Complexity of Rule Redundancy in Non-ground Answer-Set Programming over Finite Domains*

Michael Fink, Reinhard Pichler, Hans Tompits, and Stefan Woltran

Institut für Informationssysteme, Technische Universität Wien,
Favoritenstraße 9-11, A-1040 Vienna, Austria
{fink,tompits,stefan}@kr.tuwien.ac.at
pichler@dbai.tuwien.ac.at

Abstract. Recent research in answer-set programming (ASP) is concerned with the problem of finding faithful transformations of logic programs under the stable semantics. This is in particular relevant in practice when programs with variables are considered, where such transformations play a basic role in (offline) simplifications of logic programs. So far, such transformations of non-ground programs have been considered under the implicit assumption that the domain (i.e., the set of constants of the underlying language) is always suitably extensible. However, this may not be a desired scenario, e.g., if one needs to deal with a fixed number of objects. In this paper, we investigate how an explicit restriction of the domain influences the applicability of program transformations and we study in detail computational aspects for the concepts of tautological rules and rule subsumption. More precisely, we provide a full picture of the complexity to decide whether a non-ground rule is tautological or subsumed by another rule under several restrictions.

1 Introduction

Answer-set programming (ASP) has emerged as an important paradigm for declarative problem solving, and provides a host for many different application domains on the basis of nonmonotonic logic programs. The increasing popularity of ASP has also raised interest in the question of equivalence between programs [1,2], which is relevant concerning formal underpinnings for program optimization, where equivalence-preserving modifications are of primary interest; in particular, rewriting rules which allow to perform a local change in a program are important. Many such rules have been considered in the propositional setting [3,4,5,6], but just recently have they been extended to the practicably important case of non-ground programs [7].

In the latter work, a countable domain of constants is assumed, and although this is a reasonable assumption for many scenarios and also common in database theory, it is sometimes more desirable to consider the underlying language in a more restricted way, assuming only a finite, possibly fixed set of constants. While such a finite set comes for free in computing answer-sets of a (complete) program via its active domain, i.e., the set of constants occurring in the program under consideration, this is not the case

* This work was partially supported by the Austrian Science Fund (FWF) under project P18019.

C. Baral, G. Brewka, and J. Schlipf (Eds.): LPNMR 2007, LNAI 4483, pp. 123–135, 2007.
© Springer-Verlag Berlin Heidelberg 2007

for program replacements, which should be applicable in a local sense, for instance to program parts. To this end, such replacements have to take the underlying global domain into account, rather than the active domain of a given program.

In this paper, we consider two important replacements in non-ground answer-set programming under this point of view and analyze their complexity. Intuitively, one would expect that the complexity of a problem decreases as the domain under consideration decreases. In particular, one might hope to get more favorable complexity results when a finite domain is considered rather than a countable domain. On the one hand, Eiter *et al.* [1] show that the restriction to finite domains turns the, in general, undecidable problem of uniform equivalence between logic programs into a decidable one. On the other hand, there is a related problem in the literature where the complexity increases when the domain is restricted: Lassez and Marriot [8] identify so-called *implicit generalizations* as a formal basis of machine learning from counter examples. One of the main problems studied there is the following: Given an atom A and a set $\{B_1, \ldots, B_n\}$ of atoms over some domain $\mathcal{C}$, is every ground instance of A also a ground instance of some B_i? Lassez and Marriot [8] show that this problem is tractable in case of a countably infinite set of constants. This is in contrast to the case of a finite domain, where this problem becomes coNP-complete [9,10]. Now the question naturally arises whether this somewhat counter-intuitive effect of an increased complexity in case of a decreased size of domain also holds for replacements in non-ground answer-set programming. We show that this is indeed the case. In particular, our contributions are as follows, assuming the restriction to a finite domain:

- We show that the detection of tautological rules is NP-complete; and that hardness remains even for some restrictions on the syntax of rules.
- We show that the problem of deciding rule subsumption becomes Π_2^P-complete, and again hardness holds also under several restrictions.

These two main results reveal that complexity increases when we restrict our attention to finite domains, since the detection of tautological rules is tractable under countably infinite domains and rule subsumption is only NP-complete in this setting [7]. However, we also provide results where the problems under consideration are tractable under finite domains as well. In particular, we show that

- detecting tautological Horn rules remains tractable if the maximal arity of predicates is fixed by some constant, and
- detecting tautological rules, as well as rule subsumption, remains tractable in case the number of variables occurring in the involved rules is fixed by some constant.

2 Preliminaries

Our objects of interest are disjunctive logic programs formulated in a language $\mathcal{L}$ over a finite set $\mathcal{A}$ of *predicate symbols*, a finite set $\mathcal{V}$ of *variables*, and a set $\mathcal{C}$ of *constants* (also called the *domain*), which may be either finite or countably infinite.

An *atom* (over $\mathcal{L}$) is an expression of the form $p(t_1,\ldots,t_n)$, where p is a predicate symbol from $\mathcal{A}$ of arity $ar(p) = n$ and $t_i \in \mathcal{C} \cup \mathcal{V}$, for $1 \leq i \leq n$. A (*disjunctive*) *rule*

(over $\mathcal{L}$), r, is an ordered pair of the form

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

where $a_1, \ldots, a_n, b_1, \ldots, b_m$ are atoms (with $n \geq 0$, $m \geq k \geq 0$, and $n+m > 0$), and "not" denotes *default negation*. The *head* of r is $H(r) = \{a_1, \ldots, a_n\}$, and the *body* of r is $B(r) = \{b_1, \ldots, b_k, \text{ not } b_{k+1}, \ldots, \text{ not } b_m\}$. We also define $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$. We call r *positive* if $k = m$, and *Horn* if r is positive and $n = 1$. Furthermore, r is a *fact* if $m = 0$ and $n = 1$ (in which case "$\leftarrow$" is usually omitted). As well, r is *safe* if each variable occurring in $H(r) \cup B^-(r)$ also occurs in $B^+(r)$. By a *program* (over $\mathcal{L}$) we understand a finite set of safe rules (over $\mathcal{L}$).

