Strong and Uniform Equivalence in Answer-Set Programming: Characterizations and Complexity Results for the Non-Ground Case*

Thomas Eiter, Michael Fink, Hans Tompits, and Stefan Woltran
Institut für Informationssysteme 184/3,
Technische Universität Wien,
Favoritenstraße 9-11, A-1040 Vienna, Austria
e-mail: {eiter, michael, tompits, stefan}@kr.tuwien.ac.at

Abstract

Recent research in nonmonotonic logic programming under the answer-set semantics studies different notions of equivalence. In particular, strong and uniform equivalence are proposed as useful tools for optimizing (parts of) a logic program. While previous research mainly addressed propositional (i.e., ground) programs, we deal here with the more general case of non-ground programs, and provide semantical characterizations capturing the essence of equivalence, generalizing the concepts of SE-models and UE-models, respectively, as originally introduced for propositional programs. We show that uniform equivalence is undecidable, and we give decidability results and precise complexity bounds for strong equivalence (thereby correcting a previous complexity bound for strong equivalence from the literature) as well as for uniform equivalence for finite vocabularies.

Introduction

Answer-set programming (ASP) has been recognized as a promising declarative problem solving paradigm, in which problems are encoded in nonmonotonic logic programs. From their answer sets (models) computed with an answer-set solver such as DLV (Leone *et al.* 2002), Smodels (Simons, Niemelä, & Soininen 2002), ASSAT (Lin & Zhao 2002), the solutions to a given problem are then extracted. This paradigm has been successfully applied to many areas including planning, diagnosis, information integration, and security analysis, to mention just a few.

As for comparing and optimizing (parts of logic) programs, recent research in ASP studies different notions of equivalence, among which strong and uniform equivalence are most prominent (Lifschitz, Pearce, & Valverde 2001; Eiter & Fink 2003). Roughly, two programs P and Q are strongly equivalent, if any extensions $P \cup R$ and $Q \cup R$ by further rules R have the same answer sets; uniform equivalence is similarly defined but R must consist of facts only. While these notions have been studied for propositional (ground) programs in several papers, the case of programs with variables was only studied by Lin (2002), and recently by Pearce & Valverde (2004), for strong equivalence so far. The non-ground case, however, is for ASP the more important one, since program encodings in practice use variables.

In the presence of variables, it is necessary to specify the particular *domain* the variables are assumed to range over. Here, we consider both the case of finite domains as well as the case of infinite domains. Note that, when comparing programs, taking infinite domains into account is desirable from a practical point of view. Assume, for instance, programs implementing the 3-coloring problem of graphs. In general, one wants to compare these programs with respect to their behavior on *any* graph. This motivates results about ground programs obtained from finite selections from a possibly infinite domain.

Our main contributions can be summarized as follows.

(1) We provide semantical characterizations capturing the essence of strong and uniform equivalence. To this end, we generalize the concepts of *SE-models* and *UE-models* introduced for propositional programs (Turner 2003; Eiter & Fink 2003).

(2) We give decidability results and precise complexity bounds for strong equivalence. In that, we correct the previous result of coNP-completeness for deciding strong equivalence (Lin 2002), which unfortunately is exponentially harder (viz. co-NEXPTIME-complete) in the general case; coNP-membership holds in case of a fixed upper bound on the number of variables occurring in the rules of the program.

(3) We show that uniform equivalence of programs is undecidable in general. This is rather bad news, since uniform equivalence has been pointed out as a decidable relaxation of the equivalence of two Horn datalog queries P and Q on a database R (i.e., R is restricted to facts on input relations) (Sagiv 1988). In fact, undecidability holds even if only one negative literal occurs in the programs, while for negation-free (possibly disjunctive) programs, the problem is decidable.

(4) Finally, we show that for finite domains, uniform equivalence is decidable, but has higher complexity than strong equivalence.

Our results shed further light on strong and uniform equivalence, and indicate that further efforts will be needed

*This work was supported by the Austrian Science Fund under grant P18019, and by the EC via projects FET-2001-37004 WASP, IST-2001-33570 INFOMIX, and IST-2001-33123 CologNeT.

to single out algorithms for decidable classes of uniform equivalence, similar as for equivalence between datalog queries (Halevy *et al.* 2001). Indeed, while strong equivalence is decidable, it seems to be far too restrictive for ASP practice.

Preliminaries

To begin, we introduce the basic concepts of logic programs under the answer-set semantics.

Logic programs are formulated in a language $\mathcal{L}$ containing a set $\mathcal{A}$ of *predicate symbols*, a set $\mathcal{V}$ of *variables*, and a set $\mathcal{C}$ of *constants*. Each predicate symbol has an associated *arity* $n \geq 0$; the set $\mathcal{C}$ is also referred to as the *domain* of $\mathcal{L}$.

An *atom* (over $\mathcal{L}$) is an expression of form $p(t_1, \ldots, t_n)$, where $p \in \mathcal{A}$ is a predicate of arity n and $t_i \in \mathcal{C} \cup \mathcal{V}$, for $1 \leq i \leq n$. An atom is *ground* if no variable occurs in it. The set of all ground atoms over language $\mathcal{L}$ is called the *Herbrand base* of $\mathcal{L}$, denoted $B_{\mathcal{L}}$. For a set $A \subseteq \mathcal{A}$ of predicate symbols and a set $C \subseteq \mathcal{C}$ of constants, we also write $B_{A,C}$ to denote the set of all ground atoms constructed from the predicate symbols from $A \subseteq \mathcal{A}$ and the constants from C.

