Complexity Results for Checking Equivalence of Stratified Logic Programs*

Thomas Eiter, Michael Fink, Hans Tompits, and Stefan Woltran
Institut für Informationssysteme, Technische Universität Wien,
Favoritenstraße 9–11, A-1040 Vienna, Austria
{eiter,fink,tompits,stefan}@kr.tuwien.ac.at

Abstract

Recent research in nonmonotonic logic programming under the answer-set semantics focuses on different notions of program equivalence. However, previous results do not address the important classes of stratified programs and its subclass of acyclic (i.e., recursion-free) programs, although they are recognized as important tools for knowledge representation and reasoning. In this paper, we consider such programs, possibly augmented with constraints. Our results show that in the propositional setting, where reasoning is well-known to be polynomial, deciding strong and uniform equivalence is as hard as for arbitrary normal logic programs (and thus coNP-complete), but is polynomial in some restricted cases. Non-ground programs behave similarly. However, exponential lower bounds already hold for small programs (i.e., with constantly many rules). In particular, uniform equivalence is undecidable even for small Horn programs plus a single negative constraint.

1 Introduction

In recent years, a large body of work in answer-set programming (ASP) has been devoted to obtain semantical and computational characterizations of different notions of equivalence between nonmonotonic logic programs. These investigations were mainly concerned with *strong equivalence* [Lifschitz *et al.*, 2001], which facilitates a replacement property under nonmonotonicity akin, e.g., to the familiar one which holds for classical logic, and weaker notions such as *uniform equivalence* [Eiter and Fink, 2003; Pearce and Valverde, 2004]. While the general case and certain fragments like positive programs are well-understood [Eiter *et al.*, 2005b], the picture remains unclear for many other fragments.

Two important such fragments are stratified programs and its subclass of acyclic (i.e., recursion-free) programs, which are widely used in knowledge representation (cf., e.g., Baral [2003]), and also play, for instance, an important role in deductive databases. The attractive features of such programs are their unambiguous semantics as well as their (in the propositional case) polynomial complexity.

In ASP, programs often contain stratified subprograms which serve, in a generate-and-test methodology [Gelfond and Leone, 2002; Niemelä, 1999; Baral, 2003], for checking whether a candidate is in fact a solution to the problem at hand. For example, in the program

$$\begin{aligned} bad &\leftarrow not\ ok, \\ ok &\leftarrow choose(X), \\ choose(X) &\leftarrow option(X), not\ omit(X), \\ omit(X) &\leftarrow option(X), not\ choose(X), \end{aligned}$$

the first two statements check whether at least one option was chosen, and output "bad" if this is not the case. Typically, to eliminate such unwanted models, a constraint $\leftarrow bad$ is added. This remains true if bad is deleted from the head of the first rule. In fact, the sets of rules $P = \{\leftarrow bad;\ bad \leftarrow not\ ok;\ ok \leftarrow choose(X)\}$ and $Q = \{\ \leftarrow bad;\ \leftarrow not\ ok;\ ok \leftarrow choose(X)\}$ are strongly equivalent, and thus P can be replaced by Q (and vice versa) within an arbitrary program. If no rules in it define bad, then we can also safely drop the constraint $\leftarrow bad$. We note that, as in this example, checking parts are often acyclic. However, recursion may occur, e.g., in computing the transitive closure for reachability in a graph; we remind that the capability of expressing transitive closure is one of the assets of logic programming.

Given their importance in practice, in this paper, we consider testing the equivalence of stratified programs which are possibly augmented with constraints. Our contributions can be briefly summarized as follows.

(1) We provide a detailed picture of the computational complexity of deciding strong and uniform equivalence between stratified respectively acyclic logic programs in the propositional case. As we show, these problems are coNP-complete already in very simple cases, and thus as hard as in the general case of normal logic programs. This contrasts with the non-ground case, where syntactic restrictions make strong equivalence testing (which is co-NEXPTIME-time complete in general) easier in some cases.

(2) We elucidate the role of constraints in this context. Without constraints, in some cases testing strong equivalence becomes tractable and thus easier than testing uniform equivalence, which remains intractable. There is a similar picture

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

in the non-ground case. A noticeable result which we establish is that uniform equivalence of two programs P and Q is undecidable as soon as both P and Q are permitted to contain, besides Horn rules, a single negative constraint $\leftarrow \ not\ w$, where w is a propositional atom. If it is removed in only one of them, then we have decidability (also in the presence of arbitrary positive constraints).

(3) We discuss the effect of some restrictions which are important in practice including "small" programs, where the number of rules is bounded by a constant. While strong and uniform equivalence become unsurprisingly tractable in the propositional case, both remain intractable (at least exponential) in the non-ground case. In fact, uniform equivalence remains undecidable.