Let ε be an atom, a rule, or a program. The set of variables occurring in ε is denoted by $\mathcal{V}_\varepsilon$, and ε is called *ground* if $\mathcal{V}_\varepsilon = \emptyset$. Similarly, we write $\mathcal{C}_\varepsilon$ to refer to the set of constants occurring in ε and $\mathcal{A}_\varepsilon$ to refer to the set of predicates occurring in ε. Furthermore, for a set $C \subseteq \mathcal{C}$ of constants, we write $B_{\mathcal{A},C}$ to denote the set of all ground atoms constructible from the predicate symbols from $\mathcal{A}$ and the constants from C. Moreover, for a set A of predicates, $ar_{\max}(A) = \max\{ar(p) \mid p \in A\}$.

Given a rule r and some $C \subseteq \mathcal{C}$, we define $grd(r, C)$ as the set of all rules $r\vartheta$ obtained from r by all possible substitutions $\vartheta : \mathcal{V}_r \to C$. Moreover, for any program P, the *grounding of P with respect to C* is given by $grd(P, C) = \bigcup_{r \in P} grd(r, C)$. The program $grd(P)$ is $grd(P, \mathcal{C}_P)$ for $\mathcal{C}_P \neq \emptyset$, and $grd(P, \{c\})$ otherwise, where c is an arbitrary element from $\mathcal{C}$.

By an *interpretation* (over $\mathcal{L}$) we understand a set of ground atoms (over $\mathcal{L}$). A ground rule r is *satisfied* by an interpretation I, symbolically $I \models r$, iff $H(r) \cap I \neq \emptyset$ whenever $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$. I satisfies a ground program P, symbolically $I \models P$, iff $I \models r$, for each $r \in P$. The *Gelfond-Lifschitz reduct* [11] of a ground program P with respect to an interpretation I is given by $P^I = \{H(r) \leftarrow B^+(r) \mid r \in P,\ I \cap B^-(r) = \emptyset\}$. An interpretation I is an *answer set* of P iff I is a minimal model of $grd(P)^I$. $\mathcal{AS}(P)$ denotes the set of all answer sets of a program P.

Programs P_1 and P_2 are called (*ordinarily*) *equivalent* iff $\mathcal{AS}(P_1) = \mathcal{AS}(P_2)$. Furthermore, P_1 and P_2 are *strongly equivalent* (*in* $\mathcal{L}$), in symbols $P_1 \equiv_s P_2$, iff, for each set S of rules, $\mathcal{AS}(P_1 \cup S) = \mathcal{AS}(P_2 \cup S)$. $(J, I)_C$ is an *SE-model* [1] of a program P iff (i) $J \subseteq I$, (ii) $I \models grd(P, C)$, and (iii) $J \models grd(P, C)^I$. We define $SE_C(P) = \{(J, I)_C \mid (J, I)_C \text{ is an SE-model of } P\}$, for a given $C \subseteq \mathcal{C}$, and $SE(P) = \bigcup_{C \subseteq \mathcal{C}} SE_C(P)$. For all programs P_1 and P_2, the following three conditions are equivalent [1]: (i) $P_1 \equiv_s P_2$, (ii) $SE(P_1) = SE(P_2)$, and (iii) $SE_C(P_1) = SE_C(P_2)$, for every finite $C \subseteq \mathcal{C}$.

Deciding strong equivalence is co-NEXPTIME-complete both for languages over a finite domain as well as for languages over an infinite domain [1]. In what follows, we assume familiarity with the basic complexity classes P, NP, coNP, Δ_2^P, Σ_2^P, and Π_2^P from the literature (cf., e.g., Garey and Johnson [12] for an overview).

3 Tautological Rules

In this section, we syntactically characterize rules which can be deleted in any program (over a given language $\mathcal{L}$), i.e., which are *tautological*. Following Eiter *et al.* [7], let us define

$$\Theta = \{r \mid B^+(r) \cap (H(r) \cup B^-(r)) \neq \emptyset\} \text{ and}$$
$$\Xi^{\mathcal{C}} = \{r \mid \text{ for each } \vartheta : \mathcal{V}_r \to \mathcal{C}, B^+(r\vartheta) \cap (H(r\vartheta) \cup B^-(r\vartheta)) \neq \emptyset\}.$$

Note that the former set does not explicitly refer to the domain of the underlying language. The following proposition rephrases results by Eiter *et al.* [7].

Proposition 1. *Let $\mathcal{L}$ be a language over an infinite domain $\mathcal{C}$ and r a rule. Then, the following conditions are equivalent: (i) $P \equiv_{\mathrm{s}} P \setminus \{r\}$, for each program P over $\mathcal{L}$; (ii) $r \in \Theta$; and (iii) $r \in \Xi^{\mathcal{C}}$.*

In other words, both sets, Θ and $\Xi^{\mathcal{C}}$, contain exactly the same rules in case the domain $\mathcal{C}$ of the underlying language $\mathcal{L}$ is infinite. Moreover, they capture *all* rules which can be faithfully removed in programs over $\mathcal{L}$. Also observe that both sets are equal if only ground rules are taken into account.

Deciding $r \in \Theta$ is an easy syntactic check. Thus, in case the underlying language is given over an infinite domain, the problem of recognizing exactly those rules which can be deleted in any program is an easy task. In particular this test is not harder than in the ground case (which was fully established by Inoue and Sakama [5] and by Lin and Chen [6]), and can be done in linear time.

The situation differs, however, if we restrict our attention to a finite domain. Consider $r : h(1) \vee h(0) \leftarrow h(X)$, which is obviously not contained in Θ. However, under the binary domain $\mathcal{C} = \{0, 1\}$, we have $r \in \Xi^{\mathcal{C}}$. Observe that each grounding $r\vartheta$ with $\vartheta : \mathcal{V}_r \to \mathcal{C}$ yields a tautological (ground) rule contained in Θ. As shown below, each $r \in \Xi^{\mathcal{C}}$ can be faithfully removed from a program. Thus, in the setting of a finite domain $\mathcal{C}$, we have, for each rule r, that $r \in \Theta$ implies $r \in \Xi^{\mathcal{C}}$, but not vice versa.

In the remainder of this section, we assume $\mathcal{L}$ to be given over a finite domain $\mathcal{C}$. Our first result shows that $\Xi^{\mathcal{C}}$ contains all rules which can be faithfully removed from programs in this scenario. Hence, we subsequently call rules $r \in \Xi^{\mathcal{C}}$ *tautological in* (*domain*) $\mathcal{C}$.

Theorem 1. *For a language $\mathcal{L}$ over finite domain $\mathcal{C}$, we have that for any program P and any rule r, P is strongly equivalent in $\mathcal{L}$ to $P \setminus \{r\}$ iff $r \in \Xi^{\mathcal{C}}$.*

The proof of this result is along the lines of proofs given by Eiter *et al.* [7].