A (*disjunctive*) *rule* (over $\mathcal{L}$), r, is of the form

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

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" is *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, we define $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$.

A rule r of form (1) is called (i) a *fact*, if $m = 0$ and $n = 1$ (in which case the symbol $\leftarrow$ is usually omitted), (ii) a *constraint*, if $n = 0$, (iii) *normal*, if $n \leq 1$, (iv) *positive*, if $k = m$, and (v) *Horn*, if $k = m$ and $n \leq 1$. A rule r is *safe* if each variable occurring in $H(r) \cup B^-(r)$ also occurs in $B^+(r)$; r is *ground*, if all atoms occurring in it are ground.

By a *program* (over $\mathcal{L}$) we understand a set of rules (over $\mathcal{L}$). Programs are normal (resp., positive, Horn, ground, safe) if all of their rules enjoy this property.

To each program P we assign a language $\mathcal{L}(P)$ consisting of all variables, constants, and predicates occurring in P. We refer to the Herbrand base $B_{\mathcal{L}(P)}$ of $\mathcal{L}(P)$ as the Herbrand base of P, symbolically B_P. Furthermore, the set of all constants occurring in P is called the *Herbrand universe* of P, symbolically U_P. If no constant appears in P, then $U_P = \{c\}$, for an arbitrary constant c. The set of all predicates occurring in P is denoted by $\mathcal{A}_P$.

Let $\mathcal{L}$ be a language with a set $\mathcal{A}$ of predicates and domain $\mathcal{C}$, and let $C \subseteq \mathcal{C}$. For any program P over $\mathcal{L}$, we let $B_{P,C}$ stand for $B_{\mathcal{A}_P,C}$, i.e., $B_{P,C}$ is the set of all ground atoms constructed from the predicates occurring in P and the constants in C. Furthermore, given a rule r over $\mathcal{L}$, we define $grd(r, C)$ as the set of all rules obtained from r by all possible substitutions of elements of C for the variables in r. Moreover, for any program P over $\mathcal{L}$, the *grounding of P with respect to C* is given by $grd(P, C) = \bigcup_{r \in P} grd(r, C)$. In particular, $grd(P, U_P)$ is referred to as the *grounding of P* simpliciter, written $grd(P)$.

By an *interpretation* we understand a set of ground atoms. 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 iff each $r \in P$ is satisfied by I. The *Gelfond-Lifschitz reduct* (Gelfond & Lifschitz 1991) of a ground program P (with respect to an interpretation I) is given by

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

A set $I \subseteq B_P$ is an *answer set* of P iff I is a subset-minimal set satisfying $grd(P)^I$. The set of all answer sets of P is denoted by $\mathcal{AS}(P)$. Unless stated otherwise, from now we assume that programs are finite and safe.

Notions of Equivalence

We now introduce the basic notions of equivalence which will be relevant for our purposes. Besides the central concepts of *strong* and *uniform equivalence*, we also make use of the weaker concept of *ordinary equivalence*, the ancillary notion of *classical equivalence*, and equivalence notions studied in the context of datalog programs. We start with classical equivalence.

Definition 1 *Two programs, P and Q, are* classically equivalent*, symbolically $P \Leftrightarrow Q$, iff $grd(P)$ and $grd(Q)$, interpreted as propositional formulas, are equivalent in classical propositional logic.*

Deciding whether two programs are classically equivalent is well-known to be EXPTIME-complete for Horn programs and co-NEXPTIME-complete for general programs (Dantsin *et al.* 2001).

Definition 2 *Two programs, P and Q, are* ordinarily equivalent*, symbolically $P \equiv Q$, iff $\mathcal{AS}(P) = \mathcal{AS}(Q)$.*

The complexity of deciding ordinary equivalence between two programs is known to range from EXPTIME-completeness to co-NEXPTIME$^{\text{NP}}$-completeness (Dantsin *et al.* 2001), depending on the syntactical restrictions on the considered programs.

As a strengthening of ordinary equivalence, Lifschitz, Pearce, & Valverde (2001) proposed the notion of *strong equivalence* for propositional programs under the answer-set semantics. This concept was actually preceded by a similar term introduced by Maher (1988) in the context of negation-free datalog programs as "equivalence of programs segments". There are now different ways to define strong equivalence for non-ground programs; we propose the following one:

Definition 3 *Let $\mathcal{L}$ be a language and P, Q two programs over $\mathcal{L}$. Then, P and Q are* strongly equivalent*, symbolically $P \equiv_s Q$, iff $\mathcal{AS}(P \cup R) = \mathcal{AS}(Q \cup R)$, for any (possibly infinite) program R over $\mathcal{L}$.*

A slightly different notion of strong equivalence for non-ground programs has been discussed by Lin (2002)—we say more on that notion later on.

In specializing the extension programs R of Definition 3 to programs containing *facts* only, we arrive at the notion of *uniform equivalence*, originally studied by Sagiv (1988) for datalog programs.

Definition 4 *Let $\mathcal{L}$ be a language and P, Q two programs over $\mathcal{L}$. Then, P and Q are* uniformly equivalent, *symbolically $P \equiv_u Q$, iff $\mathcal{AS}(P \cup F) = \mathcal{AS}(Q \cup F)$, for any finite set $F \subseteq B_{\mathcal{L}}$ of ground facts.*

As final equivalence notions, we introduce *query equivalence* and *program equivalence*, which are the central equivalence relations for datalog programs. To define these relations, the following concepts are required: Call a predicate occurring in a program P *extensional* (in P) if it does not occur in any rule head (in P), and *intensional* otherwise. Let $\mathcal{E}_P$ be the set of all extensional predicates in P.