Our results fill a gap in the knowledge about the complexity of testing equivalence for stratified programs with constraints, and complement previous results. They show that, somewhat surprisingly, already for simple propositional programs, testing strong respectively uniform equivalence is intractable. Recall that deciding ordinary equivalence of such programs, i.e., whether they have the same answer set, is polynomial (in fact, feasible in linear time). Furthermore, our results show that also under some restrictions relevant in practice the problems remain intractable, i.e., have exponential lower bounds, or remain even undecidable.

2 Background

We use a language containing the following (possibly infinite) sets: a set $\mathcal{A}$ of *predicate symbols*, a set $\mathcal{V}$ of *variables*, and a set $\mathcal{C}$ of *constants* (called the *domain*). An *atom* is an expression of form $p(t_1, \ldots, t_n)$, where $p \in \mathcal{A}$ has arity $\alpha(p) = n$ and $t_i \in \mathcal{C} \cup \mathcal{V}$, for $1 \leq i \leq n$.

A *rule*, r, is of the form

$$a \leftarrow b_1, \ldots, b_k,\ not\ b_{k+1}, \ldots,\ not\ b_m,$$

where a is an atom or empty, $b_1, \ldots, b_m$ are atoms ($m \geq k \geq 0$), and "not" denotes *default negation*. If a is empty, r is a *constraint*; if $m = 0$, r is a *fact*; and if $m = k$, r is *Horn*. The *head* of r is given by $H(r) = a$, and the *body* of r by $B(r) = \{b_1, \ldots, b_k,\ not\ b_{k+1}, \ldots,\ not\ b_m\}$. We also use $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$. Moreover, we assume that all rules are *safe*, i.e., each variable in $H(r) \cup B^-(r)$ also occurs in $B^+(r)$.

A *program* is a finite set of rules. The set of variables occurring in an expression e (atom, rule, program, etc.) is denoted by $\mathcal{V}_e$, and the set of constants (resp., predicates) occurring in e by $\mathcal{C}_e$ (resp., $\mathcal{A}_e$). Expression e is *ground* iff $\mathcal{V}_e = \emptyset$.

A predicate $p \in \mathcal{A}_P$ is called *extensional* (in P) iff there is no $r \in P$ with $H(r) = p$, otherwise it is *intensional* (in P). The Herbrand base, $B_{\mathcal{A},\mathcal{C}}$, is the set of all ground atoms over predicates $\mathcal{A}$ and constants $\mathcal{C}$. $B_P = B_{\mathcal{A}_P,\mathcal{C}_P}$ is the Herbrand base of a program P.

Given a rule r and a set of constants $C \subseteq \mathcal{C}$, we define $Gr(r, C)$ as the set of all rules obtained from r by all possible substitutions of $\mathcal{V}_r$ by elements from C. For any program P, the *grounding of P with respect to C* is given by $Gr(P, C) = \bigcup_{r \in P} Gr(r, C)$. In particular, $Gr(P)$ stands for $Gr(P, \mathcal{C}_p)$, referred to simply as the *grounding of* P, where $\mathcal{C}_P = \{c\}$ if no constant appears in P.

A program is *constraint-free*, if no constraint occurs in it, and *Horn*, if all rules are Horn. Constraint-free Horn programs are also called *definite*. A program P is called *acyclic* (resp., *stratified*) iff there is a function $f : \mathcal{A} \to N$ such that, for each $r \in P$ with $H(r) = h$ being nonempty, (i) $f(\mathcal{A}_b) < f(\mathcal{A}_h)$, for each $b \in B^-(r)$, and (ii) $f(\mathcal{A}_b) < f(\mathcal{A}_h)$ (resp., $f(\mathcal{A}_b) \leq f(\mathcal{A}_h)$), for each $b \in B^+(r)$.

By an *interpretation*, I, we understand a set of ground atoms. A ground rule r is *satisfied* by I, symbolically $I \models r$, iff whenever $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$, then $H(r) \in I$. I satisfies a ground program P, denoted by $I \models P$, iff each $r \in P$ is satisfied by I. A set $I \subseteq B_P$ is an *answer set* of P iff I is a subset-minimal set satisfying the *Gelfond-Lifschitz reduct* P^I [Gelfond and Lifschitz, 1991] of $Gr(P)$ with respect to I, given by

$$P^I = \{H(r) \leftarrow B^+(r) \mid r \in Gr(P),\ I \cap B^-(r) = \emptyset\}.$$

The set of all answer sets of P is denoted by $\mathcal{AS}(P)$. As well known, every stratified program P has at most one answer set, which in case that P is acyclic can be defined by the *completion of* P [Ben-Eliyahu and Dechter, 1994].

We consider the following notions of equivalence between two programs P and Q:

- ordinary equivalence, $P \equiv_o Q$: $\mathcal{AS}(P) = \mathcal{AS}(Q)$;
- uniform equivalence, $P \equiv_u Q$: for each finite set F of facts, $\mathcal{AS}(P \cup F) = \mathcal{AS}(Q \cup F)$; and
- strong equivalence, $P \equiv_s Q$: for each program S, $\mathcal{AS}(P \cup S) = \mathcal{AS}(Q \cup S)$.