3.1 Complexity of Detecting Tautological Rules in a Finite Domain

In what follows, we establish results about the computational cost for deciding whether a rule is tautological in $\mathcal{C}$, i.e., whether it is contained in $\Xi^{\mathcal{C}}$. Thus, assuming a finite domain, in view of Theorem 1, the considered problem amounts to checking which rules can be faithfully deleted from any program. Recall that in the infinite case, this problem is decidable in linear time by just checking $r \in \Theta$, according to Proposition 1. As we show below, in the finite case we observe an increase of the difficulty to recognize tautological rules, even in very restricted settings.

Theorem 2. *Given a rule r, checking whether r is tautological in a fixed finite domain $\mathcal{C}$ of size ≥ 2 is* coNP*-complete. Hardness holds even for positive rules with bounded arities.*

Proof. Membership is easy. In order to test that a rule is *not* tautological in $\mathcal{C}$, we guess a ground substitution $\vartheta : \mathcal{V}_r \to \mathcal{C}$ and check in polynomial time that $B^+(r\vartheta) \cap (H(r\vartheta) \cup B^-(r\vartheta)) = \emptyset$ holds.

For hardness, suppose that the domain $\mathcal{C}$ is of size $\ell + 1$ with $\ell \geq 1$. Without loss of generality, let $\mathcal{C}$ be of the form $\mathcal{C} = \{0, 1, , \ldots, \ell\}$. We prove coNP-hardness via a reduction from UNSAT. Let $\phi = \bigwedge_{i=1}^{n} l_{i,1} \vee l_{i,2} \vee l_{i,3}$ be a formula in CNF over propositional atoms $X_1, \ldots, X_m$, and consider a positive rule r_ϕ with

$$H(r_\phi) = \{c(0,0,0)\} \cup \{v(\alpha,\beta) \mid (\alpha,\beta) \in \mathcal{C}^2 \setminus \{(0,1),(1,0)\}\} \text{ and}$$
$$B(r_\phi) = \{c(l^*_{i,1}, l^*_{i,2}, l^*_{i,3}) \mid 1 \leq i \leq n\} \cup \{v(X_j, \bar{X}_j) \mid 1 \leq j \leq m\},$$

where $l^* = X$ if $l = X$, and $l^* = \bar{X}$ if $l = \neg X$, with $\bar{X}_1, \ldots, \bar{X}_m$ being new variables. By a slight abuse of notation, we use X_i to denote either a propositional atom (in ϕ) or a first-order variable (in r_ϕ). We claim that ϕ is unsatisfiable iff r_ϕ is tautological in $\mathcal{C}$.

For the "only if"-direction, suppose that r_ϕ is not tautological in $\mathcal{C}$. Hence, since $B^-(r_\phi) = \emptyset$, there exists a substitution $\vartheta : \mathcal{V} \to \mathcal{C}$ such that $B^+(r_\phi\vartheta) \cap H(r_\phi\vartheta) = \emptyset$. Thus, for each propositional atom X_i in ϕ, the pair of first-order variables $(X_i, \bar{X}_i)$ in r_ϕ is either instantiated to $(0,1)$ or $(1,0)$, since otherwise $v(X_i, \bar{X}_i)\vartheta$ would match with some atom $v(\alpha,\beta)$ from $H(r_\phi)$. Thus, we can view ϑ as an assignment to the propositional variables X_i from ϕ. Now, since no atom in $B^+(r_\phi\vartheta)$ matches with $c(0,0,0)$ from $H(r_\phi)$, each clause in ϕ is satisfied "under ϑ", i.e., in each clause at least one propositional literal is assigned to true by the truth assignment ϑ. Hence, ϕ is satisfiable. The "if"-direction is by essentially the same arguments. □

Note that according to Theorem 2, checking whether a disjunctive rule is tautological is coNP-hard even if the domain is fixed and, moreover, the number of predicate symbols and their arities are bounded. If we drop the restriction of bounded arities, then coNP-hardness can be shown even for Horn rules.

Theorem 3. *Given a Horn rule r, checking whether r is tautological in a fixed finite domain $\mathcal{C}$ of size ≥ 2 is* coNP*-complete.*

Proof. Membership is shown as above. Hardness is along the lines of Kunen [9] and Kapur *et al.* [10]. Suppose that the domain $\mathcal{C}$ is of size $\ell + 1$ with $\ell \geq 1$. Without loss of generality, let $\mathcal{C}$ be of the form $\mathcal{C} = \{0, 1, , \ldots, \ell\}$. Again, we prove coNP-hardness via a reduction from UNSAT. Let $\phi = \bigwedge_{i=1}^{n} l_{i,1} \vee l_{i,2} \vee l_{i,3}$ be a formula in CNF over propositional atoms $X_1, \ldots, X_m$. Without loss of generality, we may assume that every propositional variable X_j occurs at most once in each clause, i.e., for every $i \in \{1, \ldots, n\}$, there cannot be two literals such that one is either identical to the other or one is the dual of the other. Note that clauses containing some propositional variable plus its dual can be faithfully deleted from ϕ.

Now consider a Horn rule r_ϕ such that

$$H(r_\phi) = \{p(X_1, \ldots, X_m)\} \text{ and}$$
$$B(r_\phi) = \{p(s_{i1}, \ldots, s_{im}) \mid 1 \leq i \leq n\} \cup$$
$$\{p(X_1, , \ldots, X_{j-1}, \alpha, X_{j+1}, \ldots, X_m) \mid \alpha \in \mathcal{C} \setminus \{0,1\} \text{ and } 1 \leq j \leq m\},$$

where the arguments s_{ij} with $1 \leq i \leq n$ and $1 \leq j \leq m$ are defined as follows:

$$s_{ij} = \begin{cases} 1 & \text{if the negative literal } \neg X_j \text{ occurs in the } i\text{-th clause of } \phi; \\ 0 & \text{if the positive literal } X_j \text{ occurs in the } i\text{-th clause of } \phi; \\ X_i & \text{otherwise.} \end{cases}$$

We claim that ϕ is unsatisfiable iff r_ϕ is tautological in $\mathcal{C}$.

