I N F S Y S

R E S E A R C H

R E P O R T

INSTITUT FÜR INFORMATIONSSYSTEME

ABTEILUNG WISSENSBASIERTE SYSTEME

UNIFORM EQUIVALENCE OF LOGIC PROGRAMS UNDER THE STABLE MODEL SEMANTICS

Thomas Eiter Michael Fink

INFSYS RESEARCH REPORT 1843-03-08

MAY 2003; SEPTEMBER 2003

Institut für Informationssysteme
Abtg. Wissensbasierte Systeme
Technische Universität Wien
Favoritenstraße 9-11
A-1040 Wien, Austria
Tel: +43-1-58801-18405
Fax: +43-1-58801-18493
sek@kr.tuwien.ac.at
www.kr.tuwien.ac.at

TECHNISCHE UNIVERSITÄT WIEN

INFSYS RESEARCH REPORT

INFSYS RESEARCH REPORT 1843-03-08, MAY 2003; SEPTEMBER 2003

UNIFORM EQUIVALENCE OF LOGIC PROGRAMS UNDER THE STABLE MODEL SEMANTICS

(Preliminary Report, May 2003; updated September 2003)

Thomas Eiter[1] and Michael Fink[2]

Abstract. In recent research on nonmonotonic logic programming, repeatedly strong equivalence of logic programs P and Q has been considered, which holds if the programs $P \cup R$ and $Q \cup R$ have the same stable models for any other program R. This property strengthens equivalence of P and Q with respect to stable models (which is the particular case for $R = \emptyset$), and has an application in program optimization. In this paper, we consider the more liberal notion of uniform equivalence, in which R ranges only over the sets of facts rather than all sets of rules. This notion, which is well-known, is particularly useful for assessing whether programs P and Q are equivalent as components in a logic program which is modularly structured. We provide semantical characterizations of uniform equivalence for disjunctive logic programs and some restricted classes, and analyze the computational cost of uniform equivalence in the propositional (ground) case. Our results, which naturally extend to answer set semantics, complement the results on strong equivalence of logic programs and pave the way for optimizations in answer set solvers as a tool for input-based problem solving.

Keywords: uniform equivalence, strong equivalence, stable models, answer set semantics, computational complexity, program optimization

---

[1]Institut of Information Systems, Knowledge Bases Systems Group, TU Vienna, Favoritenstraße 9-11, A-1040 Vienna, Austria. E-mail: eiter@kr.tuwien.ac.at.

[2]Institut of Information Systems, Knowledge Bases Systems Group, TU Vienna, Favoritenstraße 9-11, A-1040 Vienna, Austria. E-mail: michael@kr.tuwien.ac.at.

Acknowledgements: This work was partially supported by the Austrian Science Fund (FWF) Project Z29-N04, and the European Commission under projects FET-2001-37004 WASP and IST-2001-33570 INFOMIX.

An extended abstract of this paper appears in: Proceedings 19th International Conference on Logic Programming (ICLP 2003), LNCS, Springer, 2003.

Copyright © 2003 by the authors

Contents

1 Introduction **1**

2 Preliminaries **2**
2.1 Equivalences 3

3 Characterizations of Uniform Equivalence **4**

4 Restricted Classes of Programs **8**
4.1 Positive programs 9
4.2 Head-cycle free programs 10

5 Complexity **11**

6 Extensions **14**

7 Conclusion **16**

A Appendix: Proofs **17**

1 Introduction

In the last years, logic programming with non-monotonic negation, and in particular stable semantics, as a problem solving tool has received increasing attention, which led to application in several fields. To a great deal, this is due to the availability of several advanced implementations of the stable semantics such as Smodels [17], DLV [10], or ASSAT [14]. In turn, the desire of more efficient stable models solvers has raised the need for sophisticated optimization methods by which logic programs can be simplified and processed more efficiently. In this direction, properties of logic programs under the stable semantics have been investigated which may aid in optimization.

A particular useful such property is *strong equivalence* [11, 23]: Two logic programs P_1 and P_2 are strongly equivalent, if by adding any set of rules R to both P_1 and P_2, the resulting programs $P_1 \cup R$ and $P_2 \cup R$ are equivalent under the stable semantics, i.e., have the same set of stable models. Thus, if a program P contains a subprogram Q which is strongly equivalent to a program Q', then we may replace Q by Q', in particular if the resulting program is simpler to evaluate than the original one.

However, strong equivalence is a very restrictive concept. As for optimization, it is not very sensitive to a modular structure of logic programs which naturally emerges by splitting them into layered *components* that receive input from lower layers by facts and in turn output facts to a higher layer [12, 7], nor to the usage of the same logic program to compute solutions over varying inputs given as sets of facts.

In this paper, we study the more liberal notion of *uniform equivalence* [22, 15], which is better suited in this respect: Two logic programs P_1 and P_2 are uniformly equivalent, if by adding any set of *facts* F to both P_1 and P_2, the resulting programs $P_1 \cup F$ and $P_2 \cup F$ have the same set of stable models. Thus, a component C within a program P may be (offline) replaced by a uniformly equivalent set of rules C', provided the global component structure of the program is not affected (which is a simple syntactic check).

That strong equivalence and uniform equivalence are different concepts is illustrated by the following simple example.

Example 1 *Let $P = \{a \vee b\}$ and $Q = \{a \leftarrow not\, b;\ b \leftarrow not\, a\}$. Then P and Q are not strongly equivalent, since $P \cup \{a \leftarrow b;\ b \leftarrow a\}$ has the stable model $\{a, b\}$, which is not a stable model of $Q \cup \{a \leftarrow b;\ b \leftarrow a\}$. However, it can be seen that P and Q are uniformly equivalent.*

Moreover, this holds even for programs without disjunction.

Example 2 *Let $P = \{a \leftarrow not\, b;\ a \leftarrow b\}$ and $Q = \{a \leftarrow not\, c;\ a \leftarrow c\}$. Then, it is easily verified that P and Q are uniformly equivalent. However, they are not strongly equivalent: For $P \cup \{b \leftarrow a\}$ and $Q \cup \{b \leftarrow a\}$, we have that $S = \{a, b\}$ is a stable model of $Q \cup \{b \leftarrow a\}$ but not of $P \cup \{b \leftarrow a\}$.*

While strong equivalence of logic programs under stable semantics has been considered in a number of papers [4, 6, 13, 11, 18, 19, 23, 24], to our knowledge uniform equivalence of has not been considered. Sagiv [22], who coined the name, has studied the property in the context of definite Horn datalog programs, where he showed decidability of uniform equivalence testing, which contrasts with the undecidability of equivalence testing for datalog programs. Maher [15] considered the property for definite general Horn programs (with function symbols), and reported undecidability. Moreover, both [22, 15] showed that uniform equivalence coincides for the respective programs with Herbrand logical equivalence. Maher also pointed out that the for Datalog programs, this result has been independently established by Cosmadakis and Kanellakis [5].

In this paper we focus on propositional logic programs (to which general programs reduce). Our main contributions are briefly summarized as follows.

• We provide characterizations of uniform equivalence of logic programs. In particular, we use the concept of strong-equivalence models (SE-models) [23, 24] and thus give characterizations which appeal to classical models and the Gelfond-Lifschitz reduct [8, 9]. Our characterizations of uniform equivalence will elucidate the differences between strong and uniform equivalence in the examples above such that they immediately become apparent.

• For the finitary case, we provide a simple and appealing characterization of a logic program with respect to uniform equivalence in terms of its *uniform equivalence models (UE models)*, which is a special class of SE-models. The associated notion of consequence can be fruitfully used to determine redundancies under uniform equivalence.

• We consider restricted subclasses, in particular positive programs, head-cycle free programs [1], and Horn programs, and consider the relationship between uniform and strong equivalence on them.

• We analyze the computational complexity of deciding uniform equivalence of two given programs P and Q. We show that the problem is Π_2^P-complete in the general propositional case, and thus harder than deciding strong equivalence of P and Q, which is in coNP [19, 24]. However, the complexity of testing uniform equivalence decreases on important fragments; in particular, it is coNP-complete for positive and head-cycle free programs, while it is polynomial for Horn programs. In the nonground case, the complexity increases by an exponential for function-free programs.

• Finally, we address extensions of the results to extended and to nested logic programs.

Our results complement the results on strong equivalence of logic programs, and pave the way for optimization of logic programs under stable negation by exploiting uniform equivalence.

The rest of this paper is organized as follows. The next section recalls some concepts and fixes notation. After that, in Section 3 we present our characterizations of uniform equivalence. We introduce the notions of UE-model and UE-consequence and relate the latter to other notions of consequence. Section 4 is then devoted to important classes of programs, in particular to positive and to head-cycle free logic programs. The subsequent Section 5 addresses complexity issues, and contains results on the complexity of equivalence checking and deciding UE-consequence. Section 6 considers possible extension of our results to nested logic programs and answer set semantics allowing for inconsistent answer sets. The final Section 7 concludes the paper and outlines issues for further research.

For increasing readability, some proofs or parts thereof have been moved to the Appendix.

2 Preliminaries

We deal with disjunctive logic programs, which allow the use of default negation not in rules. A rule r is a triple $\langle H(r), B^+(r), B^-(r)\rangle$, where $H(r) = \{A_1, \ldots, A_l\}$, $B^+(r) = \{A_{l+1}, \ldots, A_m\}$, $B^-(r) = \{A_{m+1}, \ldots, A_n\}$, where $0 \leq l \leq m \leq n$ and A_i, $1 \leq i \leq n$, are atoms from a first-order language. Throughout, we use the traditional representation of a rule as an expression of the form

$$A_1 \vee \ldots \vee A_l \leftarrow A_{l+1}, \ldots, A_m, not\, A_{m+1}, \ldots, not\, A_n.$$

We call $H(r)$ the *head* of r, and $B(r) = \{A_{l+1}, \ldots, A_m, not\, A_{m+1}, \ldots, not\, A_n\}$ the *body* of r. If $H(r) = \emptyset$, then r is a *constraint*. As usual, r is a *fact* if $B(r) = \emptyset$, which is also represented by $H(r)$ if it is nonempty, and by $\bot$ (falsity) otherwise. A rule r is *normal* (or non-disjunctive), if $l \leq 1$; *definite*, if $l = 1$; and *positive*, if $n = m$. A rule is *Horn* if it is normal and positive.