We remark that the difficulty to deal with uniform or strong equivalence in the non-ground case has its origin in the extended programs which naturally enlarge the active domain, and thus the original Herbrand bases of the programs are not useful anymore [Eiter *et al.*, 2005a].

3 The Propositional Case

In this section, programs are assumed to be propositional (i.e., all predicates have arity 0) and normal unless specified otherwise. The following proposition collects results by Eiter *et al.* [2005b] and some known properties.

Proposition 1 *Deciding $P \equiv_e Q$ for normal programs P and Q, where $e \in \{s, u, o\}$, is in* coNP*, and if both programs are Horn, the problem is in* P*. For stratified programs, deciding $P \equiv_o Q$ is in* P*.*

We start proving a P-hardness result for the simplest class of programs considered.

Lemma 1 *Deciding $P \equiv_e Q$ for acyclic definite Horn programs is* P*-hard, for $e \in \{o, u, s\}$.*

Proof. We reduce the P-complete problem of *monotone circuit value* [Goldschlager, 1977] to the equivalence problems in question. Such a circuit is given by a directed acyclic graph (V, E), where $V = \{1, \ldots, n\}$ and each $(i, j) \in E$ satisfies $i < j$. Elements of V are so-called *gates* of specific sorts, viz. $\top$, $\bot$, $\wedge$, or $\vee$. Gates of the former two sorts have in-degree 0, the others have in-degree 2. All gates except n have outgoing edges; n itself is called the output gate. Their semantics

is defined straightforwardly by induction for each gate j as follows: We have $v(j) = 1$ (resp., $v(j) = 0$), if j is of sort $\top$ (resp., $\bot$). In the other cases, we have exactly two edges (i_1, j), (i_2, j) in E and define $v(j) = v(i_1) \cdot v(i_2)$ in case j is of sort $\wedge$ and as $v(j) = 1$ iff $v(i_1) + v(i_2) \geq 1$ in case j is of sort $\vee$. Deciding whether $v(n) = 1$ for the output gate n is the problem of monotone circuit value.

Given (V, E), we define a program P as follows: for each gate j of sort $\top$, there is a fact p_j in P; for each gate j of sort $\vee$ having (i_1, j), (i_2, j) in E, we have rules $p_j \leftarrow p_{i_1}$ and $p_j \leftarrow p_{i_2}$ in P; for each gate j of sort $\wedge$ having (i_1, j), (i_2, j) in E, we have the rule $p_j \leftarrow p_{i_1}, p_{i_2}$ in P. Now, given output gate n, we compare the programs $P \cup \{o \leftarrow\}$ and $P \cup \{o \leftarrow p_n\}$. Obviously both programs are acyclic Horn without constraints and can be constructed from the given circuit (V, E) in polynomial time using logarithmic space.

As easily verified, we have $P \cup \{o \leftarrow\} \equiv_o P \cup \{o \leftarrow p_n\}$ iff $v(n) = 1$ holds in (V, E). Moreover, by monotonicity, this correspondence extends to $\equiv_u$ and $\equiv_s$. □

Next, we establish a further P-membership result.

Lemma 2 *Deciding $P \equiv_s Q$ between a normal program P and a Horn program Q is in* P*, if at least one of the programs is constraint-free.*

Proof (Sketch). Since P or Q is constraint-free, the interpretation $I = \mathcal{A}_{P \cup Q}$ has to be a model of both P^I and Q as a necessary condition for strong equivalence to hold. This test is in P. However, this can only hold if the Horn program Q is constraint-free. That is, the test amounts to check whether P is strongly equivalent to a definite Horn program, and to test whether $P^I \equiv_s Q$, which are both in P. □

Let us now turn to acyclic normal programs. For acyclic programs with constraints, we obtain the following result:

Lemma 3 *Deciding $P \equiv_s Q$ is* coNP*-hard for acyclic programs* (*with constraints*)*, even if Q is Horn.*

Proof. We use an embedding of the coNP-complete *exact hitting set* problem, given by an instance (C, S), where C is a family of subsets of a finite set S. The problem is to decide whether there exists a subset $S' \subseteq S$ such that $|S' \cap C'| = 1$, for each $C' \in C$. Any such S' is an exact hitting set (EHS) for C. For coNP-hardness, it is sufficient to consider C containing sets of cardinality 3. So, let $C = \{\{c_{1,1}, c_{1,2}, c_{1,3}\}, \ldots, \{c_{n,1}, c_{n,2}, c_{n,3}\}\}$. We construct the acyclic program P consisting of the constraints

$$\leftarrow c_{i,1}, c_{i,2}, \quad \leftarrow c_{i,1}, c_{i,3}, \quad \leftarrow c_{i,2}, c_{i,3}, \tag{1}$$

$$\leftarrow \mathit{not}\, c_{i,1}, \mathit{not}\, c_{i,2}, \mathit{not}\, c_{i,3}, \tag{2}$$

for $1 \leq i \leq n$. Furthermore, consider $Q = \{a \leftarrow; \leftarrow a\}$, which is also acyclic and Horn. We show that there exists an EHS for C iff $P \not\equiv_s Q$. Clearly, this equivalence problem is constructible in polynomial time in the size of (C, S).