Definition 5 *Let $\mathcal{L}$ be a language with domain $\mathcal{C}$ and P, Q two programs over $\mathcal{L}$. Then, P and Q are* query equivalent *with respect to a predicate p iff, for any finite set $S \subseteq B_{\mathcal{E}_{P\cup Q},\mathcal{C}}$, $\{I \cap B_{\{p\},\mathcal{C}} \mid I \in \mathcal{AS}(P \cup S)\} = \{I \cap B_{\{p\},\mathcal{C}} \mid I \in \mathcal{AS}(Q \cup S)\}$. Similarly, P and Q are* program equivalent *iff, for any S as above, $\mathcal{AS}(P \cup S) = \mathcal{AS}(Q \cup S)$ holds.*

Query equivalence was shown to be undecidable for Horn programs over infinite domains (Shmueli 1987). This result extends to program equivalence 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 program equivalent iff P and Q are query equivalent with respect to p.

Characterizations

In this section, we provide model-theoretic characterizations of strong and uniform equivalence between non-ground programs, generalizing previous characterizations for propositional programs (Turner 2003; Eiter & Fink 2003). The following definition is central:

Definition 6 *Let $\mathcal{L}$ be a language with domain $\mathcal{C}$ and set $\mathcal{A}$ of predicates, P a program over $\mathcal{L}$, $C \subseteq \mathcal{C}$, and let $X, Y \subseteq B_{\mathcal{A},C}$ be sets of ground atoms such that $X \subseteq Y$. Then, $(X,Y)_C$ is an* SE-model *of P iff $Y \models grd(P,C)$ and $X \models grd(P,C)^Y$. By $SE(P)$ we denote the set of all SE-models of P.*

The next result rephrases the main result of Turner (2003). Note that, for a ground program P, $grd(P,C) = P$, for any C.

Proposition 7 *Let P, Q be (possibly infinite) ground programs. Then, for any set of ground rules R, $\mathcal{AS}(P \cup R) = \mathcal{AS}(Q \cup R)$ iff $SE(P) = SE(Q)$.*

A pendant to the concept of an SE-model for characterizing uniform equivalence between propositional programs has been put forth by Eiter & Fink (2003). We extend their notion in the following way:

Definition 8 *Let P be a program and $(X,Y)_C \in SE(P)$. Then, $(X,Y)_C$ is a* UE-model *of P iff, for every SE-model $(X',Y)_C$ of P, $X \subset X'$ implies $X' = Y$. By $UE(P)$ we denote the set of all SE-models of P.*

Proposition 9 *Let P, Q be ground programs. Then, $P \equiv_u Q$ iff $UE(P) = UE(Q)$.*

Note that, in contrast to Proposition 7, the characterization for uniform equivalence via UE-models does not apply to infinite programs.

We now show that Propositions 7 and 9 can be extended to non-ground programs as well. To begin with, we need some technical prerequisites. From now on, we assume that programs are defined over a language with domain $\mathcal{C}$.

Lemma 10 *Let P be a program, $C, C' \subseteq \mathcal{C}$ sets of constants such that $C \subseteq C'$, and $Y \subseteq B_{P,C}$. Then, $Y \models grd(P,C)$ iff $Y \models grd(P,C')$.*

The following lemma is a consequence of this result:

Lemma 11 *Let P be a program, $C, C' \subseteq \mathcal{C}$ sets of constants such that $C \subseteq C'$, and $X \subseteq Y \subseteq B_{P,C}$. Then, (i) $(X,Y)_C \in SE(P)$ iff $(X,Y)_{C'} \in SE(P)$, and (ii) $Y \in \mathcal{AS}(grd(P,C))$ iff $Y \in \mathcal{AS}(grd(P,C'))$.*

Theorem 12 *$P \equiv_s Q$ iff $SE(P) = SE(Q)$.*

Proof. The proof is by a reduction to the propositional case.

First, suppose $SE(P) \neq SE(Q)$. Without loss of generality, assume that $(X,Y)_C$ is an SE-model of P but not of Q with $C \supseteq U_{P\cup Q}$, and consider $grd(P,C)$ and $grd(Q,C)$. By definition, $(X,Y)_C \in SE(grd(P,C))$ and $(X,Y)_C \notin SE(grd(Q,C))$. By Proposition 7, there exists a set R of ground rules with $\mathcal{AS}(grd(P,C) \cup R) \neq \mathcal{AS}(grd(Q,C) \cup R)$. By Lemma 11, we can take $C' = C \cup U_{P\cup Q\cup R}$, and $R' = R \cup \{d(c) \mid c \in C'\}$ with a fresh predicate d and get $\mathcal{AS}(grd(P \cup R', C')) \neq \mathcal{AS}(grd(Q \cup R', C'))$. Since $C' = U_{P\cup R'} = U_{Q\cup R'}$, we obtain $\mathcal{AS}(P \cup R') \neq \mathcal{AS}(Q \cup R')$. Hence, $P \not\equiv_s Q$.

Suppose now $P \not\equiv_s Q$, i.e., there exists a set R of rules such that $\mathcal{AS}(grd(P \cup R)) \neq \mathcal{AS}(grd(Q \cup R))$. One can show by Lemmata 10 and 11 that this implies that $\mathcal{AS}(grd(P \cup R, C)) \neq \mathcal{AS}(grd(Q \cup R, C))$, for $C = U_{P\cup Q\cup R}$, and thus $\mathcal{AS}(grd(P,C) \cup grd(R,C)) \neq \mathcal{AS}(grd(Q,C) \cup grd(R,C))$. By Proposition 7, $SE(P) \neq SE(Q)$. □

An important result by Lifschitz, Pearce, & Valverde (2001) shows that strong equivalence between ground programs P and Q reduces to the test whether, for all sets R of unary rules,[1] $\mathcal{AS}(P \cup R) = \mathcal{AS}(Q \cup R)$. Inspecting the proof of Theorem 12, it is easily checked that this property also holds in the non-ground case.

