In: Proc. 19th International Conference on Logic Programming (ICLP 2003), LNCS,


# Uniform Equivalence of Logic Programs under the Stable Model Semantics ⋆

Thomas Eiter and Michael Fink

Institut für Informationssysteme, Abt. Wissensbasierte Systeme,
Technische Universität Wien
Favoritenstraße 9-11, A-1040 Vienna, Austria
{eiter,michael}@kr.tuwien.ac.at

**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 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 [18], DLV [11], or ASSAT [15]. 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* [12, 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

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

$Q$ 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 [13, 5], 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, 16], 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 (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 $Q = \{a \leftarrow \mathit{not}\, b;\ b \leftarrow \mathit{not}\, a\}$. 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 $P = \{a \leftarrow \mathit{not}\, b;\ a \leftarrow b\}$ and $Q = \{a \leftarrow \mathit{not}\, c;\ a \leftarrow c\}$. 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 [3, 4, 14, 12, 19, 23, 24], to our knowledge uniform equivalence of has not been considered. Sagiv [22] 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 [16] considered the property for definite general Horn programs, and reported undecidability. Moreover, both [22, 16] showed that uniform equivalence coincides for the respective programs with Herbrand logical equivalence.

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 [9, 10]. 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 $\Pi_2^P$-complete in the general propositional case, and thus harder than deciding strong equivalence of $P$ and $Q$, which is in $\mathrm{coNP}$ [19, 24]. However, the complexity of testing uniform equivalence decreases on important fragments; in particular, it is $\mathrm{coNP}$-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 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. For space reasons, some proofs are omitted here (see [6] for an extended version).

## 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 $H(r) = \{A_1, \dots, A_l\}$, $B^+(r) = \{A_{l+1}, \dots, A_m\}$, $B^-(r) = \{A_{m+1}, \dots, A_n\}$, 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

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

We call $H(r)$ the *head* of $r$, and $B(r) = \{A_{l+1}, \dots, A_m, \mathit{not}\, A_{m+1}, \dots, \mathit{not}\, A_n\}$ 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 $\perp$ (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 $DLP$s [10, 21], which generalizes the stable model semantics for $NLP$s [9]. 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 $I \models P$, for a program $P$, 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 $P$.

*Equivalences.* Several notions for equivalence of logic programs have been considered, cf. [12, 16, 22]. Under stable semantics, two $DLP$s $P$ and $Q$ are regarded as equivalent, denoted $P \equiv Q$, iff $\mathcal{SM}(P) = \mathcal{SM}(Q)$.

The more restrictive forms of strong equivalence [12] and uniform equivalence [22, 16] 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 [12] 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 1 ([23, 24]).** *For every $DLPs$ $P$ and $Q$, $P \equiv^s Q$ iff $SE(P) = SE(Q)$.*

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

## 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 [12]) 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 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 1 we immediately obtain the following characterization of uniform equivalence between logic programs.

**Theorem 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 $Q = \{a \leftarrow \mathit{not}\, b;\ b \leftarrow \mathit{not}\, a\}$. By Theorem 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 1 the following characterization of uniform equivalence.

**Theorem 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 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 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 $(X, Y) = (M_0, Y)$, $(M_1, Y), \ldots, (M_i, Y), \ldots$ 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. $\square$

Note that the previous theorem remains valid even if only one of $P$ 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.** *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 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, $UE(P) = UE(Q)$. □

This result shows that UE-models capture the notion of uniform equivalence, 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 $Q = \{a \leftarrow \mathit{not}\, b;\ b \leftarrow \mathit{not}\, a\}$ 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 $P = \{a \leftarrow \mathit{not}\, b;\ a \leftarrow b\}$, 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 $c$, $Q = \{a \leftarrow \mathit{not}\, c;\ a \leftarrow c\}$ has the same UE models. Hence, $P$ and $Q$ are uniformly equivalent.