A *disjunctive logic program* (DLP) P is a (possibly infinite) set of rules. A program P is a *normal logic program* (NLP) (resp., definite, positive, or Horn), if all rules in P are normal (resp., definite, positive, or Horn). Furthermore, a program P is *head-cycle free* (HCF) [1], if its dependency graph (which is defined as usual) has no directed cycle that contains two atoms belonging to the head of the same rule. *In the rest of this paper, we focus on propositional programs over a set of atoms $\mathcal{A}$ – programs with variables reduce to their ground (propositional) versions as usual.*

We recall the stable model semantics for DLPs [9, 21], which generalizes the stable model semantics for NLPs [8]. An *interpretation* I, viewed as subset of $\mathcal{A}$, models the head of a rule r, denoted $I \models H(r)$, iff $A \in I$ for some $A \in H(r)$. It models $B(r)$, i.e., $I \models B(r)$ iff (i) each $A \in B^+(r)$ is true in I, i.e., $A \in I$, and (ii) each $A \in B^-(p)$ is false in I, i.e., $A \notin I$. Furthermore, I models rule r, iff $I \models H(r)$ whenever $I \models B(r)$, and I is a model of a program P, denoted $I \models P$, iff $I \models r$, for all $r \in P$.

The *reduct* of a rule r *relative to* a set of atoms I, denoted r^I, is the positive rule r' such that $H(r') = H(r)$ and $B^+(r') = B^+(r)$ if $I \cap B^-(r) = \emptyset$, and is void otherwise. The *Gelfond-Lifschitz reduct* P^I, of a program P is $P^I = \{r^I \mid r \in P \text{ and } I \cap B^-(r) = \emptyset\}$. An interpretation I is a *stable model* of a program P iff I is a minimal model (under inclusion $\subseteq$) of P^I. By $\mathcal{SM}(P)$ we denote the set of all stable models of P.

2.1 Equivalences

Several notions for equivalence of logic programs have been considered, cf. [11, 15, 22]. Under stable semantics, two DLPs P and Q are regarded as equivalent, denoted $P \equiv Q$, iff $\mathcal{SM}(P) = \mathcal{SM}(Q)$.

The more restrictive forms of strong equivalence [11] and uniform equivalence [22, 15] are as follows.

Definition 1 *Let P and Q be two DLPs, Then*

(i) *P and Q are strongly equivalent, denoted $P \equiv^s Q$, iff for any rule set R, the programs $P \cup R$ and $Q \cup R$ are equivalent, i.e., $P \cup R \equiv Q \cup R$.*

(ii) *P and Q are* uniformly equivalent*, denoted $P \equiv^u Q$, iff for any set of non-disjunctive facts F, the programs $P \cup F$ and $Q \cup F$ are equivalent, i.e., $P \cup F \equiv Q \cup F$.*

One of the main results of [11] is a semantical characterization of strong equivalence in terms of the non-classical logic HT. For characterizing strong equivalence in logic programming terms, Turner introduced the following notion of SE-models [23, 24]:

Definition 2 *Let P be a DLP, and let X, Y be sets of atoms such that $X \subseteq Y$. The pair (X, Y) is an SE-model of P, if $Y \models P$ and $X \models P^Y$. By $SE(P)$ we denote the set of all SE-models of P.*

Strong equivalence can be characterized as follows.

Proposition 2.1 ([23, 24]) *For every DLPs P and Q, $P \equiv^s Q$ iff $SE(P) = SE(Q)$.*

Example 3 *Reconsider $P = \{a \leftarrow not\, b;\ a \leftarrow b\}$ and $Q = \{a \leftarrow not\, c;\ a \leftarrow c\}$. Recall that $P \equiv^u Q$. However, $P \not\equiv^s Q$, as $(\emptyset, \{a, b\})$ is in $SE(P)$ but not in $SE(Q)$.*

Note that from the proofs of the results in [24], it appears that for strong equivalence, only the addition of rules which have at most one positive literal in the body is crucial. That is, by constraining the rules in the set R in the definition of strong equivalence to have at most one literal in the body does not lead to a different concept. This is encountered by restriction to facts (i.e., empty rule bodies), however.

3 Characterizations of Uniform Equivalence

After the preliminary definitions, we now turn to the issue of characterizing uniform equivalence between logic programs in model-theoretic terms. As restated above, strong equivalence can be captured by the notion of SE-model (equivalently, HT-model [11]) for a logic program. The weaker notion of uniform equivalence can be characterized in terms of SE-models as well, by imposing further conditions.

We start with a seminal lemma, which allows us to derive simple characterizations of uniform equivalence.

Lemma 3.1 *Two DLPs P and Q are uniformly equivalent, i.e. $P \equiv^u Q$, iff for every SE-model (X, Y), such that (X, Y) is an SE-model of exactly one of the programs P and Q, it holds that (i) $Y \models P \cup Q$, and (ii) there exists an SE-model (M, Y), $X \subset M \subset Y$, of the other program.*

Proof. For the only-if direction, suppose $P \equiv^u Q$. If Y neither models P, nor Q, then (X, Y) is not an SE-model of any of the programs P and Q. Without loss of generality, assume $Y \models P$ and $Y \not\models Q$. Then, since in this case $Y \models P^Y$ and no strict subset of Y models $P \cup Y$, $Y \in \mathcal{SM}(P \cup Y)$, while $Y \notin \mathcal{SM}(Q \cup Y)$. This contradicts our assumption $P \equiv^u Q$. Hence, item (i) must hold.

To show (ii), assume first that (X, Y) is an SE-model of P but not of Q. In view of (i), it is clear that $X \subset Y$ must hold. Suppose now that for every set M, $X \subset M \subset Y$, it holds that (M, Y) is not an SE-model of Q. Then, since no subset of X models $Q^Y \cup X$, (Y, Y) is the only SE-model of $Q \cup X$ of form $(\cdot, Y)$. Thus, $Y \in \mathcal{SM}(Q \cup X)$ in this case, while $Y \notin \mathcal{SM}(P \cup X)$ ($X \models P^Y$ implies $X \models (P \cup X)^Y$, so (X, Y) is an SE-model of $P \cup X$). However, this contradicts $P \equiv^u Q$. Thus, it follows that for some M such that $X \subset M \subset Y$, (X, Y) is an SE-model of Q. The argument in the case where (X, Y) is an SE-model of Q but not of P is analogous. This proves item (ii).

For the if direction, assume that (i) and (ii) hold for every SE-model (X, Y) which is an SE-model of exactly one of P and Q. Suppose that there exist sets of atoms A and X, such that w.l.o.g., $X \in \mathcal{SM}(P \cup A) \setminus \mathcal{SM}(Q \cup A)$. Since $X \in \mathcal{SM}(P \cup A)$, we have that $A \subseteq X$, and, moreover, $X \models P$. Consequently, (X, X) is an SE-model of P. Since $X \notin \mathcal{SM}(Q \cup A)$, either $X \not\models (Q \cup A)^X$, or there exists $X' \subset X$ such that $X' \models (Q \cup A)^X$.

Let us first assume $X \not\models (Q \cup A)^X$, then, since $(Q \cup A)^X = Q^X \cup A$ and $A \subseteq X$, it follows that $X \not\models Q^X$. This implies $X \not\models Q$ and hence, (X, X) is not an SE-model of Q. Thus, (X, X) is an SE-model of exactly one program, P, but (X, X) violates (i) since $X \not\models Q$; this is a contradiction.

It follows that $X \models (Q \cup A)^X$ must hold, and that there must exist $X' \subset X$ such that $X' \models (Q \cup A)^X = Q^X \cup A$. So we can conclude $X \models Q$ and that (X', X) is an SE-model of Q but not of P. To see the latter, note that $A \subseteq X'$ must hold. So if (X', X) were an SE-model of P, then it would also be an SE-model of $P \cup A$, contradicting the assumption that $X \in \mathcal{SM}(P \cup A)$. Again we get an SE-model, (X', X), of exactly one of the programs, Q in this case. Hence, according to (ii), there exists an SE-model (M, X) of P, $X' \subset M \subset X$. However, because of $A \subset X'$, it follows that (M, X) is also an SE-model of $P \cup A$, contradicting our assumption that $X \in \mathcal{SM}(P \cup A)$.

This proves that, given (i) and (ii) for every SE-model (X, Y) such that (X, Y) is an SE-model of exactly one of P and Q, no sets of atoms A and X exists such that X is a stable model of exactly one of $P \cup A$ and $Q \cup A$. That is, $P \equiv^u Q$ holds. □

From Lemma 3.1 we immediately obtain the following characterization of uniform equivalence between logic programs.

Theorem 3.1 *Two DLPs, P and Q are uniformly equivalent, $P \equiv^u Q$, iff*

(i) (X, X) *is an SE-model of* P *iff it is an SE-model of* Q*, and*

(ii) (X, Y)*, where* $X \subset Y$*, is an SE-model of* P *(respectively* Q*) iff there exists a set* M*, such that* $X \subseteq M \subset Y$*, and* (M, Y) *is an SE-model of* Q *(respectively* P*).*

Example 4 *Reconsider the programs* $P = \{a \vee b\}$ *and* $Q = \{a \leftarrow not\, b;\ b \leftarrow not\, a\}$*. By Theorem 3.1, we can easily verify that* P *and* Q *are uniformly equivalent: Their SE-models differ only in* $(\emptyset, \{a, b\})$*, which is an SE-model of* Q *but not of* P*. Thus, items* (i) *and* (ii) *clearly hold for all other SE-models. Moreover,* $(\{a\}, \{a, b\})$ *is an SE-model of* P*, and thus item* (ii) *also holds for* $(\emptyset, \{a, b\})$*.*

Note that P and Q are strongly equivalent after adding the constraint $\leftarrow a, b$, which enforces exclusive disjunction. Uniform equivalence does not require such an addition.

Example 5 *Let* P *and* Q *as in the previous example. Since* $SE(R \cup S) = SE(R) \cap SE(S)$ *for any programs* P *and* S*, the pair* $(\emptyset, \{a, b\})$ *is no longer an SE-model of* $Q \cup \{c : \leftarrow a, b\}$ *(because* $\{a, b\} \not\models c$*). Hence,* $P \cup \{c\} \equiv^s Q \cup \{c\}$*.*

For finite programs, we can derive from Theorem 3.1 the following characterization of uniform equivalence.