An analogous characterization for uniform equivalence can be obtained by a similar lifting from the propositional case. For space reasons, we omit a proof of this result.

Theorem 13 *Let P and Q be programs. Then, $P \equiv_u Q$ iff, for each finite $C \subseteq \mathcal{C}$ and every interpretation X, Y, $(X,Y)_C \in UE(P)$ iff $(X,Y)_C \in UE(Q)$.*

Finally, we generalize an important result by Eiter & Fink (2003).

Theorem 14 *For positive programs P and Q, the following propositions are equivalent: (i) $P \equiv_s Q$; (ii) $P \equiv_u Q$; and (iii) $P \Leftrightarrow Q$.*

[1] A unary rule is a Horn rule with exactly one atom in the head and at most one atom in the body.

Finite Domains

In the case of a finite underlying domain $\mathcal{C}$, we can as usual view a program P as a compact representation of the propositional program $grd(P, \mathcal{C})$. Note that, therefore, Theorem 13 can be rephrased for finite domains as follows:

Theorem 15 *Let P, Q be programs over $\mathcal{L}$ whose domain is finite. Then, $P \equiv_u Q$ iff $UE(P) = UE(Q)$.*

Compared to the propositional case, a Herbrand interpretation has single-exponential size (i.e., size $O(2^{p(n)})$ for a polynomial $p(n)$) in the size of $\mathcal{C}$, both if $\mathcal{C}$ is represented explicitly or implicitly by, say, a number n such that $\mathcal{C} = \{0, 1, \ldots, n-1\}$.

For strong and uniform equivalence, we thus obtain upper complexity bounds which increase, compared to the propositional complexity, by one exponential: P to EXPTIME, NP to NEXPTIME, co-NP to co-NEXPTIME, Π_2^P to co-NEXPTIME$^{\mathrm{NP}}$, etc. Given that the inference problem of positive, disjunction-free datalog programs is EXPTIME-complete (cf. Dantsin *et al.* (2001)), it is thus of no surprise that the complexity of deciding strong and uniform equivalence has lower bounds which match the exponential analogues of the propositional case.

Theorem 16 *Over a finite domain $\mathcal{C}$, deciding $P \equiv_s Q$ between programs P and Q is* co-NEXPTIME*-complete. Hardness holds even if P is positive and Q is Horn.*

Proof (*Sketch*). We only show the hardness part. We reduce the problem of evaluating a given second-order sentence Φ of the form $\forall \mathbf{P} \exists \mathbf{x} \psi(\mathbf{x})$, where $\mathbf{P} = p_1, \ldots, p_n$ is a list of predicate variables and $\psi(\mathbf{x})$ is a quantifier-free formula in DNF over predicates $\mathbf{P} \cup \mathcal{A}$, with $\mathcal{A}$ consisting of predicate constants including special predicates $succ$, $first$, and $last$ over a given first-order structure $\mathcal{M}$ for $\mathcal{A}$ in which $succ$, $first$, and $last$ are interpreted as a successor predicate and the first and last element of a linear ordering of the universe $|\mathcal{M}|$ of $\mathcal{M}$, respectively, to checking strong equivalence. The former problem is co-NEXPTIME-complete, which follows from normal form results for second-order logic over successor structures (Leivant 1989; Eiter, Gottlob, & Gurevich 1996), and the fact that a conversion into such a normal form is feasible in polynomial time.

Consider now a second-order sentence $\Phi = \forall \mathbf{P} \exists \mathbf{x} \psi(\mathbf{x})$ over $\mathcal{M}$ as discussed above, where $\psi(\mathbf{x}) = \bigvee_{i=1}^{n} \phi_i$ and each ϕ_i is a conjunction of (possibly negated) predicate atoms. For each $p \in \mathbf{P}$ we introduce a new symbol, p', and let ϕ^* denote the disjunction over all predicate literals in ϕ, but replacing $\neg p(\cdot)$ by $p'(\cdot)$. We define

$$
\begin{aligned}
P &= \{a \vee \phi_i^* \leftarrow \mid 1 \leq i \leq n\} \cup \\
&\quad \{\leftarrow p(X_1, \ldots, X_n), p'(X_1, \ldots, X_n) \mid p \in \mathbf{P}\}, \\
Q &= \{a \leftarrow\} \cup \\
&\quad \{\leftarrow p(X_1, \ldots, X_n), p'(X_1, \ldots, X_n) \mid p \in \mathbf{P}\},
\end{aligned}
$$

where a is a new atom, and, for $\mathcal{S}_{\mathcal{M}} = \{succ(c, c'), first(c), last(c) \mid c, c' \in |\mathcal{M}|\}$, define

$$P_{\mathcal{M}} = \{\psi \in \mathcal{S}_{\mathcal{M}} \mid \mathcal{M} \models \psi\} \cup \{\leftarrow \psi \mid \psi \in \mathcal{S}_{\mathcal{M}}, \mathcal{M} \not\models \psi\}.$$

Then, it can be shown that Φ is true iff $(P \cup P_{\mathcal{M}}) \equiv_s (Q \cup P_{\mathcal{M}})$. □

Corollary 17 *Over a finite domain $\mathcal{C}$, deciding $P \equiv_u Q$ between positive programs P and Q is* co-NEXPTIME-*complete. Hardness holds even if Q is Horn.*

Applying above method to the Π_2^P-result for uniform equivalence in the ground case yields the following result:

Theorem 18 *Over a finite domain $\mathcal{C}$, deciding $P \equiv_u Q$ between programs P and Q is* co-NEXPTIME$^{\mathrm{NP}}$*-complete.*

Infinite Domains

Over infinite domains, decidability of equivalence problems for non-ground programs is no longer guaranteed; recall that e.g., query equivalence (cf. Definition 5) is undecidable.