Suppose that $P \not\equiv_s Q$. Then, there exists a classical model M of P. By the Constraints (2), the model is nonempty and $|\{c_{i,1}, c_{i,2}, c_{i,3}\} \cap M| > 0$, for each $1 \leq i \leq n$. Furthermore, in view of Constraints (1), $|\{c_{i,1}, c_{i,2}, c_{i,3}\} \cap M| \leq 1$ has to hold for each $1 \leq i \leq n$. But then M is an EHS.

For the only-if direction, one easily shows that any EHS S' of (C, S) is a classical model of P. Thus, the existence of an EHS implies $P \not\equiv_s Q$. □

$\equiv_s$ / $\equiv_u$	normal	constr.-free	Horn	definite
definite	P / coNP	P / coNP	P	P
Horn	coNP	P / coNP	P	
constr.-free	coNP	coNP		
normal	coNP			

Table 1: Complexity landscape for propositional programs.

Disallowing constraints yields coNP-hardness only if *both* programs use negation.

Lemma 4 *Deciding $P \equiv_s Q$ is* coNP*-hard for constraint-free acyclic programs.*

Proof (Sketch). Take the exact hitting set problem (C, S) from the proof of Lemma 3, and define

$$\begin{aligned} R = \bigcup_{1 \leq i \leq n} \{ & a \leftarrow c_{i,1}, c_{i,2};\ a \leftarrow c_{i,1}, c_{i,3};\ a \leftarrow c_{i,2}, c_{i,3}; \\ & a \leftarrow \mathit{not}\, c_{i,1}, \mathit{not}\, c_{i,2}, \mathit{not}\, c_{i,3}; \\ & b \leftarrow c_{i,1}, c_{i,2};\ b \leftarrow c_{i,1}, c_{i,3};\ b \leftarrow c_{i,2}, c_{i,3}; \\ & b \leftarrow \mathit{not}\, c_{i,1}, \mathit{not}\, c_{i,2}, \mathit{not}\, c_{i,3}\}, \end{aligned}$$

$P = R \cup \{b \leftarrow \mathit{not}\, a\}$, and

$Q = R \cup \{a \leftarrow \mathit{not}\, b\}$.

Both P and Q are acyclic and constructible in polynomial time from (C, S). It can be shown, that there exists an EHS for (C, S) iff $P \not\equiv_s Q$. □

In case of checking uniform equivalence, this task is coNP-hard as soon as negation is involved:

Lemma 5 *Deciding $P \equiv_u Q$ is* coNP*-hard for acyclic constraint-free programs, even if Q is Horn.*

Proof. Again, take (C, S) from the proof of Lemma 3. Define now program P as containing, for any $1 \leq i \leq n$,

$$a \leftarrow c_{i,1}, c_{i,2}, \quad a \leftarrow c_{i,1}, c_{i,3}, \quad a \leftarrow c_{i,2}, c_{i,3}, \tag{3}$$

$$a \leftarrow \mathit{not}\, c_{i,1}, \mathit{not}\, c_{i,2}, \mathit{not}\, c_{i,3}. \tag{4}$$

We show that there exists an EHS for (C, S) iff $P \not\equiv_u \{a \leftarrow\}$.

Suppose there is a set F of facts such that $\mathcal{AS}(P \cup F) \neq \{F \cup \{a\}\}$. Hence, $a \notin F$, and thus, without loss of generality, we can assume $F \subseteq S$. Since a is not derived in $P \cup F$, we must have, for each $1 \leq i \leq n$, $|\{c_{i,1}, c_{i,2}, c_{i,3}\} \cap F| \leq 1$ (by Rules (3)) and $|\{c_{i,1}, c_{i,2}, c_{i,3}\} \cap F| > 0$ (by Rules (4)). But then, F is an EHS.

For the only-if direction, one shows that adding any EHS S' of (C, S) to P yields an answer set S' of $P \cup S'$. Since a does not occur in C, we have $a \notin S'$. But $S' \cup \{a \leftarrow\}$ in turn has $S' \cup \{a\}$ as its only answer set. Thus, $P \cup S'$ is not equivalent to $S' \cup \{a \leftarrow\}$, hence $P \not\equiv_u \{a \leftarrow\}$. □

Since all established hardness results carry over to stratified programs, we eventually obtain a full picture of the complexity of deciding $P \equiv_e Q$ for $e \in \{u, s\}$ between programs of the considered classes. Note that restrictions to stratified and even to acyclic programs did not lower the complexity.