Theorem 3.2 *Two finite DLPs* P *and* Q *are uniformly equivalent, i.e.,* $P \equiv^u Q$*, iff the following conditions hold:*

(i) (X, X) *is an SE-model of* P *iff it is an SE-model of* Q *for every* X*, and*

(ii) for every SE-model $(X, Y) \in SE(P) \cup SE(Q)$ *such that* $X \subset Y$*, there exists an SE-model* $(M, Y) \in SE(P) \cap SE(Q)$ *(=*$SE(P \cup Q)$*) such that* $X \subseteq M \subset Y$*.*

Proof. Since (i) holds by virtue of Theorem 3.1, we only need to show (ii). Assume (X, Y), where $X \subset Y$, is in $SE(P) \cup SE(Q)$.

If $(X, Y) \in SE(P) \cap SE(Q)$, then the statement holds. Otherwise, by virtue of Theorem 3.1, there exists (M_1, Y), $X \subseteq M_1 \subset Y$, such that (M_1, Y) is in $SE(P) \cup SE(Q)$. By repeating this argument, we obtain a chain of SE-models $(X, Y) = (M_0, Y)$, (M_1, Y), ..., (M_i, Y), ... such that $(M_i, Y) \in SE(P) \cup SE(Q)$ and $M_i \subseteq M_{i+1}$, for all $i \geq 0$. Furthermore, we may choose M_1 such that M_1 coincides with Y on all atoms which do not occur in $P \cup Q$ (and hence all M_i, $i \geq 1$, do so). Since P and Q are finite, it follows that $M_i = M_{i+1}$ must hold for some $i \geq 0$ and hence $(M_i, Y) \in SE(P) \cap SE(Q)$ must hold. This proves the result. □

Note that the previous theorem remains valid even if only one of the programs P and Q is finite. In the light of this result, we can capture uniform equivalence of finite programs by the notion of UE-model defined as follows.

Definition 3 (UE-model) *Let* P *be a DLP. Then, any* $(X, Y) \in SE(P)$ *is a* uniform equivalence (UE) *model of* P*, if for every* $(X', Y) \in SE(P)$ *it holds that* $X \subset X'$ *implies* $X' = Y$*. By* $UE(P)$ *we denote the set of all UE-models of* P*.*

That is, the UE-models comprise all SE-models of a DLP which correspond to classical models of P (for $Y = X$), plus all its maximal 'non-classical' SE-models, i.e., $UE(P) = \{(X, X) \in SE(P)\} \cup \max_{\geq}\{(X, Y) \in SE(P) \mid X \subset Y\}$, where $(X', Y') \geq (X, Y) \Leftrightarrow Y' = Y \wedge X \subseteq X'$.

By means of UE-models, we then can characterize uniform equivalence of finite logic programs by the following simple condition.

Theorem 3.3 *Two finite DLPs P and Q are uniformly equivalent, i.e., $P \equiv^u Q$, if and only if $UE(P) = UE(Q)$.*

Proof. By Theorem 3.2 we have to show that Conditions (i) $(X, X) \models P \Leftrightarrow (X, X) \models Q$ and (ii) $(X, Y) \models P \wedge X \subset Y \Rightarrow \exists M, X \subseteq M \subset Y : (M, Y) \models P \cup Q$ hold iff $UE(P) = UE(Q)$.

For the if direction assume $UE(P) = UE(Q)$. Then (i) holds by definition of UE-models. Now let (X, Y) be an SE-model of P, such that $X \subset Y$. There are two possibilities: If (X, Y) is maximal, then $(X, Y) \in UE(Q)$ as well and thus (ii) holds $(M = X)$; otherwise, (X, Y) is not maximal, which means that there exists some $(X', Y) \in UE(P)$ such that $X \subset X' \subset Y$, and since $UE(P) = UE(Q)$ Condition (ii) holds again $(M = X')$.

For the only-if direction let $P \equiv^u Q$. Then by Condition (i) $UE(P)$ and $UE(Q)$ coincide on models (X, X). Assume w.l.o.g. that (X, Y), $X \subset Y$, is in $UE(P)$, but not in $UE(Q)$. By (ii) there exists (M, Y), $X \subseteq M \subset Y$, which is an SE-model of both P and Q. Since $X \subset M$ contradicts $(X, Y) \in UE(P)$, let $M = X$, i.e., (X, Y) is an SE-model of Q as well, but it is not in $UE(Q)$. Hence, there exists $(X', Y) \in UE(Q)$, $X \subset X' \subset Y$ and by (ii) there exists (M', Y), $X' \subseteq M \subset Y$, which is an SE-model of P. This again contradicts $(X, Y) \in UE(P)$. Hence, $UE(P) = UE(Q)$ must hold. □

This result shows that UE-models capture the notion of uniform equivalence for finite logic programs, in the same manner as SE-models capture strong equivalence. That is, the essence of a program P with respect to uniform equivalence is expressed by a semantic condition on P alone.

Example 6 *Each SE-model of the program $P = \{a \vee b\}$ satisfies the condition of an UE-model, and thus $UE(P) = SE(P)$. The program $Q = \{a \leftarrow not\, b;\ b \leftarrow not\, a\}$ has the additional SE-model $(\{\}, \{a, b\})$, and all of its SE-models except this one are UE-models of Q. Thus, $UE(P) = UE(Q) = \{(\{a\}, \{a\}), (\{b\}, \{b\}), (\{a\}, \{a, b\}), (\{b\}, \{a, b\}), (\{a, b\}, \{a, b\})\}$.*

Note that the strong equivalence between P and Q fails because $(\emptyset, \{a, b\})$ is not an SE-model of P. This SE-model is enforced by the intersection property ((X_1, Y) and (X_2, Y) in $SE(P)$ implies $(X_1 \cap X_2, Y) \in SE(P)$) which the Horn program Q^Y enjoys, which however is not satisfied by the disjunctive program P^Y (=P). The maximality condition of UE-models eliminates this intersection property.

Example 7 *Reconsider $P = \{a \leftarrow not\, b;\ a \leftarrow b\}$, which has classical models $\{a\} \cup Y$, $Y \subseteq \{b, c\}$. Its UE-models are of form $(\{a\} \cup X, \{a\} \cup Y)$ where $X \in \{Y, Y \setminus \{b\}, Y \setminus \{c\}\}$. Note that the atoms b and c have symmetric roles in $UE(P)$. Consequently, the program obtained by exchanging the roles of b and c, $Q = \{a \leftarrow not\, c;\ a \leftarrow c\}$ has the same UE models. Hence, P and Q are uniformly equivalent.*

Like Theorem 3.2, also Theorem 3.3 remains valid if only one of P and Q is finite. However, the following example shows that it fails if both P and Q are infinite.

Example 8 *Consider the programs P and Q over $\mathcal{A} = \{a\} \cup \{b_i \mid i \geq 1\}$, defined by*

$$P = \{a \leftarrow,\ \ b_i \leftarrow \mid i \geq 1\}, \quad \textit{and} \quad Q = \{a \leftarrow not\, a,\ b_i \leftarrow b_{i+1},\ b_i \leftarrow a \mid i \geq 1\}.$$

Both P and Q have the single classical model $M = \{a, b_i \mid i \geq 1\}$. Furthermore, P has no "incomplete" SE-model (X, Y) such that $X \subset Y$, while Q has the incomplete SE-models (X_i, M), where

$X_i = \{b_1, \ldots, b_i\}$ *for* $i \geq 0$. *Both* P *and* Q *have the same maximal incomplete SE-models (namely none), and hence they have the same UE-models.*

However, $P \not\equiv^u Q$, *since e.g.* P *has a stable model while* Q *has obviously not. Note that this is caught by our Theorem 3.1, item (ii): for* (X_0, M), *which is an SE-model of* Q *but not of* P, *we cannot find an SE-model* (X, M) *of* P *between* (X_0, M) *and* (M, M).

Based on UE-models, we define an associated notion of consequence under *uniform equivalence*. Recall that (X, Y) models a rule r iff $Y \models r$ and $X \models r^Y$.

Definition 4 (UE-consequence) *A rule,* r, *is an* UE-consequence *of a program* P, *denoted* $P \models_u r$, *if* $(X, Y) \models r$ *for all* $(X, Y) \in UE(P)$.

Clearly, $P \models_u r$ for all $r \in P$, and $\emptyset \models r$ iff r is a classical tautology. The next result shows that a program remains invariant under addition of UE-consequences.

Proposition 3.1 *For any finite program* P *and rule* r, *if* $P \models_u r$ *then* $P \cup \{r\} \equiv^u P$.

Proof. We show that $UE(P) = UE(P \cup \{r\})$, given $P \models_u r$.
"$\subseteq$": Let $(X, Y) \in UE(P)$. Then, by hypothesis $Y \models r$ and $X \models r^Y$. Hence, $Y \models P \cup \{r\}$ and $X \models (P \cup \{r\})^Y$. Suppose $(X, Y) \notin UE(P \cup \{r\})$. Then there exists a set X', $X \subset X' \subset Y$, such that $(X', Y) \models P \cup \{r\}$. But then $X' \models P^Y$, which contradicts $(X, Y) \in UE(P)$. It follows that $(X, Y) \in UE(P \cup \{r\})$.
"$\supseteq$": Let $(X, Y) \in UE(P \cup \{r\})$. Then $X \models P^Y$ and $Y \models P$. Suppose $(X, Y) \notin UE$. Then, some $(X', Y) \in UE(P)$ exists such that $X \subset X' \subset Y$. By hypothesis, $(X', Y) \models r$ (otherwise $P \not\models_u r$), hence $X' \models (P \cup \{r\})^Y$. But then $(X, Y) \in UE(P \cup \{r\})$, which is a contradiction. It follows $(X, Y) \in UE(P)$. □

From this proposition, we obtain an alternative characterization of uniform equivalence in terms of UE-consequence. As usual, we write $P \models_u R$ for any set of rules R if $P \models_u r$ for all $r \in R$.

Theorem 3.4 *Let* P *and* Q *be any finite DLPs. Then* $P \equiv^u Q$ *iff* $P \models_u Q$ *and* $Q \models_u P$.

