-complete, cf. (Papadimitriou 1994). W.l.o.g., each $c_i = L_{i,1} \vee L_{i,2} \vee L_{i,3}$ 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 $c_1 = x_1 \vee \neg x_2 \vee x_3$, we add

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

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 (locally) stratified. The maximum satisfying truth assignment for C is computed in the layers of $\mathcal{P}$, and encoded by $val_{x_i}(b_i)$ 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. □

Lemma 16 *For* DL$[\vee_h, w]$ *programs, both inference tasks are* Δ_3^P*-hard.*

Proof. Consider the open QBF $\Phi[X] = \exists Y\, c_1 \wedge \cdots \wedge c_k$, with $X = \{x_1, \ldots, x_n\}$. Deciding the last bit of the lexicographic maximum assignment to the atoms $x_1, \ldots, x_n$ making $\Phi[X]$ false is Δ_3^P-complete.

Consider now 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. □

Lemma 17 *For* DL$[\vee, w]$ *programs, both inference tasks are* Δ_4^P*-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 $\Phi[X_1] = \forall X_2 \exists Y\, c_1 \wedge \cdots \wedge c_k$ 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 $\Phi[X_1]$ is Δ_4^P-complete. Let $\mathcal{Q} = Rules(\mathcal{P})$, where $\mathcal{P}$ is as in Lemma 10, which is positive and disjunctive, but not HCF. $\mathcal{Q}$ guesses a truth assignment σ for X_1, and w is in the corresponding answer set iff $\Phi[X_1]\sigma$ is true. We then add weak constraints $:\sim \text{not}\, w.\ [:\ n+1]$ 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 Δ_4^P-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 P to D^P and from co-NP to Π_2^P for program classes without weak constraints and with weak constraints but without disjunctions and unstratified negation, respectively, under both SR and BR. For other classes, complexity shifts from co-NP to co-NEXP and from Π_2^P to $\mathrm{co\text{-}NEXP}^{\mathrm{NP}}$ if SR is chosen, while it moves from co-NP to Π_2^P and from Π_2^P to Π_3^P (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 $\mathrm{EXP}^{\Sigma_2^P}$ to an area from NP to Δ_4^P. 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 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/GnT, 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$[\vee]$ programs have been proposed, see e.g. (Yahya 2002) and references therein. Very recently, a method for top-down cautious query answering for DL$[\mathrm{not}_s, \vee]$ pro-

grams 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

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

Anger, C.; Konczak, K.; and Linke, T. 2001. NoMoRe: A System for Non-Monotonic Reasoning. In Eiter, T.; Faber, W.; and Truszczyński, M., eds., *Logic Programming and Nonmonotonic Reasoning — 6th International Conference, LPNMR'01, Vienna, Austria, September 2001, Proceedings*, number 2173 in Lecture Notes in AI (LNAI), 406–410. Springer Verlag.

Apt, K. R.; Blair, H. A.; and Walker, A. 1988. Towards a Theory of Declarative Knowledge. In Minker, J., ed., *Foundations of Deductive Databases and Logic Programming*. Washington DC: Morgan Kaufmann Publishers, Inc. 89–148.

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

Ben-Eliyahu-Zohary, R., and Palopoli, L. 1997. Reasoning with Minimal Models: Efficient Algorithms and Applications. *Artificial Intelligence* 96:421–449.

Bonatti, P. A. 2001. Resolution for Skeptical Stable Model Semantics. *Journal of Automated Reasoning* 27(4):391–421.

Buccafurri, F.; Leone, N.; and Rullo, P. 2000. Enhancing Disjunctive Datalog by Constraints. *IEEE Transactions on Knowledge and Data Engineering* 12(5):845–860.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and Expressive Power of Logic Programming. *ACM Computing Surveys* 33(3):374–425.

Eiter, T., and Gottlob, G. 1995. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Annals of Mathematics and Artificial Intelligence* 15(3/4):289–323.

Eiter, T.; Gottlob, G.; and Mannila, H. 1997. Disjunctive Datalog. *ACM Transactions on Database Systems* 22(3):364–418.