Theorem 1 *The complexity of deciding strong* (*resp., uniform*) *equivalence between normal propositional programs is summarized in Table 1. All entries are completeness results, which hold also for stratified and acyclic normal programs.*

We complement these results by giving complexity bounds for ordinary equivalence.

Theorem 2 *Deciding ordinary equivalence between normal stratified programs is* P*-complete. Hardness holds already for acyclic definite Horn programs. Moreover, if one of the programs is arbitrary normal, the problem is* coNP*-complete.*

Structure-Preserving Notions of Equivalence An important observation in case of strong equivalence is that the added rules may destroy syntactical properties of the programs under comparison, which may be unwanted. Given two acyclic (resp., stratified) programs P and Q, it is natural to compare the two programs under any *admissible extension*, that is to say, only under programs which do not introduce cycles (resp., cycles involving at least one negation) when joined with P or Q. To the best of our knowledge, equivalence notions which are sensitive to the compared programs have not been introduced so far, with the notable exception of strong equivalence between prioritized logic programs [Faber and Konczak, 2005].

Definition 1 *Programs P and Q are* acyclicity-preserving (*resp.,* stratification-preserving) strongly equivalent*, denoted $P \equiv_s^{ap} Q$ (resp., $P \equiv_s^{sp} Q$), iff, for each program R such that both $P \cup R$ and $Q \cup R$ are acyclic* (*resp., stratified*)*, $P \cup R$ and $Q \cup R$ possess the same answer sets.*

For programs without negations, $P \equiv_s^{sp} Q$ is clearly the same as $P \equiv_s Q$. Moreover, from known results, we already have that in this case $P \equiv_s Q$ equals $P \equiv_u Q$. Thus, also $P \equiv_s^{ap} Q$ collapses to uniform and strong equivalence for positive programs. Indeed, since strong and uniform equivalence coincide, $P \not\equiv_s Q$ is witnessed by some set F of facts. However, F does not create cycles if added to P and Q.

In general, the relation $\equiv_s^{sp}$ differs from strong equivalence, as illustrated by the following example: Consider $P = \{a \leftarrow not\, b;\ a \leftarrow b\}$ and $Q = \{a \leftarrow not\, c;\ a \leftarrow c\}$. We have $P \equiv_u Q$ but $P \not\equiv_s Q$. However, $P \equiv_s^{sp} Q$ holds since every counterexample for $P \equiv_s Q$ destroys the structure of at least one of the programs.

Theorem 3 *Deciding $P \equiv_s^{ap} Q$ (resp., $P \equiv_s^{sp} Q$) is* coNP*-hard for constraint-free programs P and Q, even if Q is Horn.*

Proof (Sketch). We use the same construction as in the proof of Lemma 5, i.e., an exact hitting set problem (C, S) and the corresponding program P. One can show that there exists an EHS for (C, S) iff $P \not\equiv_s^i \{a \leftarrow\}$, for $i \in \{ap, sp\}$. □

4 Results for the General Non-Ground Case

In this section, we first review known results for the non-ground case, and then provide new results revealing the following insights: (i) Contrary to the propositional case, we prove that for non-ground programs the complexity of deciding strong and uniform equivalence decreases for acyclic programs in certain cases; and (ii) we show a new undecidability result for uniform equivalence, strengthening an earlier result [Eiter *et al.*, 2005a] given for disjunctive programs to the case of normal programs.

Let us first review the complexity of ordinary equivalence. As follows from Dantsin *et al.* [2001], deciding $P \equiv_o Q$ is

- EXPTIME-complete for Horn programs P, Q; and
- co-NEXPTIME-complete for normal programs P, Q.

Furthermore, it is well known that the program complexity for acyclic Horn programs (without constraints) is PSPACE-complete (cf., e.g., again Dantsin *et al.* [2001]). Hence, deciding ordinary equivalence between acyclic Horn programs can easily be shown to be PSPACE-complete.

Strong Equivalence Note that for (acyclic) Horn programs it is known that strong and uniform equivalence coincide with classical equivalence, which is refutable in nondeterministic polynomial time using an oracle for the ordinary equivalence of (acyclic) Horn programs, and thus feasible in EXPTIME (resp., in PSPACE for acyclic programs). In turn, classical equivalence is also hard for these classes, as shown by a reduction of deciding inference of an atom from a program.