The following proposition is important for strong equivalence. Define, for $Z \subseteq B_{\mathcal{A},\mathcal{C}}$ and $C \subseteq \mathcal{C}$, $Z|_C = Z \cap B_{\mathcal{A},C}$.

Proposition 19 (SE submodels) *For any program P, the class of SE-models of P, $SE(P)$, is closed under submodels, i.e., for any $U_P \subseteq C' \subseteq C$, if $(X, Y)_C \in SE(P)$, then $(X|_{C'}, Y|_{C'})_{C'} \in SE(P)$.*

Proof. Towards a contradiction, assume $U_P \subseteq C' \subseteq C$, $(X, Y)_C \in SE(P)$, and $(X|_{C'}, Y|_{C'})_{C'} \notin SE(P)$. I.e., there is some $r \in grd(P, C')^{Y|_{C'}}$ such that $X|_{C'} \not\models r$. Since $grd(P, C') \subseteq grd(P, C)$, $r \in grd(P, C)$, and since r does not contain any atom from $Y \setminus (Y|_{C'})$, $r \in grd(P, C)^Y$. Moreover, since $X|_{C'} \subseteq X$ and $(X, Y)_C \in SE(P)$, $X \models B(r)$ and $X \models H(r)$ follows. Thus, $X \cap H(r) \neq \emptyset$, and since r is a ground instance over C', i.e., $U_{\{r\}} \subseteq C'$, $X|_{C'} \cap H(r) \neq \emptyset$ holds, a contradiction. □

As a consequence, we can decide strong equivalence between programs P and Q using finite restrictions of the universe. Informally, the argument is that if $(X, Y)_C \in SE(P)$ and $(X, Y)_C \notin SE(Q)$, for any (possibly infinite) C, then there also exists a finite restriction C' of C such that $(X|_{C'}, Y|_{C'})_{C'} \notin SE(Q)$, while $(X|_{C'}, Y|_{C'})_{C'} \in SE(P)$ holds in view of the above result.

Theorem 20 *For programs P and Q, deciding $P \equiv_s Q$ is* co-NEXPTIME-*complete, even if $\mathcal{C}$ is infinite.*

Proof (*Sketch*). We only show the membership part. Assume $P \not\equiv_s Q$. Then, without loss of generality, there is some $(X, Y)_C \in SE(P)$ with $(X, Y)_C \notin SE(Q)$, for some C. Due to Lemma 10, we can assume that $U_{P \cup Q} \subseteq C$. We show that there exists a finite set C' of constants such that $(X|_{C'}, Y|_{C'})_{C'} \in SE(P)$ and $(X|_{C'}, Y|_{C'})_{C'} \notin SE(Q)$. Since $(X, Y)_C \notin SE(Q)$, there exists a ground rule $r \in grd(Q, C)$ such that either $Y \not\models r$ or $X \not\models r^Y$. Consider $C' = U_{P \cup Q} \cup U_{\{r\}}$. Then, C' is finite. Moreover, $r \in grd(Q, C')$ and $r^{Y|_{C'}} = r^Y$, thus either $Y|_{C'} \not\models r$ or $X|_{C'} \not\models r^{Y|_{C'}}$. That is, $(X|_{C'}, Y|_{C'})_{C'} \notin SE(Q)$, while $(X|_{C'}, Y|_{C'})_{C'} \in SE(P)$ by Proposition 19. As a consequence, we can decide $P \equiv_s Q$ by comparing SE-models $(X, Y)_C$, having $C = U_{P \cup Q} \cup C_V$, where C_V is an arbitrary set of different new constants $c_1, \ldots, c_n$ for all variables $X_1, \ldots, X_n$ occurring in $P \cup Q$. Since C is finite, the result follows from Theorem 16. □

This complexity result implies that it is possible to implement corresponding tests by reductions to answer-set programming. We also remark that the SE-submodels property continues to hold in the presence of certain built-in relations which are preserved under submodels as well (e.g., inequalities or a total orderings).

Unfortunately, a property similar to Proposition 19 cannot hold for UE-models, which is a consequence of Theorem 26 below. However, this only comes into play for non-positive programs, as witnessed by Theorem 14. We thus can immediately state the following:

Corollary 21 *For positive programs P and Q, $P \equiv_u Q$ is* co-NEXPTIME*-complete, even if $\mathcal{C}$ is infinite.*

In what follows, we formally prove the undecidability of checking uniform equivalence in the general case. To this end, we shall construct, from a given Horn program P, a disjunctive program that possesses certain SE-models of the form $(X,Y)_C$, $X \neq Y$, such that the interpretations X amount to countermodels of P. Then, minimal models of P correlate to certain maximal SE-models $(X,Y)_C$ of the new program, i.e., to certain UE-models. Via this correspondence, we then reduce program equivalence between Horn programs to uniform equivalence. In what follows, we use D_d in a rule to abbreviate the sequence $d(X_1),\ldots,d(X_n)$, where X_i occurs in the rule but not in its positive body (in order to guarantee safety of rules).