Proof. For the if-direction, we apply Proposition 3.1 repeatedly and obtain $P \equiv^u P \cup Q \equiv^u Q$. For the only-if direction, we have $UE(P) = UE(Q)$ if $P \equiv^u Q$ by Theorem 3.3, and thus P and Q have the same UE-consequences. Since $(X, Y) \models P$ (resp. $(X, Y) \models Q$), for all $(X, Y) \in UE(P)$ (resp. $(X, Y) \in UE(Q)$, it follows $Q \models_u P$ and $P \models_u Q$. □

We note that with respect to uniform equivalence, every program P has a canonical normal form, P^*, given by its UE-consequences, i.e., $P^* = \{r \mid P \models_u r\}$.

Clearly, $P \subseteq P^*$ holds for every program P, and P^* has exponential size. Applying optimization methods which build on UE-consequence, P resp. P^* may be transformed into smaller uniform equivalent programs; we leave this for further study.

As for the relationship of UE-consequence to classical consequence and cautious consequence under stable semantics, we note the following hierarchy. Let $\models_c$ denote consequence from the stable models, i.e., $P \models_c r$ iff $M \models r$ for every $M \in \mathcal{SM}(P)$.

Proposition 3.2 *For any finite program* P *and rule* r, *(i)* $P \models_u r$ *implies* $P \cup A \models_c r$, *for each set of facts* A; *(ii)* $P \cup A \models_c r$, *for each set of facts* A, *implies* $P \models_c r$; *and (iii)*$P \models_c r$ *implies* $P \models r$.

Proof. Since each stable model is a classical model, it remains to show (i).

Suppose $P \models_u r$. Then, $P \equiv^u P \cup \{r\}$ by Proposition 3.1, i.e., $\mathcal{SM}(P \cup A) = \mathcal{SM}(P \cup \{r\} \cup A)$, for each set of facts A. Since $X \models r$ for each $X \in \mathcal{SM}(P \cup \{r\} \cup A)$, it follows that $P \cup A \models_c r$, for each set of facts A. □

This hierarchy is strict, i.e., none of the implications holds in the converse direction. (For (i), note that $\{a \leftarrow not\, a\} \models_c a$ but $\{a \leftarrow not\, a\} \not\models_u a$, since the UE-model $(\emptyset, \{a\})$ violates a.)

We next present a semantic characterization in terms of UE-models, under which UE-and classical consequence and thus all four notions of consequence coincide.

Lemma 3.2 *Let P be a finite DLP. Suppose that $(X, Y) \in UE(P)$ implies $X \models P$ (i.e., X is a model of P). Then, $P \models r$ implies $P \models_u r$, for every rule r.*

Proof. Consider $(X, Y) \in UE(P)$. By hypothesis, $X \models P$ and $P \models r$, thus $X \models r$, which implies $X \models r^X$. Furthermore, $Y \models r$ since $Y \models P$. We need to show that $X \models r^Y$. Note that either r^Y is void, or, since $X \subseteq Y$ $r^Y = r^X$. In both cases $X \models r^Y$ follows, which proves $(X, Y) \models r$. □

Lemma 3.3 *Let P be a finite DLP. Then, $P \models_u r$ implies $P \models r$, for every rule r.*

Proof. Since $(X, Y) \models r$ implies $Y \models r$, for every rule r, and since $(Y, Y) \in UE(P)$ for all Y such that $Y \models P$, $P \models r$ must hold in case of $P \models_u r$, for every rule r. □

Theorem 3.5 *Let P be any finite DLP. Then the following conditions are equivalent:*

(i) $P \models_u r$ *iff* $P \models r$, *for every rule* r.

(ii) For every $(X, Y) \in UE(P)$, *it holds that* $X \models P$.

Proof. $(ii) \Rightarrow (i)$ Follows immediately from Lemmas 3.2 and 3.3.
$(i) \Rightarrow (ii)$ Suppose $P \models_u r$ iff $P \models r$, for every rule r, but there exists some UE-model (X, Y) of P such that $X \not\models P$. Hence $X \not\models r$ for some rule $r \in P$. Let r' be the rule which results from r by shifting the negative literals to the head, i.e., $H(r') = H(r) \cup B^-(r)$, $B^+(r') = B^+(r)$, and $B^-(r') = \emptyset$. Then, $X \not\models r'$. On the other hand, $r \in P$ implies $(X, Y) \models r$. Hence, $Y \models r$ and thus $Y \models r'$. Moreover, $B^-(r') = \emptyset$ implies that $r' \in P^Y$, and hence $X \models r'$. This is a contradiction. It follows that $X \models P$ for each UE-model (X, Y) of P. □

An immediate corollary to this result is that for finite positive programs, the notion of UE-consequence collapses with classical consequence, and hence uniform equivalence of finite positive programs amounts to classical equivalence. We shall obtain these results as corollaries of more general results in the next section, though.

4 Restricted Classes of Programs

After discussing uniform equivalence of general propositional programs, let us now consider two prominent subclasses of programs, namely positive and head-cycle free programs.

4.1 Positive programs

While for programs with negation, strong equivalence and uniform equivalence are different, the notions coincide for positive programs, as shown next.

Proposition 4.1 *Let P and Q be positive DLPs. Then $P \equiv^u Q$ iff $P \equiv^s Q$.*

Proof. The if-direction is immediate as $P \equiv^s Q$ implies $P \equiv^u Q$.

For the only-if-direction, we show that if P and Q are not strongly equivalent then P and Q are not uniformly equivalent. To start with, observe that $P^X = P$ holds for any positive program P and any set of literals X.

W.l.o.g., let (X, Y) be an SE-model of P but not of Q. By definition of SE-model we have $X \models P^Y$, i.e. $X \models P$. On the other hand, since (X, Y) is not SE-model of Q, either (i) $X \not\models Q^Y$, i.e., $X \not\models Q$, or (ii) $Y \not\models Q$.

(i) Consider the programs $P_X = P \cup X$ and $Q_X = Q \cup X$. Clearly, $X \models P_X$ and for each $X' \subset X$, $X' \not\models P_X = P_X^X$. Hence, X is an answer set of P_X. On the other hand, $X \not\models Q$ and thus $X \not\models Q_X$. Hence, X cannot be an answer set of Q_X.

(ii) Consider the programs $P_Y = P \cup Y$ and $Q_Y = Q \cup Y$. Clearly, $Y \models P_Y$ and for each $Y' \subset Y$, $Y' \not\models P_Y = P_Y^Y$. Hence, Y is an answer set of P_Y. On the other hand, $Y \not\models Q$ and thus $Y \not\models Q_Y$. Hence, Y cannot be an answer set of Q_Y.

In any case we must conclude that P and Q are not uniformly equivalent. □

As known and easy to see from the main results of [11, 23, 24], on the class of positive programs classical equivalence and strong equivalence coincide. By combining this and the previous result, we obtain

Theorem 4.1 *Let P and Q be positive DLPs. Then $P \equiv^u Q$ if and only if $P \models Q$ and $Q \models P$, i.e., P and Q have the same set of classical models.*

Note that Sagiv [22] showed that uniform equivalence of datalog programs Π and Π' is equivalent to equivalence of Π' and Π over Herbrand models; this implies the above result for definite Horn programs. Maher [15] showed a generalization of Sagiv's result for definite Horn logic programs with function symbols. Furthermore, Maher also pointed out that for Datalog programs, this result has been independently established by Cosmadakis and Kanellakis [5], and that Buntine [3] has, like Sagiv, provided an algorithm for deciding the stronger property of uniform containment, which can be exploited to easily decide uniform equivalence.

Example 9 *Consider the positive programs $P = \{a \vee b \leftarrow a;\ b \leftarrow a\}$ and $Q = \{b \leftarrow a\}$. Clearly, $P \models Q$ since $Q \subset P$, but also $Q \models P$ holds (note that $b \leftarrow a$ is a subclause of $a \vee b \leftarrow a$). Hence, P and Q are uniformly equivalent, and even strongly equivalent (which is also easily verified).*

Example 10 *Consider the positive programs $P = \{a \vee b;\ c \leftarrow a;\ c \leftarrow b\}$ and $Q = \{a \vee b;\ c\}$. Their classical models are $\{a, c\}$, $\{b, c\}$, and $\{a, b, c\}$. Hence, P and Q are uniformly equivalent, and even strongly equivalent (due to Prop. 4.1).*

4.2 Head-cycle free programs

The class of head-cycle free programs generalizes the class of NLPs by permitting a restricted form of disjunction. Still, it is capable of expressing nondeterminism such as a guess for the value of an atom a, which does not occur in the head of any other rule.

As shown by Ben-Eliyahu and Dechter, each head-cycle free program can be rewritten to an NLP, obtained by shifting atoms from the head to the body, which has the same stable models. More formally, let us define the following notation:

Definition 5 *For any rule r, let $r^{\rightarrow} = \{r' \mid H(r') = \{a\}, a \in H(r), B^+(r') = B^+(r), B^-(r') = B^-(r) \cup H(r) \setminus \{a\}\}$ if $H(r) \neq \emptyset$ and $r^{\rightarrow} = \{r\}$ otherwise. For any DLP P, let $P^{\rightarrow} = \bigcup_{r \in P} r^{\rightarrow}$.*

It is well-known that for any head-cycle free program P, it holds that $P \equiv P^{\rightarrow}$ (cf. [1]). This result can be strengthened to uniform equivalence.

Theorem 4.2 *For any head-cycle free program P, it holds that $P \equiv^u P^{\rightarrow}$.*

Proof. For any set of facts A, it holds that $(P \cup A)^{\rightarrow} = P^{\rightarrow} \cup A$ and that this program is head-cycle free. Thus, $P \cup A \equiv (P \cup A)^{\rightarrow} \equiv P^{\rightarrow} \cup A$. Hence, $P \equiv^u P^{\rightarrow}$. □

We emphasize that a similar result for strong equivalence fails, as shown by the canonical counterexample in Example 1. Moreover, the program $P = \{a \vee b \leftarrow .\}$ is not strongly equivalent to any NLP. Thus, we can not conclude without further consideration that a simple disjunctive "guessing clause" like the one in P (such that a and b do not occur in other rule heads) can be replaced in a more complex program by the unstratified clauses $a \leftarrow not\, b$ and $b \leftarrow not\, a$; addition of a further constraint $\leftarrow a, b$ is required. However, we can conclude this under uniform equivalence taking standard program splitting results into account [12, 7].

We close this section with the following result, which provides a characterization of arbitrary programs which are strongly equivalent to their shift variant.