Theorem 4 *Deciding $P \equiv_e Q$, where $e \in \{o, u, s\}$, is* EXPTIME*-complete given that P, Q are Horn, and* PSPACE*-complete given that P, Q are acyclic and Horn. Moreover, in each case, hardness already holds for definite programs.*

Let us now consider normal programs. For arbitrary such programs, deciding strong equivalence is in co-NEXPTIME [Lin, 2002] as well as co-NEXPTIME-hard [Eiter *et al.*, 2005a]. By suitable modifications, the hardness proof can be strengthened to acyclic programs. We therefore obtain:

Theorem 5 *Deciding $P \equiv_s Q$ is* co-NEXPTIME*-complete for acyclic* (*stratified*) *normal programs P, Q. Hardness even holds for constraint-free programs.*

Uniform Equivalence As shown next, uniform equivalence between normal programs is undecidable. The proof reduces query equivalence between Horn programs [Shmueli, 1987] to uniform equivalence. Query equivalence (with respect to a predicate q) is the problem of deciding, given Horn programs P and Q over extensional predicates $\mathcal{E}$ and intensional predicates $\mathcal{I}$, whether for any finite database (i.e., set of facts) D for $\mathcal{E}$, it holds that $P \cup D$ and $Q \cup D$ derive the same facts for q. We use *linear* programs here (i.e., programs with at most one intensional predicate in the rule bodies) for which query equivalence remains undecidable [Feder and Saraiya, 1992], and adapt a construction due to Eiter *et al.* [2005a].

First, we reduce query equivalence to *program equivalence* (i.e., to the test whether $P \cup D$ and $Q \cup D$ have the same answer sets for every finite database D) as follows: Define

$$P^* = P \cup Q' \cup \{\, p^*(X) \leftarrow p(X) \,\} \quad \text{and}$$
$$Q^* = P \cup Q' \cup \{\, p^*(X) \leftarrow p'(X) \,\},$$

where Q' results from Q by replacing each intensional predicate symbol i by i' and p^* is a fresh predicate which refers to the query predicate p. Then, P and Q are query equivalent with respect to p iff P^* and Q^* are program equivalent.

We now map the problem of program equivalence between (linear) Horn programs to uniform equivalence between normal programs. The basic idea is that the computation of *minimal* models of programs is transformed to the computation of certain *maximal* models of program reducts, viz. so-called

UE-models, characterizing uniform equivalence [Eiter *et al.*, 2005a].

In what follows, given a linear Horn program P over extensional predicates $\mathcal{E}_P$ and intensional predicates $\mathcal{I}_P$, let d be a new unary predicate, w be a new propositional atom, and, for each rule $r \in P$, denote by

- E_r the list of atoms in $B(r)$ having a predicate from $\mathcal{E}_P$,
- D_r the list $d(X_1), \ldots, d(X_m)$, where $\mathcal{V}_{I_r} \setminus (\mathcal{V}_{H(r)} \cup \mathcal{V}_{E_r}) = \{X_1, \ldots, X_m\}$, and
- I_r the intensional predicate in $B(r)$, if one exists, otherwise let $I_r = w$.

Let now $\mathcal{A}^*_P = \mathcal{A}_P \cup \bar{\mathcal{E}}_P$, where $\bar{\mathcal{E}}_P = \{\bar{e} \mid e \in \mathcal{E}_P\}$ are new predicate symbols, and define $P^{\hookrightarrow}$ as follows:

$$\{w \leftarrow H(r) \mid r \in P,\ B(r) = \emptyset\} \cup \quad (5)$$
$$\{I_r \leftarrow E_r, H(r), D_r \mid r \in P,\ B(r) \neq \emptyset\} \cup \quad (6)$$
$$\{d(c) \leftarrow \mid c \in \mathcal{C}_P\} \cup \quad (7)$$
$$\{d(X_i) \leftarrow p(X_1, \ldots, X_n) \mid p \in \mathcal{A}^*_P,\ 1 \leq i \leq n\} \cup \quad (8)$$
$$\{p(X_1, \ldots, X_n) \leftarrow w, d(X_1), \ldots, d(X_n) \mid p \in \mathcal{A}^*_P\} \cup \quad (9)$$
$$\{w \leftarrow e(X_1, \ldots, X_n), \bar{e}(X_1, \ldots, X_n) \mid e \in \mathcal{E}_P\} \cup \quad (10)$$
$$\{\leftarrow not\ w\}. \quad (11)$$

Observe that $P^{\hookrightarrow}$ is normal and contains a single negation. The forthcoming result relies on the observation that two Horn programs P and Q over the same domain (i.e., such that $\mathcal{C}_P = \mathcal{C}_Q$) are program equivalent iff $P^{\hookrightarrow} \equiv_u Q^{\hookrightarrow}$. Together with the result of Feder and Saraiya [1992], we thus obtain:

Theorem 6 *Deciding $P \equiv_u Q$ between normal programs P and Q is undecidable. This holds already for definite Horn programs augmented by a single negative constraint.*

We remark that the problem becomes decidable if one of the programs, say Q, is negation-free and the other has negation only in negative constraints (possibly augmented with "domain predicates" for guaranteeing safety). Intuitively, by monotonicity of Q, we can then disregard all the negative constraints in P if Q is not inconsistent, while otherwise (if Q is inconsistent) we obtain $P \not\equiv_u Q$.