Eiter, T.; Leone, N.; and Saccà, D. 1998. Expressive Power and Complexity of Partial Models for Disjunctive Deductive Databases. *Theoretical Computer Science* 206(1–2):181–218.

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

Gottlob, G.; Leone, N.; and Veith, H. 1999. Succinctness as a Source of Expression Complexity. *Annals of Pure and Applied Logic* 97(1–3):231–260.

Greco, S. 1999. Optimization of Disjunction Queries. In Schreye, D. D., ed., *Proceedings of the 16th International Conference on Logic Programming (ICLP'99)*, 441–455. Las Cruces, New Mexico, USA: The MIT Press.

Greco, S. 2003. Binding Propagation Techniques for the Optimization of Bound Disjunctive Queries. *IEEE Transactions on Knowledge and Data Engineering* 15(2):368–385. Extended Abstract appeared as (Greco 1999).

Janhunen, T.; Niemelä, I.; Simons, P.; and You, J.-H. 2000. Partiality and Disjunctions in Stable Model Semantics. In Cohn, A. G.; Giunchiglia, F.; and Selman, B., eds., *Proceedings of the Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2000), April 12-15, Breckenridge, Colorado, USA*, 411–419. Morgan Kaufmann Publishers, Inc.

Johnson, D. S. 1990. A Catalog of Complexity Classes. In van Leeuwen, J., ed., *Handbook of Theoretical Computer Science*, volume A. Elsevier Science Pub. chapter 2.

Johnson, C. A. 2003. Computing only minimal answers in disjunctive deductive databases. Technical Report cs.LO/0305007, arXiv.org.

Leone, N.; Pfeifer, G.; Faber, W.; Eiter, T.; Gottlob, G.; Perri, S.; and Scarcello, F. 2002. The DLV System for Knowledge Representation and Reasoning. Technical Report cs.AI/0211004, arXiv.org. To appear in ACM TOCL.

Lierler, Y., and Maratea, M. 2004. Cmodels-2: Sat-based answer set solver enhanced to non-tight programs. In Lifschitz, V., and Niemelä, I., eds., *Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-7)*, number 2923 in Lecture Notes in Computer Science, 346–350. Fort Lauderdale, Florida, USA: Springer.

Lin, F., and Zhao, Y. 2002. ASSAT: Computing Answer Sets of a Logic Program by SAT Solvers. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, 112–117. Edmonton, Alberta, Canada: AAAI Press / MIT Press.

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

Nicolas, P.; Saubion, F.; and Stéphan, I. 2002. Answer Set Programming by Ant Colony Optimization. In Flesca, S.; Greco, S.; Ianni, G.; and Leone, N., eds., *Proceedings of the 8th European Conference on Logics in Artificial Intelligence (JELIA 2002)*, number 2424 in Lecture Notes in Computer Science, 481–492. Cosenza, Italy: Springer.

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

Provetti, A., and Son, T. C., eds. 2001. *Proceedings AAAI 2001 Spring Symposium on Answer Set Programming: Towards Efficient and Scalable Knowledge Representation and Reasoning*. Stanford, CA: AAAI Press.

Przymusinski, T. C. 1988. On the Declarative Semantics of Deductive Databases and Logic Programs. In Minker, J., ed., *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann Publishers, Inc. 193–216.

Simons, P.; Niemelä, I.; and Soininen, T. 2002. Extending and Implementing the Stable Model Semantics. *Artificial Intelligence* 138:181–234.

Swift, T. 2004. Deduction in Ontologies via ASP. In Lifschitz, V., and Niemelä, I., eds., *Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-7)*, number 2923 in Lecture Notes in Computer Science, 275–288. Fort Lauderdale, Florida, USA: Springer.

Vardi, M. 1995. On the Complexity of Bounded-Variable Queries. In *Proceedings PODS-95*, 266–276.

Yahya, A. H. 2002. Duality for Goal-Driven Query Processing in Disjunctive Deductive Databases. *Journal of Automated Reasoning* 28(1):1–34.