Definition 22 *Let P be a Horn program, $\mathcal{E}$ and $\mathcal{A}_{P_\mathcal{E}}$ sets of predicates, $U_{P_\mathcal{E}}$ a set of constants, and d, w, $\bar{\mathcal{E}} = \{\bar{e} \mid e \in \mathcal{E}\}$ new predicate symbols. The program $P_\mathcal{E}(\mathcal{A}_{P_\mathcal{E}}, U_{P_\mathcal{E}})$ contains, for every $r \in P$, the rule*

(1) $w \leftarrow H(r)$, *if r is a fact, or otherwise*

(2) $\bigvee_{p(X_1,\ldots,X_n)\in B^+(r)} p(X_1,\ldots,X_n) \leftarrow H(r), D_d$.

Additionally, $P_\mathcal{E}(\mathcal{A}_{P_\mathcal{E}}, U_{P_\mathcal{E}})$ contains the following rules:

(3) $\{d(c) \mid c \in U_{P_\mathcal{E}}\}$,
$\{d(X_i) \leftarrow p(X_1,\ldots,X_n) \mid p \in \mathcal{A}_{P_\mathcal{E}} \cup \bar{\mathcal{E}}, 1 \leq i \leq n\}$;

(4) $\{p(X_1,\ldots,X_n) \leftarrow w, D_d \mid p \in \mathcal{A}_{P_\mathcal{E}} \cup \bar{\mathcal{E}}\}$;

(5) $\{e(X_1,\ldots,X_n) \vee \bar{e}(X_1,\ldots,X_n) \leftarrow D_d \mid e \in \mathcal{E}\}$,
$\{w \leftarrow e(X_1,\ldots,X_n), \bar{e}(X_1,\ldots,X_n) \mid e \in \mathcal{E}\}$;

(6) $w \leftarrow \text{not } w$.

We first analyze the SE-models of $P_\mathcal{E}(\mathcal{A}_{P_\mathcal{E}}, U_{P_\mathcal{E}})$.

Lemma 23 *Let P be a Horn program, $\mathcal{E}$ a subset of its extensional predicates, $\mathcal{A}_P \subseteq \mathcal{A}_{P_\mathcal{E}}$, $U_P \subseteq U_{P_\mathcal{E}}$, C a set of constants such that $U_{P_\mathcal{E}} \subseteq C$, and $P_\mathcal{E}(\mathcal{A}_{P_\mathcal{E}}, U_{P_\mathcal{E}})$ as above. Moreover, let $X \subseteq Y \subseteq B_{\mathcal{A}_{P_\mathcal{E}},C}$, and $\bar{X} = B_{P,C} \setminus X$.*

Then, $(X,Y)_C \in SE(P_\mathcal{E}(\mathcal{A}_{P_\mathcal{E}}, U_{P_\mathcal{E}}))$ iff for some $U_{P_\mathcal{E}} \subseteq C'' \subseteq C' \subseteq C$:

- *$Y = B_{\mathcal{A}_{P_\mathcal{E}},C'}$; and if $X \subset Y$, then either:*
- *$X = B_{\mathcal{A}_{P_\mathcal{E}},C''}$ and $C'' \subset C'$; or*
- *the following conditions jointly hold: (a) $w \notin X$, (b) $d(c) \in X$ iff $c \in C''$, (c) for each $e(c_1,\ldots,c_n) \in B_{\mathcal{E},C''}$, either $e(c_1,\ldots,c_n) \in X$ or $\bar{e}(c_1,\ldots,c_n) \in X$, and (d) $\bar{X} \models grd(P,C)$.*

Proof (*Sketch*). For all SE-models $(X,Y)_C$ of the program $P_\mathcal{E}(\mathcal{A}_{P_\mathcal{E}}, U_{P_\mathcal{E}})$, Y is of the form $B_{\mathcal{A}_{P_\mathcal{E}},C'}$, and $U_{P_\mathcal{E}} \subseteq C' \subseteq C$. This is due to Rule (6) (which implies $w \in Y$) and by the Rules (3) and (4). Similarly, for $X \subset Y$, if $w \in X$, and, of course, for a $C'' \subset C'$. For any other SE-model, we conclude that $w \notin X$. Moreover, (b) and (c) are simple consequences of the positive Rules (3) and Rules (5). To see (d), note that X satisfies all ground instances of Rules (1) and (2) iff $\bar{X} \models grd(P,C'')$, by construction (in particular since the corresponding rules are safe). This can be shown by a simple indirect argument. Since $C'' \subseteq C$, Lemma 10 yields $\bar{X} \models grd(P,C)$. □

We next show for certain UE-models $(X,Y)_C$, $X \neq Y$, our intended property, i.e., that they correlate to minimal models of P. In view of Lemma 11, we can fix Y, i.e., we consider SE-models having $Y = B_{\mathcal{A}_{P_\mathcal{E}},C}$.