5 Restricted Cases

Finally, we consider results for equivalence checking in case some parameter associated with the compared programs is fixed by a constant. Due to space reasons, the discussion here is rather succinct and informal. A detailed formal account of the results will be given in the full version of the paper.

Few predicates In the propositional case, a fixed number of atoms clearly makes all considered equivalence problems tractable. For the non-ground case, the traditional technique for rewriting arbitrary programs to programs over a single predicate can be shown to be faithful with respect to strong equivalence. Hence, all the hardness (and undecidability) results for the non-ground case carry over to programs over a few predicate symbols.

Small programs For *small* programs, i.e., programs with a constant bound on the number of rules, again the propositional case shows decreasing complexity, i.e., given a fixed integer k, deciding $P \equiv_e Q$ between programs with at most k rules, for $e \in \{o, s, u\}$, is feasible in polynomial time.

The result in the non-ground setting is obtained from the programs proving the undecidability of query equivalence between definite linear Horn programs [Feder and Saraiya, 1992]. While these programs, say P and Q, have only three rules, the number of predicates and constants is not bounded. Thus, the programs $P^{\hookrightarrow}$ and $Q^{\hookrightarrow}$, used to prove Theorem 6, are not small yet. While the number of predicate symbols in the original programs is easily bound (which in turn bounds the number of rules in Sets (9) and (10)), Sets (7) and (8) still introduce an unbounded number of rules. However, both sets can be fixed to have just three rules using the following *argument rotation* technique: replace (7) by the three rules

$$d'(c_1, \ldots, c_k) \leftarrow,$$
$$d'(X_1, \ldots, X_k) \leftarrow d'(X_2, \ldots, X_k, X_1),$$
$$d(X_1) \leftarrow d'(X_1, \ldots, X_k),$$

where $c_1, \ldots, c_k$ are all constants occurring in P. For each $p \in \mathcal{A}^*_P$, let p' be a new predicate and replace (8) in $P^{\hookrightarrow}$ by

$$p'(X_1, \ldots, X_n) \leftarrow p(X_1, \ldots, X_n),$$
$$p'(X_1, \ldots, X_n) \leftarrow p'(X_2, \ldots, X_1, X_n),$$
$$d(X_1) \leftarrow p(X_1, \ldots, X_n),$$

where n is the arity of p. With theses modifications, undecidability can be shown as before.

Theorem 7 *$P \equiv_u Q$ is undecidable for small programs P, Q.*

Another method to restrict the number of rules is due to Gottlob and Papadimitriou [2003], who show how Horn programs can be faithfully transformed into *single-rule programs*. They also prove that logical implication between two single Horn rules is EXPTIME-hard. From this result and the above argument rotation technique, we can derive:

Theorem 8 *Deciding $P \equiv_e Q$, where $e \in \{s, u, o\}$, is* EXPTIME*-complete for small Horn programs P, Q.*

Bounded Predicates Arities Finally, we briefly discuss bounding the predicate arities by a constant. Checking strong and ordinary equivalence then become significantly easier. Note that checking answer-set existence is Σ^P_2-complete in this setting [Eiter *et al.*, 2004a]. Indeed, the size of the Herbrand base B_P of a program P is polynomial in the size of P, and classical model checking (i.e., deciding $I \models P$ given P and $I \subseteq B_P$) is in coNP (note that the grounding of P can still be exponential). For strong equivalence, it is sufficient to consider an extended Herbrand base $B_{\mathcal{A}_P, \mathcal{C}}$, where $\mathcal{C}$ augments $\mathcal{C}_P$ by polynomially many new constant symbols (this readily follows from proofs given by Eiter *et al.* [2005a]).

In case of strong equivalence and ordinary equivalence, we thus obtain that equivalence checking between normal programs is on the second level of the polynomial hierarchy. For uniform equivalence, a similar restriction of the domain seems not possible. In fact, in case of disjunctive programs,

uniform equivalence of stratified programs with constraints remains undecidable even under bounded predicate arities (which follows from the proof given by Eiter *et al.* [2005a]). A complete picture with respect to different classes of programs in this setting is subject of future work.

Few negations A further topic for future work is concerned with programs having a fixed number of negations. We expect that all considered equivalence notions are tractable in the propositional case, while for the non-ground setting, Theorem 6, for instance, shows that adding a single negation to Horn programs makes uniform equivalence undecidable.

6 Conclusion

We presented a number of complexity results about equivalence testing for the important classes of stratified and acyclic programs, which have not been studied so far, in the possible presence of constraints. We have also considered the effect of various syntactic restrictions on programs relevant in practice. The results reveal that, already in the propositional case, deciding strong or uniform equivalence is intractable for simple programs. Similarly, in the non-ground case, the problems are as difficult as for arbitrary normal logic programs, and uniform equivalence is undecidable with a single negative constraint. However, they become (much) easier under certain restrictions. Therefore, our results provide useful insights for the development of algorithms and implementations of equivalence checkers and for program optimization.