Theorem 4.3 *Let P be any DLP. Then, $P \equiv^s P^{\rightarrow}$ if and only if for every disjunctive rule $r \in P$ it holds that $P^{\rightarrow}$ has no SE-model (X, Y) such that (i) $|H(r) \cap Y| \geq 2$ and (ii) $X \cap H(r) = \emptyset$ and $X \models B^+(r)$, i.e., X violates the reduced rule r^Y.*

Proof. For the only-if direction, assume $P \equiv^s P^{\rightarrow}$. Suppose (X, Y) is an SE-model of $P^{\rightarrow}$ such that (i) and (ii) holds. Then $X \not\models P^Y$, since r^Y is in P^Y, $X \models B^+(r) = B(r^Y)$, but $X \cap H(r) = \emptyset$. Hence, (X, Y) is not an SE-model of P, which contradicts $P \equiv^s P^{\rightarrow}$.

For the if-direction, assume P is such that for every disjunctive rule $r \in P$, it holds that there is no SE-model (X, Y), of $P^{\rightarrow}$ such that (i) and (ii) holds.

Towards a contradiction, assume that $P \not\equiv^s P^{\rightarrow}$. It is easily verified that every SE-model of P is also an SE-model of $P^{\rightarrow}$ and that every model of $P^{\rightarrow}$ is also a model of P. Hence, $P \not\equiv^s P^{\rightarrow}$ implies that there exists some SE-model (X, Y) of $P^{\rightarrow}$ such that $Y \models P$ which is not an SE-model of P. Thus, $X \models P^{\rightarrow Y}$, but $X \not\models P^Y$. Consequently, there must be a rule r^Y in P^Y such that $X \not\models r^Y$. Since each non-disjunctive rule from P belongs to $P^{\rightarrow}$, rule r must be disjunctive. The hypothesis on the properties of SE-models of P with respect r and $X \not\models r^Y$ implies that (i) is false, i.e., $|H(r) \cap Y| \leq 1$, since (ii) is true. Since $Y \models r$, it follows $|H(r) \cap Y| = 1$. Hence, there exists some rule $r' \in r^{\rightarrow}$ such that its reduct r'^Y belongs to $P^{\rightarrow Y}$. Since $Y \models r$ and $X \subseteq Y$, it follows that $X \models B(r')$. On the other hand, $X \not\models r^Y$ implies

that $H(r) \cap X = \emptyset$, hence $H(r') \cap X = \emptyset$. This means that $X \not\models r'^Y$, and thus $X \not\models P^{\rightarrow Y}$. This is a contradiction. It follows that $P \equiv^s P^{\rightarrow}$ must hold. □

Example 11 *Reconsider $P = \{a \vee b \leftarrow\}$. Then $P^{\rightarrow} = \{a \leftarrow not\, b, b \leftarrow not\, a\}$ has the SE-model $(\emptyset, \{a, b\})$, which satisfies the conditions (i) and (ii) for $r : a \vee b \leftarrow$. Note that also the extended program $P' = \{a \vee b \leftarrow, a \leftarrow b, b \leftarrow a\}$ is not strongly equivalent to its shifted program $P'^{\rightarrow}$. Indeed, $(\emptyset, \{a, b\})$ is also an SE-model of $P'^{\rightarrow}$. Furthermore, P is also not uniformly equivalent to $P'^{\rightarrow}$, since $(\emptyset, \{a, b\})$ is moreover a UE-model of $P'^{\rightarrow}$, but P has the single SE-model (and thus UE-model) $(\{a, b\}, \{a, b\})$.*

5 Complexity

In this section, we address the computational complexity of uniform equivalence. While our main interest is with the problem of deciding uniform equivalence between two given programs, we also consider the related problems of UE-model checking and UE-consequence.

For UE-model checking, we have the following result. Let $\|\alpha\|$ denote the size of an object α.

Theorem 5.1 *Given a pair of sets (X, Y) and a program P, deciding whether $(X, Y) \in UE(P)$ is (i) coNP-complete in general, and (ii) feasible in polynomial time with respect to $\|P\| + \|X\| + \|Y\|$, if P is head-cycle free. Hardness in case (i) holds even for positive programs.*

Proof. Testing $Y \models P$ and $X \models P^Y$ is possible in polynomial time. If $X \subset Y$ it remains to check that no X', $X' \models P^Y$, exists such that $X \subset X' \subset Y$. This can be done in polynomial time with a coNP-oracle. This shows coNP-membership for (i) .

If P is head-cycle free, then also P^Y is. Let P' result from P^Y by removing

- any rule r such that $B(r) \not\subseteq Y$
- all atoms not in Y from the heads of the remaining rules,

i.e., $P' = \{H(r) \cap Y \leftarrow B(r) \mid r \in P^Y, B(r) \not\subseteq Y\}$. Note that $X \models P'$, $Y \models P'$, and that P' is positive and head-cycle free. Moreover, Y contains each minimal model of P'. We use the following lemma, which is due to Ben-Eliyahu and Palopoli [2]:

Lemma 5.1 *Given a positive, head-cycle free program Q without constraints, a minimal model of Q can be computed in polynomial time.*

Now, there does not exist X', $X' \models P^Y$, $X \subset X' \subset Y$ iff for all $a \in Y \setminus X$, Y is the unique minimal model of the positive, head-cycle free program $P' \cup \{a \leftarrow\}$. Since this program has no constraints, by Lemma 5.1, this can be decided in polynomial time. Hence, deciding $(X, Y) \in UE(P)$ is polynomial for head-cycle free programs P.

It remains to show coNP-hardness of UE-model checking for positive programs. We show this by a reduction to tautology checking. Let $F = \bigvee_{k=1}^{m} D_k$ be a propositional formula in 3DNF containing literals over the (propositional) variables $X = \{x_1, \ldots, x_n\}$, and consider the following program P:

$$P = \{\ x_i \vee x'_i \leftarrow x_j. \quad x_i \vee x'_i \leftarrow x'_j. \qquad 1 \le i \ne j \le n;$$
$$x_i \leftarrow D^*_k. \qquad x'_i \leftarrow D^*_k. \qquad 1 \le k \le m, 1 \le i \le n\ \},$$

where D_k^* results from D_k by replacing literals $\neg x_i$ by x_i'. Note that P is positive.

Note that $(\emptyset, X \cup X')$, $X' = \{x' \mid x \in X\}$, is an SE-model of P and $(\emptyset, X \cup X') \in UE(P)$ iff F is a tautology. This proves coNP-hardness. □

Corollary 5.1 *UE-model checking for Horn programs is polynomial.*

We now consider the problem of our main interest, namely deciding uniform equivalence. By the previous theorem, the following upper bound on the complexity of this problem is obtained.

Lemma 5.2 *Given two DLPs P and Q, deciding whether $P \equiv^u Q$ is in the class Π_2^P.*

Recall that $\Pi_2^P = \mathrm{coNP}^{\mathrm{NP}}$ is the class of problems such that the complementary problem is nondeterministically decidable in polynomial time with the help of an NP oracle (i.e., in $\Sigma_2^P = \mathrm{NP}^{\mathrm{NP}}$).

Proof. To show that two DLPs P and Q are not uniformly equivalent, we can by Theorem 3.3 guess an SE-model (X, Y) such that (X, Y) is an UE-model of exactly on of the programs P and Q. By Theorem 5.1, the guess for (X, Y) can be verified in polynomial time with the help of an NP oracle. This proves Π_2^P-membership of $P \equiv^u Q$. □

This upper bound has a complementary lower bound proved in the following result.

Theorem 5.2 *Given two DLPs P and Q, deciding whether $P \equiv^u Q$ is Π_2^P-complete.*

Proof. Membership in Π_2^P has already been established in Lemma 5.2. To show Π_2^P-hardness, we provide a polynomial reduction of evaluating a quantified Boolean formula (QBF) from a fragment which is known Π_2^P -complete to deciding uniform equivalence of two DLPs P and Q.

Consider a $QBF_{2,\exists}$ F of form $F = \exists X \forall Y \bigvee_{i=1}^{i=l} D_i$, where each D_i is a conjunct of at most three literals over the boolean variables in $X \cup Y$, $X = \{x_i \mid 1 \leq i \leq n\}$ and $Y = \{y_i \mid 1 \leq i \leq m\}$. Deciding whether a given such F is true is well-known to be Σ_2^P-complete; thus deciding whether F is false is Π_2^P-complete.

W.l.o.g., we assume that each D_i contains some literal over Y. Now let P and Q be the following programs:

$$
\begin{array}{lllll}
P = \{\ x_i \vee x_i' \leftarrow . & 1 \leq i \leq n; & & & \\
\quad y_i \vee y_i' \leftarrow y_j. & 1 \leq i \neq j \leq m; & & & \\
\quad y_i \vee y_i' \leftarrow y_j'. & 1 \leq i \neq j \leq m; & & & \\
\quad w_0 \leftarrow x_i, x_i'. & 1 \leq i \leq n; & w_1 \leftarrow y_i, y_i'. & 1 \leq i \leq m; & \\
\quad x_i \leftarrow w_0. & 1 \leq i \leq n; & w_1 \leftarrow D_i^*. & 1 \leq i \leq l; & \\
\quad x_i' \leftarrow w_0. & 1 \leq i \leq n; & y_i \leftarrow w_1. & 1 \leq i \leq m; & \\
\quad y_i \leftarrow w_0. & 1 \leq i \leq m; & y_i' \leftarrow w_1. & 1 \leq i \leq m; & \\
\quad y_i' \leftarrow w_0. & 1 \leq i \leq m; & w_1 \leftarrow not\, w_1. & & \\
\quad w_1 \leftarrow w_0. & & & & \},
\end{array}
$$

and $\quad Q = P \cup \{y_1 \vee y_1' \leftarrow .\}$,

where D_i^* results from D_i by replacing literals $\neg x_i$ and $\neg y_i$ by x_i' and y_i', respectively.