For the "only if"-direction, suppose that ϕ is unsatisfiable. Moreover, let $\vartheta : \mathcal{V} \to \mathcal{C}$ be an arbitrary ground substitution over the variables in r_ϕ. Since $B^-(r_\phi) = \emptyset$, we have to show that $B^+(r_\phi\vartheta) \cap H(r_\phi\vartheta) \neq \emptyset$. First suppose that ϑ instantiates at least one variable X_j to $\alpha \in \mathcal{C} \setminus \{0, 1\}$. Then, $p(X_1, \ldots, X_{j-1}, \alpha, X_{j+1}, \ldots, X_m)\vartheta$ and $p(X_1, \ldots, X_m)\vartheta$ are identical. Thus, in this case, $p(X_1, \ldots, X_m)\vartheta \in B^+(r_\phi\vartheta) \cap H(r_\phi\vartheta)$, and therefore $B^+(r_\phi\vartheta) \cap H(r_\phi\vartheta) \neq \emptyset$.

It remains to consider the case that ϑ instantiates all variables X_j to either 0 or 1. Hence, ϑ defines a truth assignment of $\{X_1, \ldots, X_m\}$. Since ϕ is unsatisfiable, there must exist some clause $l_{i,1} \vee l_{i,2} \vee l_{i,3}$ with truth value false in ϑ. We claim that then $p(X_1, \ldots, X_m)\vartheta = p(s_{i1}, \ldots, s_{im})\vartheta$ holds, i.e., for every j, $X_j\vartheta = s_{ij}\vartheta$. We prove this claim by distinguishing the three cases of the definition of s_{ij}:

- If the negative literal $\neg X_j$ occurs in the i-th clause, then $s_{ij} = 1$. On the other hand, the i-th clause, and therefore also the literal $\neg X_j$, is false under the assignment ϑ. Thus, X_j has the value true in this assignment, i.e., $X_j\vartheta = 1 = s_{ij}\vartheta$.
- If the positive literal X_j occurs in the i-th clause, then $s_{ij} = 0$. On the other hand, the i-th clause, and therefore also the literal X_j, is false under the assignment ϑ. Thus, $X_j\vartheta = 0 = s_{ij}\vartheta$.
- If X_j does not occur in the i-th clause, then $s_{ij} = X_j$, and therefore $X_j\vartheta = s_{ij}\vartheta$ trivially holds.

The "if"-direction is shown analogously and is therefore omitted. □

Note that the coNP-completeness results in Theorems 2 and 3 were shown for an arbitrary but *fixed* finite domain $\mathcal{C}$ of size $|\mathcal{C}| \geq 2$. As far as coNP-hardness is concerned, we thus get slightly stronger results than if we considered the domain $\mathcal{C}$ as part of the problem input. On the other hand, it can be easily verified that the membership proofs clearly also work if the finite domain $\mathcal{C}$ is not fixed (i.e., if it is part of the problem input). In other words, detecting tautological rules has the same complexity no matter whether we have to deal with a specific finite domain or with all finite domains.

3.2 Tractable Cases

We conclude our discussion on the detection of tautological rules by identifying two tractable cases. The first one combines the restrictions considered in Theorems 2 and 3; the second one is obtained by a restriction on the variables occurring in a rule.

Theorem 4. *Given a Horn rule r, where the arity of all predicate symbols is bounded by some fixed constant, checking whether r is tautological in a finite domain is in* P.

Proof. Since r is Horn, the head $H(r)$ of r consists of a single atom A and $B^-(r)$ is empty. Hence, r is tautological in $\mathcal{C}$ iff $A\vartheta \in B^+(r\vartheta)$ holds for every ground substitution ϑ. In order to check this condition, we loop over all possible ground instantiations $A\sigma$ of A. Since the arity of the predicate symbols is bounded and the domain is finite, there are only polynomially many such ground instantiations. For each $A\sigma$, we have to check whether σ can be extended to a substitution ϑ such that $A\sigma = A\vartheta \in B^+(r\vartheta)$. This is a matching problem, which can be clearly solved in polynomial time. □

Theorem 5. *Given a rule r such that $|\mathcal{V}_r| \leq k$ for some fixed constant k, checking whether r is tautological in a finite domain is in* P.

Proof. Since the number of variables in r is bounded by a constant and the domain is finite, there are only polynomially many ground instances $r\vartheta$ of r. In order to test whether r is tautological in $\mathcal{C}$, we just have to test for each ground instance $r\vartheta$ of r whether $B^+(r\vartheta) \cap (H(r\vartheta) \cup B^-(r\vartheta)) \neq \emptyset$ holds. □

4 Rule Subsumption

Rule subsumption is a syntactic criterion to identify a rule r which can be faithfully deleted from any program containing another ("more general") rule s. For the ground case, Lin and Chen [6] generalized replacements from the literature [3,13] and showed that their syntactic criterion captures *all* such pairs of rules. The non-ground case was first studied by Eiter *et al.* [7] and Traxler [14]. As shown by the latter author, also rule subsumption can be characterized in two alternative ways (similarly as before for tautological rules), which turn out to be equivalent for languages over an infinite domain. To formulate these characterizations, let us define the following relations (for any pair of rules r, s):

$$
\begin{aligned}
s \leq r \text{ iff } & \text{there exists a substitution } \vartheta : \mathcal{V}_s \to \mathcal{V}_r \cup \mathcal{C}_r \text{ such that} \\
& H(s\vartheta) \subseteq H(r) \cup B^-(r) \text{ and } B(s\vartheta) \subseteq B(r); \\
s \preceq^{\mathcal{C}} r \text{ iff, for each } \vartheta_r : \mathcal{V}_r \to \mathcal{C}, & \text{ there exists a } \vartheta_s : \mathcal{V}_s \to \mathcal{C} \text{ such that} \\
& H(s\vartheta_s) \subseteq H(r\vartheta_r) \cup B^-(r\vartheta_r) \text{ and } B(s\vartheta_s) \subseteq B(r\vartheta_r).
\end{aligned}
$$

Observe that $\leq$ does not take the underlying domain into account, but only variables and constants involved in the two rules, r and s, while $\preceq^{\mathcal{C}}$ explicitly refers to the domain $\mathcal{C}$ of the underlying language. The following proposition collects results from Eiter *et al.* [7] and Traxler [14].