Lemma 24 *Let P, $\mathcal{E}$, $\mathcal{A}_{P_\mathcal{E}}$,$U_{P_\mathcal{E}}$, C, $P_\mathcal{E}(\mathcal{A}_{P_\mathcal{E}}, U_{P_\mathcal{E}})$, X, Y, and $\bar{X}$ be as in Lemma 23. Furthermore, let $w \notin X$, $Y = B_{\mathcal{A}_{P_\mathcal{E}},C}$, $\bar{\mathcal{E}}_X = \{e(c_1,\ldots,c_n) \mid \bar{e}(c_1,\ldots,c_n) \in X\}$, and $\mathcal{E}_{\bar{X}} \subseteq B_{\mathcal{E},C} \cap \bar{X}$. Consider $X_\mathcal{E} = \bar{\mathcal{E}}_X \cup \mathcal{E}_{\bar{X}}$. Then, $(X,Y)_C \in UE(P_\mathcal{E}(\mathcal{A}_{P_\mathcal{E}}, U_{P_\mathcal{E}}))$ iff $\bar{X} \in \mathcal{AS}(P \cup X_\mathcal{E})$.*

Proof (*Sketch*). Let X and Y be as in the lemma and $(X,Y)_C$ an SE-model of $P_\mathcal{E}(\mathcal{A}_{P_\mathcal{E}}, U_{P_\mathcal{E}})$. It is easily verified that $(X,Y)_C \in UE(P_\mathcal{E}(\mathcal{A}_{P_\mathcal{E}}, U_{P_\mathcal{E}}))$ iff $d(c) \in X$ for all $c \in C$. Due to this, by Lemma 23 and the construction of $\bar{X}$ and $X_\mathcal{E}$, we get for the only-if direction $\bar{X} \models grd(P \cup X_\mathcal{E}, C)$. It remains to show the minimality of $\bar{X}$. Towards a contradiction, suppose $\bar{Z} \subset \bar{X}$ satisfies $grd(P \cup X_\mathcal{E}, C)$. If $p(c_1,\ldots,c_n) \in \bar{X} \setminus \bar{Z}$, for some $p \in \mathcal{E}$, we arrive at a contradiction since neither $p(c_1,\ldots,c_n) \in X$ nor $\bar{p}(c_1,\ldots,c_n) \in X$ follows. If $p \notin \mathcal{E}$, however, it can be shown by Lemma 23 that $(Z,Y)_C$ is an SE-model of $P_\mathcal{E}(\mathcal{A}_{P_\mathcal{E}}, U_{P_\mathcal{E}})$, for $Z = (B_{P,C} \setminus \bar{Z}) \cup \{d(c) \mid c \in C\} \cup (X \cap B_{\bar{\mathcal{E}},C})$, again a contradiction. For the if direction, assume $(Z,Y)_C \in SE(P_\mathcal{E}(\mathcal{A}_{P_\mathcal{E}}, U_{P_\mathcal{E}}))$ and $X \subset Z \subset Y$. It can be verified that for any $p(c_1,\ldots,c_n) \in (Z \setminus X)$, it holds that $p \notin \mathcal{E}$. Thus, $\bar{X}$ and $\bar{Z}$ coincide on $B_{\mathcal{E},C}$ and, hence, $\bar{Z} \models X_\mathcal{E}$. Moreover, by Lemma 23, $\bar{Z}$ satisfies $grd(P,C)$, contradicting $\bar{X} \in \mathcal{AS}(P \cup X_\mathcal{E})$. □

Lemma 25 *For Horn programs P and Q sharing extensional predicates $\mathcal{E}$, let $\bar{P} = P_\mathcal{E}(\mathcal{A}_{P\cup Q}, U_{P\cup Q})$ and $\bar{Q} = Q_\mathcal{E}(\mathcal{A}_{P\cup Q}, U_{P\cup Q})$. Then, $UE(\bar{P}) = UE(\bar{Q})$, for any finite $C \subseteq \mathcal{C}$, iff P and Q are program equivalent.*

Proof (*Sketch*). One can show that two Horn programs, P and Q, are program equivalent iff, for each finite input $D \subseteq B_{\mathcal{E},C}$ such that $U_{P\cup Q} \subseteq C$ and $\mathcal{E}$ is the set of common extensional predicates of P and Q, $\mathcal{AS}(P \cup D) = \mathcal{AS}(Q \cup D)$. Thus, the lemma holds by the following arguments:

If part. Suppose P and Q are not program equivalent, i.e., there exists a (finite) set $X_\mathcal{E} \subseteq B_{\mathcal{E},C}$ such that, say, $\bar{X} \subseteq B_{P,C}$, $\bar{X} \in \mathcal{AS}(P \cup X_\mathcal{E})$, but $\bar{X} \notin \mathcal{AS}(Q \cup X_\mathcal{E})$ for a finite set of constants C. Note that $U_{P\cup Q} \subseteq C$ holds. Consider $Y = B_{\mathcal{A}_{P\cup Q},C}$ and $X = (B_{P,C} \setminus \bar{X}) \cup \{d(c) \mid c \in C\} \cup \{\bar{e}(c_1,\ldots,c_n) \mid e(c_1,\ldots,c_n) \in \bar{X}\}$. By Lemma 24, $(X,Y)_C \in UE(\bar{P})$ but $(X,Y)_C \notin UE(\bar{Q})$ (otherwise $\bar{X} \in \mathcal{AS}(Q \cup X_\mathcal{E})$), i.e., $UE(\bar{P}) \neq UE(\bar{Q})$.