Informally, the disjunctive clauses with x_i and x_i' in the head resp. y_i and y_i' serve for selecting a truth assignment to the variable x_i (resp., y_i). The atom w_0 serves for handling a spoiled assignment to some x_i, which occurs if both x_i and x_i' are true, and enforces the maximal interpretation as the unique model of

P, by the rules with w_0 in the body. Similarly, w_1 recognizes a spoiled assignment to some variable y_i or that some disjunct D_j of the QBF is true, and enforces for all atoms y_i, y_i' and w_1 the unique value true. However, for P the selection of a truth assignment is conditional to the truth of any atom y_i or y_i', while for Q it is mandatory by the additional rule $y_1 \vee y_1' \leftarrow$. This difference leads to a suite of candidate SE-models (A_χ, M_χ) of P which do not satisfy Q, where χ corresponds to a truth assignment to X and all atoms not in $X \cup X'$ are false, such that (A_χ, M_χ) violates uniform equivalence of P and Q via (ii) just if there is no way to make all D_j false by some assignment μ to Y. These candidate models for spoiling uniform equivalence are eliminated iff formula F evaluates to false; a formalization and proof of this intuition is provided in the Appendix. Since P and Q are obviously constructible in polynomial time, our result follows. □

The previous result shows that deciding uniform equivalence of DLPs P and Q is more complex than deciding strong equivalence, which is in coNP [19, 24]. Thus, the more liberal notion of uniform equivalence comes at higher computational cost in general. However, for important classes of programs, it has the same complexity.

Theorem 5.3 *Let P and Q be DLPs without simultaneous negation and head-cycles (i.e., each program is either positive or head-cycle free). Then, deciding whether $P \equiv^u Q$ is* coNP*- complete, where* coNP*-hardness holds if P is either positive or a NLP, and Q is Horn.*

Proof. For the membership part, by Theorem 3.4, $P \equiv^u Q$ iff $P \models_u Q$ and $Q \models_u P$. Furthermore, deciding $P' \models_u r$ is in coNP for programs P' which are positive or head-cycle free (see Theorem 5.4 below). Since the class coNP is closed under conjunction, it follows that deciding $P \equiv^u Q$ is in coNP.

Next we show coNP-hardness for a positive DLP P and a Horn program Q by a reduction from UNSAT. Given a propositional formula in CNF $F = \bigwedge_{i=1}^m C_i$ over the (propositional) variables $X = \{x_1, \ldots, x_n\}$, let $P_1 = \{C_i^\square \vee a \leftarrow . \mid 1 \leq i \leq m\}$, where $C_i^\square$ results from C_i by replacing literals $\neg x_i$ by x_i', let $P_2 = \{\leftarrow x_j, x_j'. \mid 1 \leq j \leq n\}$, and let $P = P_1 \cup P_2$, and $Q = \{a \leftarrow .\} \cup P_2$. By Theorem 4.1, $P \equiv^u Q$ iff P and Q have the same classical models. Clearly, the latter holds iff $P \models a \leftarrow$ holds, which is equivalent to F being unsatisfiable. This proves the result for positive P. We notice that P is head-cycle free. Hence, by Theorem 4.2 the result for NLPs P follows if we use $P = P_1^\rightarrow \cup P_2$ in the above argument. □

Note that Sagiv showed [22] that deciding $P \equiv^u Q$ for given definite Horn programs P and Q is polynomial. This clearly generalizes to arbitrary Horn programs. We further remark that for NLPs deciding strong equivalence is also coNP-hard.

Finally, we complement the results on uniform equivalence and UE-model checking with briefly addressing the complexity of UE-consequence.

Theorem 5.4 *Given a DLP P and a rule r, deciding $P \models_u r$ is (i) Π_2^P-complete in general, (ii)* coNP*-complete if P is either positive or head-cycle free, and (iii) polynomial if P is Horn.*

Proof. The complementary problem, $P \not\models_u r$, is in Σ_2^P for general P and in NP for head-cycle free P, since a guess for a UE-model (X, Y) of P which violates r can, by Theorem 5.1 be verified with a call to a NP-oracle resp. in polynomial time. In case of a positive P, by Theorem 3.5, $P \models_u r$ iff $P \models r$, which is in coNP for general P and polynomial for Horn P.

The Π_2^P-hardness part for (i) is easily obtained from the reduction proving the Π_2^P-hardness part of Theorem 5.2. For the program P constructed there, it holds by Theorem 3.4 that $P \models_u y_1 \vee y_1'$ if and only if $P \equiv^u Q$ holds, which is Π_2^P-hard to decide.

The coNP-hardness in case of (ii) follows easily from the reduction which proves the coNP-hardness part of Theorem 5.3: the positive program P constructed there satisfies, by Theorem 3.5, $P \models_u a \leftarrow$ if and only if $P \equiv^u a \leftarrow$ holds, which is equivalent to unsatisfiability of the CNF F there. Again, for showing the coNP-hardness for non-positive programs, we can use the NLP $P_1^{\rightarrow} \cup P_2$ (=$P^{\rightarrow}$) in place of P. □

We conclude this section with some remarks on the complexity of programs with variables. For such programs, in case of a given finite Herbrand universe the complexity of equivalence checking increases by an exponential. Intuitively, this is explained by the exponential size of the ground instance of a program over the universe. Note that Lin reported [13], without a full proof, that checking strong equivalence for programs in this setting is in coNP, and thus has the same complexity as in the propositional case; however, this is not correct. Unsurprisingly, uniform equivalence of logic programs over an arbitrary Herbrand universe is undecidable according to Maher [15].

6 Extensions

Our results easily carry over to extended logic programs, i.e., programs where classical (also called strong) negation is allowed as well. If the inconsistent answer set is disregarded, i.e., an inconsistent program has no models, then, as usual, the extension can be semantically captured by representing strongly negated atoms $\neg A$ by a positive atom A' and adding constraints $\leftarrow A, A'$, for every atom A, to any program.

Furthermore, since the proofs of our main results are generic in the use of reducts, they can be easily generalized to nested logic programs considered in [11, 23, 24, 19], i.e., we get the same characterizations and the same complexity (Π_2^P).

However, if in the extended setting the inconsistent answer set is taken into account, then the given definitions have to be slightly modified such that the characterizations of uniform equivalence capture the extended case properly. The same holds true for the characterization of strong equivalence by SE-models as illustrated by the following example. Note that the redefinition of $\equiv^u$ and $\equiv^s$ is straightforward.

Let $Lit_{\mathcal{A}} = \{A, \neg A \mid A \in \mathcal{A}\}$ denote the (inconsistent) set of all literals using strong negation over $\mathcal{A}$. Note that an extended DLP P has an inconsistent answer set iff $Lit_{\mathcal{A}}$ is an answer set of it; moreover, it is in the latter case the only answer set of P. Call any DLP P *contradiction-free*, if $Lit_{\mathcal{A}}$ is not an answer set of it, and *contradictory* otherwise.

Example 12 *Consider the extended logic programs* $P = \{a \vee b \leftarrow ;\ \neg a \leftarrow a;\ \neg b \leftarrow b\}$ *and* $Q = \{a \leftarrow not\, b;\ b \leftarrow not\, a;\ \neg a \leftarrow a;\ \neg b \leftarrow b\}$. *They both have no SE-model; hence, by the criterion of Prop. 2.1,* $P \equiv^s Q$ *would hold, which implies* $P \equiv^u Q$ *and* $P \equiv Q$. *However,* P *has the inconsistent answer set* $Lit_{\mathcal{A}}$, *while* Q *has no answer set. Thus formally,* P *and* Q *are not even equivalent if* $Lit_{\mathcal{A}}$ *is admitted as answer set.*

Since [23, 11, 24] made no distinction between no answer set and inconsistent answer set, we start adapting the definition of SE-models.

Definition 6 *A pair* (X, Y), $X \subseteq Y \subseteq Lit_{\mathcal{A}}$, *is an SEE-model of an extended* DLP P, *if each of* X *and* Y *is either consistent or equals* $Lit_{\mathcal{A}}$ *and* $Y \models P \wedge X \models P^Y$.

The following lemma is immediate from the definition.

Lemma 6.1 *Programs* P *and* Q *with the same SEE-models are equivalent, and the SEE-models of* $P \cup Q$ *are exactly the SEE-models common to both* P *and* R.

From previous characterizations we get more general characterizations in terms of SEE-models for extended programs.

Theorem 6.1 *Two extended DLPs P and Q are*

- *strongly equivalent iff they have the same SEE-models, and*
- *uniformly equivalent iff*

  (i) (X, X) *is an SEE-model of P iff it is an SEE-model of Q, and*

  (ii) (X, Y), $X \subset Y$, *is an SEE-model of P (resp. Q) iff there exists a set M, such that $X \subseteq M \subset Y$, and (M, Y) is an SEE-model of Q (resp. P).*

Proof. See Appendix. □

For positive programs, uniform and strong equivalence coincide also in the extended case.

Theorem 6.2 *Let P and Q be positive, extended DLPs. Then $P \equiv^s Q$ iff $P \equiv^u Q$.*

Proof. The only-if direction is again straightforward from the definitions.

For the if direction, note that in view of Theorem 4.1 we only have that show that if $P \equiv^u Q$, there does not exist a set of rules R such that one of the programs $P \cup R$ and $Q \cup R$ has the answer set $Lit_{\mathcal{A}}$, while the other has no answer set.

Suppose the contrary, i.e., that there exists a set of rules R, such that w.l.o.g. $P \cup R$ has the answer set $Lit_{\mathcal{A}}$, while $Q \cup R$ has no answer set. Then since $P \cup R$ is inconsistent, there must not exist a constraint in P or in R with $B^-(r) = \emptyset$, (otherwise $Lit_{\mathcal{A}}$ would model it and, thus, not be an answer set of $P \cup R$). Also, there must not exist a constraint c in Q, otherwise, $L = B(c) = B^+(c)$ is a set such that $Q \cup L$ has no answer set while $P \cup L$ must have an answer set (since it is positive and constraint free), which contradicts $P \equiv^u Q$. Hence, because $Lit_{\mathcal{A}}$ is no an answer set for $q \cup R$ there must exist a consistent set X such that $X \models (Q \cup R)^{Lit_{\mathcal{A}}}$. Thus $X \models R^{Lit_{\mathcal{A}}}$ and (recall that P and Q are positive) $X \models Q\ (= Q^{Lit_{\mathcal{A}}})$, but $X \not\models P\ (= P^{Lit_{\mathcal{A}}})$ (otherwise X would also model $(P \cup R)^{Lit_{\mathcal{A}}}$, a contradiction to the facs that $Lit_{\mathcal{A}}$ is an answer set of $P \cup R$). What follows is that X is an answer set of $Q \cup X$ but not of $P \cup X$; this is a contradiction to $P \equiv^u Q$.