Proposition 2. *Let $\mathcal{L}$ be given over an infinite domain $\mathcal{C}$ and let r, s be rules. Then, the following relations hold:* (*i*) *$s \leq r$ iff $s \preceq^{\mathcal{C}} r$; and* (*ii*) *if $s \leq r$* (*or, equivalently, $s \preceq^{\mathcal{C}} r$*), *then $P \equiv_{\mathrm{s}} P \setminus \{r\}$, for each P with $s \in P$.*

Note that not only in case $\mathcal{C}$ is infinite, but also if r is ground, the equivalence between $s \leq r$ and $s \preceq^{\mathcal{C}} r$ holds, for arbitrary s. As shown by Eiter *et al.* [7], given rules r, s, deciding whether $s \leq r$ holds is NP-complete. The proof was carried out by a reduction of the 3-coloring problem to checking containment in $\leq$. Inspecting the proof reveals

that NP-hardness already holds if r is restricted to be ground. Again, we can show by a simple example that the equivalence between $\leq$ and $\preceq^{\mathcal{C}}$ does not hold in case $\mathcal{C}$ is finite. Consider, e.g., $s: \ q(X) \leftarrow p(X,X)$, $r: \ q(0) \leftarrow p(1,Y), p(Y,0), \text{not } q(1)$, and $\mathcal{C} = \{0,1\}$. Then, $s \not\leq r$, while $s \preceq^{\mathcal{C}} r$. Again, we have the effect that, for each pair s, r of rules, $s \leq r$ implies $s \preceq^{\mathcal{C}} r$, but not vice versa.

Whenever $s \preceq^{\mathcal{C}} r$ holds, we say that r *is subsumed by* s *(in* $\mathcal{C}$*)*. We next show that $s \preceq^{\mathcal{C}} r$ implies that r can be removed from each program containing s also in case $\mathcal{C}$ is finite.

Theorem 6. *If $s \preceq^{\mathcal{C}} r$, for a finite domain $\mathcal{C}$, then $P \equiv_{\mathrm{s}} P \setminus \{r\}$, for each program P with $s \in P$.*

Proof. We first show that $\{r,s\} \equiv_{\mathrm{s}} \{s\}$. To this end, we show that $grd(\{r,s\},C) \equiv_{\mathrm{s}} grd(\{s\},C)$, for any $C \subseteq \mathcal{C}$.

Consider any $C \subseteq \mathcal{C}$. By assumption, for every $\vartheta : \mathcal{V}_r \to C$, there exists a substitution $\vartheta_s : \mathcal{V}_s \to \mathcal{C}$ such that $H(s\vartheta_s) \subseteq H(r\vartheta_r) \cup B^-(r\vartheta_r)$ and $B(s\vartheta_s) \subseteq B(r\vartheta_r)$. Note that this implies $\vartheta_s(x) \in C$, for each $x \in \mathcal{V}_s$. We thus have $grd(\{r,s\},C) = grd(\{s\},C) \cup \{r\vartheta, s\vartheta_s \mid \vartheta : \mathcal{V}_r \to C\}$, with ϑ_s as above. To every subset $\{r\vartheta, s\vartheta_s\} \subseteq grd(\{r,s\},C)$, i.e., for every $\vartheta : \mathcal{V}_r \to C$, we can apply Theorem 6 of Lin and Chen [6] and replace it by $\{s\vartheta_s\}$. By construction, the resulting program is strongly equivalent to $grd(\{r,s\},C)$, and exactly matches $grd(\{s\},C)$. Thus, $grd(\{r,s\},C) \equiv_{\mathrm{s}} grd(\{s\},C)$, for any $C \subseteq \mathcal{C}$.

Assume now that there exists a program P such that $s \in P$ but $P \not\equiv_{\mathrm{s}} P \setminus \{r\}$, i.e., $P \cup Q \not\equiv (P \setminus \{r\}) \cup Q$, for some program Q. For $P' = (P \setminus \{r,s\}) \cup Q$, we thus have $\{r,s\} \cup P' \not\equiv \{s\} \cup P'$, which implies $\{r,s\} \not\equiv_{\mathrm{s}} \{s\}$, a contradiction. Hence, $P \equiv_{\mathrm{s}} P \setminus \{r\}$ must hold for any program P with $s \in P$. □

4.1 Complexity of Rule Subsumption

Concerning complexity, we already mentioned the NP-completeness of checking rule subsumption given an infinite domain. As in the previous section, we observe an increase of complexity when a finite domain is considered. Note that there is a subtle difference between the Π_2^P-hardness result in Theorem 7 below and the hardness results in Section 3: In Theorem 7, the domain $\mathcal{C}$ is considered to be part of the input and therefore $|\mathcal{C}|$ is not bounded by a fixed constant. In contrast, Section 3 provided hardness results even for a fixed domain.

Theorem 7. *Given rules r and s, checking whether r is subsumed by s in a finite domain $\mathcal{C}$ is Π_2^P-complete. Hardness holds even for Horn rules over a bounded number of predicate symbols with bounded arities.*

Proof. For membership, we show that the complementary problem is in Σ_2^P: Guess ϑ_r and check that for each $\vartheta_s : \mathcal{V}_s \to \mathcal{C}$, either $H(s\vartheta_s) \not\subseteq H(r\vartheta_r) \cup B^-(r\vartheta_r)$ or $B(s\vartheta_s) \not\subseteq B(r\vartheta_r)$ holds. The latter check is in coNP, since checking whether there exists some $\vartheta_s : \mathcal{V}_s \to \mathcal{C}$ with $H(s\vartheta_s) \subseteq H(r\vartheta_r) \cup B^-(r\vartheta_r)$ and $B(s\vartheta_s) \subseteq B(r\vartheta_r)$ is clearly in NP.