While the picture is complete for the propositional case, in the non-ground case several issues remain open. One is the question of (un)decidability of uniform equivalence between stratified programs in the absence of constraints. While decidability for monadic programs follows from results by Halevy *et al.* [2001] and is known for programs with joint stratification [Levy and Sagiv, 1993], the general case is open. Similarly, the effect of bounded predicate arities remains to be clarified. Finally, the study of further notions of equivalence, such as relativized versions of equivalence [Eiter *et al.*, 2005b], is subject to future work. Clearly, relativized equivalence is undecidable for very limited fragments, as follows from results on program equivalence. Nevertheless, our results provide lower complexity bounds and may help to derive characterizations for relativized equivalence in the non-ground case, which have not been formulated so far.

References

[Baral, 2003] C. Baral. *Knowledge Representation, Reasoning and Declarative Problem Solving*. CUP, 2003.

[Ben-Eliyahu and Dechter, 1994] R. Ben-Eliyahu and R. Dechter. Propositional Semantics for Disjunctive Logic Programs. *AMAI*, 12:53–87, 1994.

[Dantsin *et al.*, 2001] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and Expressive Power of Logic Programming. *ACM CSUR*, 33(3):374–425, 2001.

[Eiter and Fink, 2003] T. Eiter and M. Fink. Uniform Equivalence of Logic Programs under the Stable Model Semantics. In *Proc. ICLP 2003*, pp. 224–238, LNCS 2916.

[Eiter *et al.*, 2004a] T. Eiter, W. Faber, M. Fink, G. Pfeifer, and S. Woltran. Complexity of Answer Set Checking and Bounded Predicate Arities for Non-ground Answer Set Programming. In *Proc. KR 2004*, pp. 377–387, 2004.

[Eiter *et al.*, 2005a] T. Eiter, M. Fink, H. Tompits, and S. Woltran. Strong and Uniform Equivalence in Answer-Set Programming: Characterizations and Complexity Results for the Non-Ground Case. In *Proc. AAAI 2005*, pp. 695–700.

[Eiter *et al.*, 2005b] T. Eiter, M. Fink, and S. Woltran. Semantical Characterizations and Complexity of Equivalences in Stable Logic Programming. *ACM TOCL*. Forthcoming.

[Faber and Konczak, 2005] W. Faber and K. Konczak. Strong Equivalence for Logic Programs with Preferences. In *Proc. IJCAI 2005*, pp. 430–435.

[Feder and Saraiya, 1992] T. Feder and Y.P. Saraiya. Decidability and Undecidability of Equivalence for Linear Datalog with Applications to Normal-Form Optimizations. In *Proc. ICDT 1992*, pp. 297–311, LNCS 646.

[Gelfond and Leone, 2002] M. Gelfond and N. Leone. Logic Programming and Knowledge Representation - The A-Prolog Perspective. *Artificial Intell.*, 138(1-2):3–38, 2002.

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

[Goldschlager, 1977] L. M. Goldschlager. The Monotone and Planar Circuit Value Problems are Complete for P. *SIGACT News*, 9, 1977.

[Gottlob and Papadimitriou, 2003] G. Gottlob and C. Papadimitriou. On the Complexity of Single-Rule Datalog Queries. *Inf. Comput.*, 183(1):104–122, 2003.

[Halevy *et al.*, 2001] A.Y. Halevy, I.S. Mumick, Y. Sagiv, and O. Shmueli. Static Analysis in Datalog Extensions. *JACM*, 48(5):971–1012, 2001.

[Levy and Sagiv, 1993] A.Y. Levy and Y. Sagiv. Queries Independent of Updates. In *Proc. VLDB 1993*, pp. 171–181.

[Lifschitz *et al.*, 2001] V. Lifschitz, D. Pearce, and A. Valverde. Strongly Equivalent Logic Programs. *ACM TOCL*, 2(4):526–541, 2001.

[Lin, 2002] F. Lin. Reducing Strong Equivalence of Logic Programs to Entailment in Classical Propositional Logic. In *Proc. KR 2002*, pp. 170–176.

[Niemelä, 1999] I. Niemelä. Logic Programs with Stable Model Semantics as a Constraint Programming Paradigm. *AMAI*, 25(3,4), 1999.

[Pearce and Valverde, 2004] D. Pearce and A. Valverde. Uniform Equivalence for Equilibrium Logic and Logic Programs. In *Proc. LPNMR 2004*, LNCS 2923.

[Shmueli, 1987] O. Shmueli. Decidability and Expressiveness Aspects of Logic Queries. In *Proc. PODS 1987*, pp. 237–249.