So we must conclude that if $P \equiv^u Q$, no set of rules R can exist, such that one of the programs $P \cup R$ and $Q \cup R$ has the answer set $Lit_{\mathcal{A}}$ while the other has no answer set. This proves our result. □

As a consequence of previous complexity results, checking $P \equiv^u Q$ (resp. $P \equiv^s Q$) for extended logic programs, P and Q, is Π_2^P-hard (resp. coNP-hard).

Theorem 6.3 *Checking whether extended DLPs P and Q are (i) uniformly equivalent is Π_2^P-complete; (ii) strongly equivalent is* coNP*-complete.*

However, not all properties do carry over. As Example 12 reveals, in general a head-cycle free extended DLP P is no longer equivalent, and hence not uniformly equivalent, to its shift variant $P^{\leftarrow}$. The next proposition gives a characterization of the head-cycle and contradiction free programs for which this equivalence holds.

For any program P, let us denote $P_{\leq 1} = \{r \in P \mid |H(r)| \leq 1\}$ and $D_H(P) = \{L \mid L \in H(r), |H(r)| > 1, r \in P\}$.

Proposition 6.1 *Let P be a head-cycle free and contradiction-free DLP. Then the following are equivalent:*

(i) $P \equiv^u P^{\rightarrow}$

(ii) For each $A \subseteq Lit_{\mathcal{A}}$, the program $P \cup A$ is contradiction-free if $P_{\leq 1} \cup A$ is contradiction-free.

(iii) For each $A \subseteq Lit_{\mathcal{A}}$ such that $D_H(P) \not\subseteq A$, the program $P \cup A$ is contradiction-free if $P_{\leq 1} \cup A$ is contradiction-free.

Proof. Note that (ii) trivially implies (iii). Thus it remains to show that (i) implies (ii) and that (iii) implies (i).
$(i) \Rightarrow (ii)$ Suppose $P \equiv^u P^{\rightarrow}$. Towards a contradiction, assume that there exists a set of literals $A \subseteq Lit_{\mathcal{A}}$, such that $P \cup A$ is not contradiction-free, i.e., has the inconsistent answer set $Lit_{\mathcal{A}}$, while $P_{\leq 1} \cup A$ is contradiction-free. Then $(P_{\leq 1} \cup A)^{Lit_{\mathcal{A}}}$ is contradiction-free as well. Since $(P^{\rightarrow} \cup A)^{Lit_{\mathcal{A}}}$ coincides with $(P_{\leq 1} \cup A)^{Lit_{\mathcal{A}}}$, we obtain that $Lit_{\mathcal{A}}$ is not an answer set of $(P^{\rightarrow} \cup A)^{Lit_{\mathcal{A}}}$. On the other hand, $Lit_{\mathcal{A}}$ is an answer set of $(P \cup A)^{Lit_{\mathcal{A}}}$, since it is an answer set of $P \cup A$. It follows that $P \cup A \not\equiv P^{\rightarrow} \cup A$, and thus $P \not\equiv^u P^{\rightarrow}$.
$(iii) \Rightarrow (i)$ Towards a contradiction, suppose it holds that $D_H(P) \not\subseteq A$ implies that $P \cup A$ is contradiction-free if $P_{\leq 1} \cup A$ is contradiction-free, but $P \not\equiv^u P^{\rightarrow}$, i.e., $P \cup A \not\equiv P^{\rightarrow} \cup A$ for some A. There are two cases: if $D_H(P) \subseteq A$, then as easily seen $P \cup A \equiv P_{\leq 1} \cup A \equiv P^{\rightarrow} \cup A$: this contradicts $P \cup A \not\equiv P^{\rightarrow} \cup A$. Hence, $D_H(P) \not\subseteq A$ must hold. Now suppose $P \cup A$ is contradiction-free. Since P is head-cycle free, also $P \cup A$ is head-cycle free, which implies $P \cup A \equiv (P \cup A)^{\rightarrow}$; this follows from a straightforward generalization of Ben-Eliyahu and Dechter's shifting result $Q \equiv Q^{\rightarrow}$ [1] to head-cycle -free programs Q which are contradiction-free, if $Lit_{\mathcal{A}}$ is an admissible answer set. Since $(P \cup A)^{\rightarrow} = P^{\rightarrow} \cup A$, this again contradicts $P \cup A \not\equiv P^{\rightarrow} \cup A$. Thus, $P \cup A$ must have the (single) answer set $Lit_{\mathcal{A}}$. In this case, however, our assumption implies that $P_{\leq 1} \cup A$ also has the (single) answer set $Lit_{\mathcal{A}}$. Due to the fact that $(P^{\rightarrow} \cup A)^{Lit_{\mathcal{A}}} = (P_{\leq 1} \cup A)^{Lit_{\mathcal{A}}}$, we conclude that $P^{\rightarrow} \cup A$ has the (single) answer set $Lit_{\mathcal{A}}$. This implies $P \cup A \equiv P^{\rightarrow} \cup A$, again a contradiction. □

7 Conclusion

Uniform equivalence of logic programs, which has been considered earlier for datalog and general Horn logic programs [22, 15], under stable semantics is an interesting concept which can be exploited for program optimization. We have presented characterizations of uniform equivalence in terms of Turner's SE-models [23, 24] (equivalently, HT models [11]), and we have analyzed the computational cost for testing uniform equivalence and related problems.

While we have presented a number of results, there are several issues which remain to be considered. We have found a simple and appealing characterization of uniform equivalence in terms of UE-models for finitary programs, which single out from the SE-models certain maximal models. However, in case of infinite programs, such maximal models do not always exist. It thus would be interesting to see whether also in this case uniform equivalence can be captured by a set of SE-models, or whether a completely novel notion of model (like SE-model for strong equivalence) is needed. Another direction of research is to investigate the usage of uniform equivalence for program replacement and program rewriting in optimization. To this end, it remains to analyze compliances with current optimization techniques and to develop genuine methods which are tailored for uniform equivalence. Researching these issues is part of our ongoing work.

Acknowledgments.

The authors are thank David Pearce for interesting discussions and comments about this work and pointers to related literature. As communicated to us by him, he and Augustin Valverde have generalized some of our results in this paper to the context of equilibrium logic and defined a tableaux-style proof system for uniform equivalence [20]. We are also grateful to the anonymous reviewers of ICLP 2003 for their comments on an extended abstract submitted to this conference, which helped to improve the paper.

A Appendix: Proofs

Proof of Theorem 5.2 (continued).
Let $X' = \{x' \mid x \in X\}$ and $Y' = \{y' \mid y \in Y\}$ and consider models of P. For every model M of P either $w_0 \in M$, or $w_0 \notin M$. In the first case, it is easily seen that $M = X \cup X' \cup Y \cup Y' \cup \{w_0, w_1\} =: M_0$. In the second case, i.e., if $M \neq M_0$, for all $1 \leq i \leq n$ it must hold that $\{x_i, x_i'\} \not\subseteq M$ (otherwise $w_0 \in M$). Furthermore, $w_1 \in M$ must always hold due to the rule $r : w_1 \leftarrow not\, w_1$. Thus, $Y \cup Y' \subseteq M$ holds for every $M \neq M_0$. Consequently, every such M is a choice χ on x_i, x_i', for every $1 \leq i \leq n$, and of form $M = \chi \cup Y \cup Y' \cup \{w_1\} =: M_\chi$. As for the classical models of Q, it is easily verified that they equal those of P.

Now let us consider $P \equiv^u Q$ by using Theorem 3.2. Item (i), $(X, X) \models P$ iff $(X, X) \models Q$, obviously holds. As for item (ii), since P is a subset of Q, $(A, B) \models Q$ implies that $(A, B) \models P$, so there is no violation possible. Conversely, suppose $(A, B) \models P$, $A \subset B$. Note that $P^B = P \setminus \{r\}$ and $Q^B = Q \setminus \{r\}$. That is, w.r.t. the reducts, models not containing w_1 are possible as well. Assume that $A \not\models Q^B$. Then $A \cap (Y \cup Y' \cup \{w_0, w_1\}) = \emptyset$ holds, as well as $\{x_i, x_i'\} \not\subseteq A$, for all $1 \leq i \leq n$. Hence, $A \subseteq X \cup X'$ corresponds to a choice χ on x_i, x_i', for every $1 \leq i \leq n$, i.e., $A = A_\chi$ where $A_\chi = \{\chi(\{x_i, x_i'\}) \mid 1 \leq i \leq n\}$.

Let us consider possible values of B. First, if $B = M_0$, then the model M_χ satisfies $A \subseteq M_\chi \subset M_0$ and $M_\chi \models Q^B$. Thus, condition (ii) is satisfied. Otherwise, $B = M_\chi$ since $A_\chi \subseteq B$; here condition (ii) is satisfied iff there exists a set μ, $\mu \subset Y \cup Y' \cup \{w_1\}$, such that $A \cup \mu \models Q^B$. For this to hold, we must have $w_1 \notin \mu$ and $\mu \cap \{y_1, y_1'\} \neq \emptyset$. Moreover, this means that $|\{y_i, y_i'\} \cap \mu| = 1$, for all $1 \leq i \leq m$, and that $A \cup \mu \not\models D_i^*$, for $1 \leq i \leq l$. That is, $\chi \cup \mu$ corresponds to an assignment to the variables of F such that $\bigvee_{i=1}^{i=l} D_i$ evaluates to false.

In conclusion, $P \not\equiv^u Q$ iff there exists an SE-model (A_χ, M_χ) of P such that $\bigvee_{i=1}^{i=l} D_i\chi$ is a tautology, where $D_i\chi$ denotes substitution of true resp. false for all x_j variables in D_i according to χ. Furthermore, since every D_i contains some literal over Y, we have that $(A_\chi, M_\chi) \models P$ for each choice χ on the x_i, x_i'. Combined with the preceding observation, it follows that $P \not\equiv^u Q$ iff $\bigvee_{i=1}^{i=l} D_i\chi$ is a tautology for some χ, i.e., , $\exists X \forall Y \bigvee_{i=1}^{i=l} D_i\chi$ (=F) is true.

Thus, we have shown that F is false iff $P \equiv^u Q$. □

Proof of Theorem 6.1.
As for strong equivalence, in view of Proposition 2.1, we only need to consider SEE-models of form $(X, Lit_{\mathcal{A}})$.