For showing hardness, we proceed along the lines of Pichler [15]. We reduce the Π_2^P-complete decision problem of $\forall\exists$-QSAT to testing whether $s \preceq^{\mathcal{C}} r$ holds. To this end, let $\Phi = \forall X_1 \ldots \forall X_k \exists X_{k+1} \ldots \exists X_m \phi$ be a QBF with $\phi = \bigwedge_{i=1}^{n} l_{i,1} \vee l_{i,2} \vee l_{i,3}$, and let the domain $\mathcal{C}$ be of size $k+1$, i.e., without loss of generality, assume $\mathcal{C} = \{0, 1, , \ldots, k\}$. We use two rules, r_Φ and s_Φ, which have empty heads and purely positive bodies:

$$\begin{aligned} B^+(r_\Phi) &= \{p(1, X_1), \ldots, p(k, X_k)\} \cup \{v(\alpha, 0), v(0, \alpha) \mid \alpha \in \mathcal{C} \setminus \{0\}\} \cup \\ &\quad \{c(\alpha, \beta, \gamma) \mid (\alpha, \beta, \gamma) \in \mathcal{C}^3 \setminus \{(0,0,0)\}\}, \\ B^+(s_\Phi) &= \{p(1, X_1), \ldots, p(k, X_k)\} \cup \{v(X_j, \bar{X}_j) \mid 1 \leq j \leq m\} \cup \\ &\quad \{c(l^*_{i,1}, l^*_{i,2}, l^*_{i,3}) \mid 1 \leq i \leq n\}, \end{aligned}$$

where $l^* = X$ if $l = X$, and $l^* = \bar{X}$ if $l = \neg X$, with $\bar{X}_1, \ldots, \bar{X}_m$ being new atoms. We show that Φ is true iff $s_\Phi \preceq^{\mathcal{C}} r_\Phi$.

For the "only if"-direction, suppose that Φ is true and let ϑ_r be an arbitrary ground substitution of the variables $\{X_1, \ldots, X_k\}$ in r_Φ. We define a truth assignment I for $\{X_1, \ldots, X_k\}$ with $I(X_i)$ = false if $X_i\vartheta_r = 0$ and $I(X_i)$ = true otherwise. By hypothesis, Φ is true. Hence, there exists an extension J of I for $\{X_1, \ldots, X_m\}$ such that ϕ is true in J. From J, we define the ground substitution ϑ_s as follows:

$$X_i\vartheta_s = \begin{cases} 0 & \text{if } X_i \text{ is false in } J; \\ X_i\vartheta_r & \text{if } X_i \text{ is true in } J \text{ and } i \leq k; \\ 1 & \text{if } X_i \text{ is true in } J \text{ and } i > k; \end{cases} \qquad \bar{X}\vartheta_s = \begin{cases} 0 \text{ if } X_i \text{ is true in } J\,; \\ 1 \text{ if } X_i \text{ is false in } J. \end{cases}$$

It remains to show that $B^+(s_\Phi\vartheta_s) \subseteq B^+(r_\Phi\vartheta_r)$ holds for ϑ_s. For every $i \leq k$, we have $X_i\vartheta_s = X_i\vartheta_r$ by construction. Hence, every atom $p(i, X_i)\vartheta_s$ in $s_\Phi\vartheta_s$ is contained in $B^+(r_\Phi\vartheta_r)$. Moreover, by construction, for every $j \in \{1, \ldots, m\}$, exactly one of the variables X_j and $\bar{X}_j$ is instantiated to 0 by ϑ_s. Hence, every atom $v(X_j, \bar{X}_j)\vartheta_s$ is either of the form $v(\alpha, 0)$ or $v(0, \alpha)$, for some $\alpha \neq 0$. Thus, every atom $v(X_j, \bar{X}_j)\vartheta_s$ is contained in $B^+(r_\Phi\vartheta_r)$. Finally, ϕ is true in J, i.e., in all clauses of ϕ, at least one literal is true in J. Hence, by construction, for each i, at least one of the first-order variables $l^*_{i,1}, l^*_{i,2}, l^*_{i,3}$ is instantiated to a constant different from 0 by ϑ_s. Thus, all atoms $c(l^*_{i,1}, l^*_{i,2}, l^*_{i,3})\vartheta_s$ are different from $c(0,0,0)$ and are therefore contained in $B^+(r_\Phi)$.

For the "if"-direction, suppose that $s_\Phi \preceq^{\mathcal{C}} r_\Phi$, and let I be an arbitrary truth assignment for $\{X_1, \ldots, X_k\}$. Then, we define the ground substitution ϑ_r over $\{X_1, \ldots, X_k\}$ as $X_i\vartheta_r = 0$ if $I(X_i)$ = false and $X_i\vartheta_r = 1$ if $I(X_i)$ = true. By hypothesis, $s_\Phi \preceq^{\mathcal{C}} r_\Phi$. Thus, there is a substitution ϑ_s over $\{X_1, \ldots, X_m\}$ where $B^+(s_\Phi\vartheta_s) \subseteq B^+(r_\Phi\vartheta_r)$. From ϑ_s we define the extension J of I for $\{X_1, \ldots, X_m\}$ as follows: $J(X_i)$ = false if $X_i\vartheta_s = 0$ and $J(X_i)$ = true if $X_i\vartheta_s \neq 0$.

For every $i \leq k$, $X_i\vartheta_s = X_i\vartheta_r$ holds. Hence, J and I coincide on $\{X_1, \ldots, X_k\}$, and therefore J is indeed an extension of I. By assumption, every atom $v(X_j, \bar{X}_j)\vartheta_s$ is either of the form $v(\alpha, 0)$ or $v(0, \alpha)$, for some $\alpha \neq 0$. Thus, by the definition of J, we also have that $J(\bar{X}_i)$ = false if $\bar{X}_i\vartheta_s = 0$ and $J(\bar{X}_i)$ = true if $\bar{X}_i\vartheta_s \neq 0$. Finally, all atoms $c(l^*_{i,1}, l^*_{i,2}, l^*_{i,3})\vartheta_s$ are contained in $B^+(r_\Phi)$ and are therefore different from $c(0,0,0)$, i.e., for each i, at least one of the first-order variables $l^*_{i,1}, l^*_{i,2}, l^*_{i,3}$ is instantiated to a constant different from 0 by ϑ_s. But then, in all clauses $l_{i,1} \vee l_{i,2} \vee l_{i,3}$ of ϕ, at least one literal is true in J. Thus, ϕ is true in J. This holds for arbitrary I, consequently Φ is true. □

Note that in the proof of Theorem 7, the domain $\mathcal{C}$ is part of the input. This is in stark contrast to Theorems 2 and 3, were the domain $\mathcal{C}$ is arbitrary but fixed—thus leading to slightly stronger hardness results. However, in the Π_2^P-hardness proof of Theorem 7, it is crucial that there is no fixed bound on the size of the domain. In particular, we indeed need k pairwise distinct domain elements (where k corresponds to the number of universally quantified propositional variables in Φ) in order to make sure that any instantiation of a first-order variable X_i in r_Φ forces the same instantiation of the variable X_i (with precisely the same index i) in s_Φ.