Only-if part. Suppose $UE(\bar{P}) \neq UE(\bar{Q})$, i.e., without loss of generality, assume $(X,Y)_C \in UE(\bar{P})$ such that $(X,Y)_C \notin UE(\bar{Q})$, for some $X \subseteq Y \subseteq B_{\mathcal{A}_{P\cup Q},C}$ and a finite set of constants C. First, note that $U_{P\cup Q} \subseteq C$. Hence, if $w \notin X$, then the claim follows by Lemma 24. Thus, it remains to consider SE-models of form $(X, Y)_C$ such that either $X = Y$ or $X = B_{\mathcal{A}_{P\cup Q}, C'}$, for some $C' \subset C$. By construction, it holds that $U_{\bar{P}_\mathcal{E}} = U_{\bar{Q}_\mathcal{E}} = U_{P\cup Q} \subseteq C'$. Since, furthermore, $\mathcal{A}_{\bar{P}_\mathcal{E}} = \mathcal{A}_{\bar{Q}_\mathcal{E}} = \mathcal{A}_{P\cup Q}$, it is easily verified that $\bar{P}$ and $\bar{Q}$ coincide on SE-models of the above form. For all SE-models of different form, we have that $w \notin X$ and thus they have no influence on their maximality. That is, $UE(\bar{P}) = UE(\bar{Q})$ for SE-models of form $(Y, Y)_C$ or $(B_{\mathcal{A}_{P\cup Q}, C'}, Y)_C$. This proves that if $UE(\bar{P}) \neq UE(\bar{Q})$ for finite C, then P and Q are not program equivalent. □

Since program equivalence is undecidable for Horn programs over an infinite domain, as shown above, we obtain the following result:

Theorem 26 *Given programs P and Q, it is undecidable whether $P \equiv_u Q$ holds.*

We remark that the programs $P_\mathcal{E}(\mathcal{A}_{P_\mathcal{E}}, U_{P_\mathcal{E}})$ constructed above do not contain any constraints, and just a single rule containing default negation (viz. $w \leftarrow \mathrm{not}\ w$). For the purpose of our construction, this rule can equivalently be replaced with the constraint $\leftarrow \mathrm{not}\ w$. The undecidability result of Theorem 26 thus applies to a minimal extension of positive (disjunctive) programs, and reveals that positive disjunctive programs are a maximal decidable class, while a single use of negation might lead to undecidability.

Related Work

Lin (2002) defines a different notion of strong equivalence between two non-ground programs P and Q over $\mathcal{L}$ (which are not necessarily safe) as checking whether, for any set of constants C in $\mathcal{L}$, $\mathcal{AS}(grd(P, C) \cup R) = \mathcal{AS}(grd(Q, C) \cup R)$, for any finite set of ground rules R. Denote this test by $P \equiv_L Q$. Although this is apparently a different concept, it can be shown to coincide with our notion of strong equivalence on the class of safe programs.

Proposition 27 *For (safe) programs P, Q, $P \equiv_s Q$ iff $P \equiv_L Q$.*

There are two basic differences between $\equiv_s$ and $\equiv_L$. The first one is that $\equiv_L$ explicitly reduces a test on non-ground programs to several tests on ground programs, while $\equiv_s$ directly "lifts" the original definition of strong equivalence by Lifschitz, Pearce, & Valverde (2001) to the non-ground case. Therefore, $\equiv_s$ allows for non-ground rules in the possible extensions R, which seems to be more natural from an ASP perspective. To this end, $\equiv_s$ has to employ the notion of answer sets as defined in non-ground ASP which are given over the programs' Herbrand universes. Roughly speaking, this makes the safety condition necessary. For $\equiv_L$, one can refrain from the safety condition, since the definition of answer sets for non-ground programs is circumvented but rather the entire test is reduced to the propositional case.

Lin (2002) showed decidability of $P \equiv_L Q$ claiming that the problem is coNP-complete. Our complexity analysis, together with Proposition 27, however, shows that the complexity of $P \equiv_L Q$ is one exponential higher, viz. co-NEXPTIME-complete.

Finally, we note that a characterization of a notion of strong equivalence, similar to $\equiv_L$, in terms of a first-order variant of the logic of *here-and-there* has recently been discussed by Pearce & Valverde (2004). Extending an analogous situation for the propositional case, it holds that the models in this first-order here-and-there logic, for constant domains, correspond in fact to our concept of SE-models.

References

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 Fink, M. 2003. Uniform equivalence of logic programs under the stable model semantics. In *Proc. ICLP'03*, 224–238.

Eiter, T.; Gottlob, G.; and Gurevich, Y. 1996. Normal forms for second-order logic over finite structures, and classification of NP optimization problems. *Annals of Pure and Applied Logic* 78:111–125.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 9:365–385.

Halevy, A. Y.; Mumick, I. S.; Sagiv, Y.; and Shmueli, O. 2001. Static analysis in datalog extensions. *Journal of the ACM* 48(5):971–1012.

Leivant, D. 1989. Descriptive characterizations of computational complexity. *JCSS* 39:51–83.

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.

Lifschitz, V.; Pearce, D.; and Valverde, A. 2001. Strongly equivalent logic programs. *ACM ToCL* 2(4):526–541.

Lin, F., and Zhao, Y. 2002. ASSAT: Computing answer sets of a logic program by SAT solvers. In *Proc. AAAI'02*, 112–117.

Lin, F. 2002. Reducing strong equivalence of logic programs to entailment in classical propositional logic. In *Proc. KR'02*, 170–176.

Maher, M. J. 1988. Equivalences of logic programs. In Minker (1988). 627–658.

Minker, J., ed. 1988. *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann.

Pearce, D., and Valverde, A. 2004. Towards a first order equilibrium logic for nonmonotonic reasoning. In *Proc. JELIA'04*, 147–160.

Sagiv, Y. 1988. Optimizing datalog programs. In Minker (1988). 659–698.

Shmueli, O. 1987. Decidability and expressiveness aspects of logic queries. In *Proc. PODS'87*, 237–249.

Simons, P.; Niemelä, I.; and Soininen, T. 2002. Extending and implementing the stable model semantics. *Artificial Intelligence* 138:181–234.

Turner, H. 2003. Strong equivalence made easy: Nested expressions and weight constraints. *TPLP* 3(4-5):602–622.