($\Rightarrow$) For the if-direction, assume P and Q have the same SEE-models. For any program R, $P \cup R$ and $Q \cup R$ have the same SEE-models by Lemma 6.1 and are equivalent. This proves $P \equiv^s Q$.

($\Leftarrow$) For the only-if direction, assume $P \equiv^s Q$. W.l.o.g., suppose that there exists an SEE-model $(X, Lit_{\mathcal{A}})$ of P but not of Q. Then either $Lit_{\mathcal{A}} \not\models Q$, or $X \not\models Q^{Lit_{\mathcal{A}}}$.

In the first case $Lit_{\mathcal{A}}$ is an answer set of $P \cup \{A, \neg A\}$, $A \in \mathcal{A}$, while it is not of $Q \cup \{A, \neg A\}$, contradiction.

So $Lit_{\mathcal{A}} \models Q$ and $X \not\models Q^{Lit_{\mathcal{A}}}$ must hold. This implies $X \subset Lit_{\mathcal{A}}$. Let $R = X \cup \{L \leftarrow L' \mid L, L' \in Lit_{\mathcal{A}} \setminus X\}$. It is easily verified that $(X, Lit_{\mathcal{A}})$ is an SEE-model of $P \cup R$. On the other hand, there is no SEE-model $(Y, Lit_{\mathcal{A}})$ of Q, such that $Y \neq Lit_{\mathcal{A}}$. To see this, observe that by construction of R no consistent strict superset of X can model R. Also, no subset of X can model R and X does not model $Q^{Lit_{\mathcal{A}}}$. As a consequence, $Q \cup R$ has the answer set $Lit_{\mathcal{A}}$, while $P \cup R$ has not, contradicting $P \equiv^s Q$.

We now consider uniform equivalence.

($\Rightarrow$) For the only-if direction, suppose $P \equiv^u Q$. If $Lit_{\mathcal{A}} \not\models P$ and $Lit_{\mathcal{A}} \not\models Q$, then no SEE-model of form $(X, Lit_{\mathcal{A}})$ exists for P and Q, and by Theorem 3.1, the result holds.

W.l.o.g., assume $Lit_{\mathcal{A}} \models P$ and $Lit_{\mathcal{A}} \not\models Q$. Then, there is a constraint c in $Q^{Lit_{\mathcal{A}}}$ with $B^-(c) = \emptyset$, while $P^{Lit_{\mathcal{A}}}$ has no such constraint. Hence, $Lit_{\mathcal{A}}$ is an answer set of $P \cup \{A, \neg A\}$, for any $A \in \mathcal{A}$, while it is not an answer set of $P \cup \{A, \neg A\}$, This contradicts our assumption $P \equiv^u Q$. Hence, (i) must hold.

Assume now that, w.l.o.g. $(X, Lit_{\mathcal{A}})$ is an SEE-model of P but not of Q. In view of (i) it is clear that $X \subset Lit_{\mathcal{A}}$ must hold. Suppose that for no set M, $X \subseteq M \subset Lit_{\mathcal{A}}$, $(M, Lit_{\mathcal{A}})$ is an SE-model of Q. Then, since no proper subset of X models $Q \cup X$, $(Lit_{\mathcal{A}}, Lit_{\mathcal{A}})$ is the only SEE-model of $Q \cup X$ of form $(\cdot, Lit_{\mathcal{A}})$. Thus, $Lit_{\mathcal{A}}$ is an answer set of $Q \cup X$ in this case, while it is not an answer set for $P \cup X$ ($X \models P^{Lit_{\mathcal{A}}}$ implies $X \models (P \cup X)^{Lit_{\mathcal{A}}}$, so $(X, Lit_{\mathcal{A}})$ is an SEE-model of $P \cup X$). This contradicts $P \equiv^u Q$, which proves $(ii.2)$.

($\Leftarrow$) For the if direction, assume that (i) holds and (ii) holds for every SEE-model (X, Y) such that $X \subset Y$.

Suppose that there exists a set of literals, A, such that $P \cup A \not\equiv Q$. If both $P \cup A$ and $Q \cup A$ are contradiction-free, then it follows from Lemma 3.1 that $P \equiv^u Q$, which contradicts $P \cup A \not\equiv Q \cup A$.

Thus, assume w.l.o.g. that $Q \cup A$ has the answer set $Lit_{\mathcal{A}}$ while $P \cup A$ has not.

Then, $P \cup A$ has either no SEE-model of form $(X, Lit_{\mathcal{A}})$, or it has an SEE-model $(X, Lit_{\mathcal{A}})$, such that $X \neq Lit_{\mathcal{A}}$. The first case contradicts $(ii.1)$: Since $Q \cup A$ is inconsistent, $(Lit_{\mathcal{A}}, Lit_{\mathcal{A}})$ is an SEE-model of $Q \cup A$ and hence of Q. Condition $(ii.1)$ implies that $Lit_{\mathcal{A}} \models P$, but then $(Lit_{\mathcal{A}}, Lit_{\mathcal{A}})$ is an SEE-model of P, contradiction. The second case contradicts $(ii.2)$: If $(X, Lit_{\mathcal{A}})$ is an SEE-model of $P \cup A$, then it is also an SEE-model of P. By $(ii.2)$, there exists an SEE-model $(M, Lit_{\mathcal{A}})$ of Q, where $A \subseteq X \subseteq M \subset Lit_{\mathcal{A}}$. Thus, $M \models Q^{Lit_{\mathcal{A}}}$, and since $A \subseteq M$, it also models $(Q \cup A)^{Lit_{\mathcal{A}}}$. This means that $(M, Lit_{\mathcal{A}})$ is also an SEE-model of $Q \cup A$, contradicting the assumption that $Q \cup A$ is contradictory (cought in $M \subset Lit_{\mathcal{A}}$).

This proves that, given (i) and (ii) for every SEE-model (X, Y) such that (X, Y) is an SEE-model of exactly one of the programs P and Q, no set of literals A can exist, such that one of the programs $P \cup A$ and $Q \cup A$ is inconsistent, while the other has no answer set. □

References

[1] R. Ben-Eliyahu and R. Dechter. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence*, 12:53–87, 1994.

[2] R. Ben-Eliyahu and L. Palopoli. Reasoning with minimal models: Efficient algorithms and applications. In *Proceedings Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR-94)*, pages 39–50, 1994.

[3] W. L. Buntine. Generalized subsumption and its applications to induction and redundancy. *Artificial Intelligence*, 36(2):149–176, 1988. Preliminary paper in Proc. ECAI 1986, pp. 15-23.

[4] P. Cabalar. A three-valued characterization for strong equivalence of logic programs. In *Proceedings Eighteenth National Conference on Artificial Intelligence (AAAI '02), July 28 – August 1, 2002, Edmonoton, Alberta (Canada)*, pages 106–111. AAAI Press, July 2002. Extended version "Alternative characterizations for strong equivalence of logic programs"in Proc. NMR 2002.

[5] S. Cosmadakis and P. Kanellakis. Parallel evaluation of recursive rule queries. In *Proceedings PODS-86*, pages 280–293, 1986.

[6] D. J. de Jongh and L. Hendriks. Characterizations of strongly equivalent logic programs in intermediate logics. *Theory and Practice of Logic Programming*, 3(3):259–270, 2003.

[7] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive datalog. *ACM Transactions on Database Systems*, 22(3):364–417, September 1997.

[8] M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Logic Programming: Proceedings Fifth Intl Conference and Symposium*, pages 1070–1080, Cambridge, Mass., 1988. MIT Press.

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

[10] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, C. Koch, C. Mateis, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. Technical Report INFSYS RR-1843-02-14, Institut für Informationssysteme, Technische Universität Wien, A-1040 Vienna, Austria, October 2002. Submitted for publication.

[11] V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Trans. on Computational Logic*, 2(4):526–541, 2001.

[12] V. Lifschitz and H. Turner. Splitting a Logic Program. In *Proceedings ICLP-94*, pages 23–38, Santa Margherita Ligure, Italy, June 1994. MIT-Press.

[13] F. Lin. Reducing strong equivalence of logic programs to entailment in classical propositional logic. In D. Fensel, F. Giunchiglia, D. McGuiness, and M.-A. Williams, editors, *Proceedings Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR-02), April 22-25, Toulouse, France*, pages 170–176. Morgan Kaufmann, 2002.

[14] F. Lin and Y. Zhao. ASSAT: Computing answer sets of a logic program by SAT solvers. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, pages 112-117, Edmonton, Alberta, Canada, 2002. AAAI Press / MIT Press.

[15] M. J. Maher. Equivalences of logic programs. In Minker [16], pages 627–658.

[16] J. Minker, editor. *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufman, Washington DC, 1988.

[17] I. Niemelä, P. Simons, and T. Syrjänen. Smodels: A system for answer set programming. In C. Baral and M. Truszczyński, editors, *Proceedings of the 8th International Workshop on Non-Monotonic Reasoning (NMR'2000)*, Breckenridge, Colorado, USA, April 2000.

[18] M. Osorio, J. NAvarro, and J. Arrazola. Equivalence in answer set programming. In A. Pettorossi, editor, *Proceedings of the 11th International Workshop on Logic Based Program Synthesis and Transformation (LOPSTR 2001), Paphos, Cyprus, November 28-30, 2001*, number 2372 in LNCS, pages 57–75. Springer, 2001.

[19] D. Pearce, H. Tompits, and S. Woltran. Encodings for equilibrium logic and logic programs with nested expressions. In P. Brazdil and A. Jorge, editors, *Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA 2001), Porto, Portugal, December 17-20, 2001*, volume 2258 of *Lecture Notes in Computer Science*, pages 306–320. Springer, 2001.

[20] D. Pearce and A. Valverde. Some types of equivalence for logic programs and equilibrium logic. In *Proc. Joint Conf. Declarative Programming (APPIA-GULP-PRODE)*, 2003.

[21] T. Przymusinski. Stable semantics for disjunctive programs. *New Generation Computing*, 9:401–424, 1991.

[22] Y. Sagiv. Optimizing datalog programs. In Minker [16], pages 659–698.

[23] H. Turner. Strong equivalence for logic programs and default theories (made easy). In T. Eiter, W. Faber, and M. Truszczyński, editors, *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-01)*, number 2173 in LNCS, pages 81–92. Springer, 2001.

[24] H. Turner. Strong equivalence made easy: nested expressions and weight constraints. *Theory and Practice of Logic Programming*, 3(4-5):609–622, 2003.