Alternatively, we could have considered the domain $\mathcal{C}$ to be fixed (with $|\mathcal{C}| \geq 2$) and either let the number of predicate symbols or the arity of the predicate symbols be unbounded. In case of an unbounded number of predicate symbols, we can simply replace the atoms $p(1, X_1), \ldots, p(k, X_k)$ in both r_Φ and s_Φ by atoms of the form $p_1(X_1), \ldots, p_k(X_k)$. Likewise, if the domain is fixed and the arities of predicate symbols are unbounded, then we replace the atoms $p(1, X_1), \ldots, p(k, X_k)$ in both r_Φ and s_Φ by a single atom $p(X_1, \ldots, X_k)$.

However, if the domain $\mathcal{C}$ is fixed (or at least its cardinality is bounded) and, moreover, both the number of predicate symbols and their arity is bounded, then Π_2^P-completeness no longer holds unless the polynomial hierarchy collapses.

Theorem 8. *Given rules r and s, with $ar_{\max}(\mathcal{A}_{\{r,s\}}) \leq k$ and $|\mathcal{A}_{\{r,s\}}| \leq k'$ for fixed constants k, k', checking whether r is subsumed by s in a domain of fixed size is in Δ_2^P.*

Proof. Since the cardinality of $\mathcal{C}$, the number of predicate symbols, and the arity of predicate symbols are all bounded, there is only a constant number, K, of different ground rules in this language. Note that $s \not\preceq^{\mathcal{C}} r$ iff there exists a ground instance $r\vartheta_r$ of r such that, for every ground substitution $\vartheta_s : \mathcal{V}_s \to \mathcal{C}$, either $H(s\vartheta_s) \not\subseteq H(r\vartheta_r) \cup B^-(r\vartheta_r)$ or $B(s\vartheta_s) \not\subseteq B(r\vartheta_r)$.

In order to check whether such a ground instance $r\vartheta_r$ of r exists, we loop over all K ground rules t and check by one NP-oracle call that t is a ground instance of r and by another NP-oracle call that $s \not\preceq^{\mathcal{C}} t$. Note that both checks are indeed feasible by NP-oracles: On the one hand, checking whether t is a ground instance of r amounts to guessing a ground substitution ϑ_r and checking that $r\vartheta_r = t$ holds. On the other hand, checking whether $s \preceq^{\mathcal{C}} t$ amounts to guessing a ground substitution ϑ_s and checking that both $H(s\vartheta_s) \subseteq H(t) \cup B^-(t)$ and $B(s\vartheta_s) \subseteq B(t)$ hold. □

4.2 Restricting Variable Occurrences and Tractability

As in Section 3, we conclude our discussion by considering restrictions on the variables occurring in the rules. Since, for subsumption, we are dealing with two rules, we distinguish those cases where variable occurrences are restricted in either one of the rules, or in both. It turns out that just a restriction of variable occurrences in both rules guarantees tractability of subsumption detection.

Theorem 9. *Given rules r, s, checking whether r is subsumed by s in a domain of fixed size ≥ 2 is (a) NP-hard, if $|\mathcal{V}_r|$ is bounded by a fixed constant, and even if r is ground and purely positive, and (b) coNP-hard, if $|\mathcal{V}_s|$ is bounded by a fixed constant, and even if s consists of a single body atom.*

Proof. (a) Even when r is ground and purely positive, the problem of subsumption detection in answer-set programming corresponds to "normal" first-order subsumption of clauses, which is a well known NP-hard problem (cf. Problem [LO18] in Garey and Johnson [12]).

(b) Let $|\mathcal{C}| = k$. Without loss of generality, assume $\mathcal{C} = \{1, \ldots, k\}$. We first suppose that $k \geq 3$. In this case, we reduce the k-colorability problem to the complementary problem of rule subsumption. Let an instance of the k-colorability problem be given by the graph $G = (V, E)$, where V denotes the vertices and E the edges. We construct two rules, r_G and s_G, as follows:

$$B^+(r_G) = \{e(X_i, X_j) \mid \{v_i, v_j\} \in E\};$$
$$B^+(s_G) = \{e(X, X)\}.$$

We claim that G is k-colorable iff $s_G \not\preceq^{\mathcal{C}} r_G$.

For the 'if"-direction, suppose that G is k-colorable, i.e., there exists a coloring $f : V \to \{1, \ldots, k\}$ such that no two adjacent vertices are assigned with the same color. Now define the ground substitution $\vartheta_r : \mathcal{V} \to \mathcal{C}$ as $X_i\vartheta_r = f(v_i)$. Then, by construction, $B^+(r_G\vartheta_r)$ does not contain an atom $e(X_i, X_j)\vartheta$ with $X_i\vartheta = X_j\vartheta$, since otherwise also $f(v_i) = f(v_j)$ for some edge $\{v_i, v_j\}$ of the graph G. But this is impossible for a valid k-coloring f.

For the "only if"-direction, suppose that $s_G \not\preceq^{\mathcal{C}} r_G$. Hence, there exists a ground substitution $\vartheta : \mathcal{V} \to \mathcal{C}$ such that $B^+(r_G\vartheta_r)$ does not contain an atom $e(X_i, X_j)\vartheta$ with $X_i\vartheta = X_j\vartheta$. But then we can clearly define a valid k-coloring of the graph G as $f : V \to \{1, \ldots, k\}$ such that $f(v_i) = X_i\vartheta_r$.

It remains to consider the case where $|\mathcal{C}| = 2$. Without loss of generality, assume $\mathcal{C} = \{0, 1\}$. In this case, we establish coNP-hardness by a reduction of the 4-colorability problem to the complementary problem of rule subsumption. Let an instance of the 4-colorability problem be given by the graph $G = (V, E)$. We construct the rules r_G and s_G as follows:

$$B^+(r_G) = \{e(X_i, Y_i, X_j, Y_j) \mid \{v_i, v_j\} \in E\};$$
$$B^+(s_G) = \{e(X, Y, X, Y)\}.$$

We claim that G is 4-colorable iff $s_G \not\preceq^{\mathcal{C}} r_G$. The proof is essentially as in the case $k \geq 3$ above. However, now pairs of variables (X, Y) are considered as a binary encoding of the four colors in the graph. □

Theorem 10. *Given rules r, s such that $|\mathcal{V}_r \cup \mathcal{V}_s|$ is bounded by a fixed constant, checking whether r is subsumed by s in a finite domain is in* P.

Proof. Since the number of variables in r and s is bounded by a constant and the number of domain elements is finite, there are only polynomially many ground instances $r\vartheta_r$ and $s\vartheta_s$, respectively. Hence, we may test in a loop over all ground instances $r\vartheta_r$ if there exists an instance $s\vartheta_s$ such that $H(s\vartheta_s) \subseteq H(r\vartheta_r) \cup B^-(r\vartheta_r)$ and $B(s\vartheta_s) \subseteq B(r\vartheta_r)$ hold. The latter test requires simply a nested loop over polynomially many ground instances $s\vartheta_s$. □

Table 1. Complexity of detecting tautological rules in finite (possibly fixed) domains

	general case	$ar_{\max}(\mathcal{A}_r) \leq k$	$\lvert\mathcal{V}_r\rvert \leq k$
r disjunctive	coNP-complete	coNP-complete	in P
r positive	coNP-complete	coNP-complete	in P
r Horn	coNP-complete	in P	in P

Table 2. Complexity of detecting rule subsumption $s \preceq^{\mathcal{C}} r$ in fixed finite domains

	general case	$ar_{\max}(\mathcal{A}_{\{r,s\}}) \leq k$	$\lvert\mathcal{V}_r\rvert \leq k$
general case	Π_2^P-complete	Π_2^P-complete	NP-hard
$\lvert\mathcal{A}_{\{r,s\}}\rvert \leq k$	Π_2^P-complete	in Δ_2^P	NP-hard
$\lvert\mathcal{V}_s\rvert \leq k$	coNP-hard	coNP-hard	in P

5 Discussion and Conclusion

We investigated the complexity of applying rule eliminations in the setting of finite domains, and provided a full complexity picture with respect to several restrictions, in particular restricting the syntax to Horn rules, imposing a bound on predicate arities and/or on the number of variables (a summary of our results is given in Tables 1 and 2). Note that the concept of bounded predicate arities was suggested by Eiter *et al.* [16] in order to reduce the complexity of basic reasoning tasks in answer-set programming from nondeterministic exponential time classes to classes from the polynomial hierarchy. Similarly, Vardi [17] used bounded variables in order to narrow the gap between expression and data complexity of database queries (i.e., Horn programs).

The main observation of our results is that if we consider finite domains then the detection of tautological or subsumed rules becomes, in general, harder. More specifically, we observed an increase from P to coNP as well as from NP to Π_2^P. However, we also identified restrictions such that complexity does not increase. To wit, a restriction to Horn clauses makes the detection of tautological rules tractable, but only if we additionally impose a bound on the arities of predicate symbols (cf. the first two columns of Table 1).

As for the detection of subsumed rules, restricting to Horn clauses is irrelevant since all hardness results in Section 4 were shown for Horn clauses. On the other hand, Table 2 reflects the effects of other restrictions: In the second row, the case of fixing the number of predicate symbols by some constant is considered. This restriction leads to a decrease of complexity if it is combined with a bound on the arities of predicate symbols. However, in order to obtain tractability, more severe restrictions are required. For instance, a restriction on the number of variable occurrences in both r and s is a sufficient condition for the tractability of detecting subsumed rules (cf. the third row, last column, of Table 2).

We finally remark, however, that local checks for rule redundancy, as presented here, may pay off in program simplification since the complexity of checking rule redundancy (which amounts to testing strong equivalence) is in general much harder, viz. complete for co-NEXPTIME.

References

1. Eiter, T., Fink, M., Tompits, H., Woltran, S.: Strong and Uniform Equivalence in Answer-Set Programming: Characterizations and Complexity Results for the Non-Ground Case. In Proc. AAAI'05, AAAI Press (2005) 695–700
2. Lifschitz, V., Pearce, D., Valverde, A.: Strongly Equivalent Logic Programs. ACM Transactions on Computational Logic **2(4)** (2001) 526–541
3. Brass, S., Dix, J.: Semantics of (Disjunctive) Logic Programs Based on Partial Evaluation. Journal of Logic Programming **38(3)** (1999) 167–213
4. Osorio, M., Navarro, J.A., Arrazola, J.: Equivalence in Answer Set Programming. In Proc. LOPSTR'01. Volume 2372 of LNCS, Springer-Verlag (2001) 57–75
5. Inoue, K., Sakama, C.: Equivalence of Logic Programs Under Updates. In: Proc. JELIA'04. Volume 3229 of LNCS, Springer-Verlag (2004) 174–186
6. Lin, F., Chen, Y.: Discovering Classes of Strongly Equivalent Logic Programs. In Proc. IJCAI'05, (2005) 516–521
7. Eiter, T., Fink, M., Tompits, H., Traxler, P., Woltran, S.: Replacements in Non-Ground Answer-Set Programming. In Proc. KR'06, AAAI Press (2006) 340–351
8. Lassez, J.L., Marriott, K.: Explicit Representation of Terms Defined by Counter Examples. Journal of Automated Reasoning **3(3)** (1987) 301–317
9. Kunen, K.: Answer Sets and Negation as Failure. In Proc. ICLP'87, MIT Press (1987) 219–228
10. Kapur, D., Narendran, P., Rosenkrantz, D., Zhang, H.: Sufficient-Completeness, Ground-Reducibility and their Complexity. Acta Informatica **28(4)** (1991) 311–350
11. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing **9** (1991) 365–385
12. Garey, M.R., Johnson, D.S.: Computers and Intractability: A Guide to the Theory of NP-Completeness. Freeman (1979)
13. Wang, K., Zhou, L.: Comparisons and Computation of Well-founded Semantics for Disjunctive Logic Programs. ACM Transactions on Computational Logic **6(2)** (2005) 295–327
14. Traxler, P.: Techniques for Simplifying Disjunctive Datalog Programs with Negation. Master's thesis, Technische Universität Wien, Institut für Informationssysteme (2006)
15. Pichler, R.: On the Complexity of H-Subsumption. In Proc. CSL'98. Volume 1584 of LNCS, Springer-Verlag (1998) 355–371
16. Eiter, T., Faber, W., Fink, M., Pfeifer, G., Woltran, S.: Complexity of Answer Set Checking and Bounded Predicate Arities for Non-ground Answer Set Programming. In Proc. KR'04, AAAI Press (2004) 377–387
17. Vardi, M.: On the Complexity of Bounded-Variable Queries. In Proc. PODS'95, (1995) 266–276