Like Theorem 2, also Theorem 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

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

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 $X_i = \{b_1, \dots, b_i\}$ 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 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 2.** *For any finite program $P$ and rule $r$, if $P \models_u r$ then $P \cup \{r\} \equiv^u P$.*

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 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 Prop. 2 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, 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 $P \models_u Q$. $\square$

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.** *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 $P \models r$.*

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 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 $r$.*

**Lemma 3.** *Let $P$ be a finite DLP. Then, $P \models_u r$ implies $P \models r$, for every rule $r$.*

**Theorem 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 2 and 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.** *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 [12, 23, 24], on the class of positive programs classical equivalence and strong equivalence coincide. By combining this and the previous result, we obtain

**Theorem 6.** *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 $\Pi$ and $\Pi'$ is equivalent to equivalence of $\Pi'$ and $\Pi$ over Herbrand models; this implies the above result for definite Horn programs. Maher [16] showed a generalization of Sagiv's result for definite Horn logic programs with function symbols.

*Example 9.* 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).

### 4.2 Head-cycle free programs

The class of head-cycle free programs generalizes the class of *NLP*s 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 7.** *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 [13, 5].

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

**Theorem 8.** *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 $r^Y$.*

*Example 10.* 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 $\alpha$.

**Theorem 9.** *Given a pair of sets $(X, Y)$ and a program $P$, deciding whether $(X, Y) \in UE(P)$ is $(i)$ $\mathrm{coNP}$-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.*

**Corollary 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 4.** *Given two DLPs $P$ and $Q$, deciding whether $P \equiv^u Q$ is in the class $\Pi_2^P$.*

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 $DLP$s $P$ and $Q$ are not uniformly equivalent, we can by Theorem 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 9, the guess for $(X, Y)$ can be verified in polynomial time with the help of an NP oracle. This proves $\Pi_2^P$-membership of $P \equiv^u Q$. □

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

**Theorem 10.** *Given two DLPs $P$ and $Q$, deciding $P \equiv^u Q$ is $\Pi_2^P$-complete.*

*Proof.* (Sketch) Membership in $\Pi_2^P$ has already been established in Lemma 4. To show $\Pi_2^P$-hardness, we provide a polynomial reduction of evaluating a quantified Boolean formula (QBF) from a fragment which is known $\Pi_2^P$ -complete to deciding uniform equivalence of two $DLP$s $P$ 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 $\Sigma_2^P$-complete; thus deciding whether $F$ is false is $\Pi_2^P$-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:

$$
\begin{array}{lrcll}
P = \{ & x_i \vee x_i' & \leftarrow & . & 1 \leq i \leq n; \\
 & y_i \vee y_i' & \leftarrow & y_j. & 1 \leq i \neq j \leq m; \\
 & y_i \vee y_i' & \leftarrow & y_j'. & 1 \leq i \neq j \leq m;
\end{array}
$$

$$
\begin{array}{llll}
w_0 \leftarrow x_i, x_i'. & 1 \leq i \leq n; & w_1 \leftarrow y_i, y_i'. & 1 \leq i \leq m; \\
x_i \leftarrow w_0. & 1 \leq i \leq n; & w_1 \leftarrow D_i^*. & 1 \leq i \leq l; \\
x_i' \leftarrow w_0. & 1 \leq i \leq n; & y_i \leftarrow w_1. & 1 \leq i \leq m; \\
y_i \leftarrow w_0. & 1 \leq i \leq m; & y_i' \leftarrow w_1. & 1 \leq i \leq m; \\
y_i' \leftarrow w_0. & 1 \leq i \leq m; & w_1 \leftarrow not\, w_1. & \\
w_1 \leftarrow w_0. & & & \},
\end{array}
$$

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 $(A_\chi, M_\chi)$ of $P$ which do not satisfy $Q$, where $\chi$ corresponds to a truth assignment to $X$ and all atoms not in $X \cup X'$ are false, such that $(A_\chi, M_\chi)$ violates uniform equivalence of $P$ and $Q$ via $(ii)$ just if there is no way to make all $D_j$ false by some assignment $\mu$ to $Y$. These candidate models for spoiling uniform equivalence are eliminated iff formula $F$ evaluates to false. Since $P$ and $Q$ are obviously constructible in polynomial time, our result follows. □

The previous result shows that deciding uniform equivalence of *DLP*s $P$ 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 11.** *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 $P \equiv^u Q$ is* coNP*- complete, where* coNP*-hardness holds if $P$ is either positive or a NLP, and $Q$ is Horn.*

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 *NLP*s 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 12.** *Given a DLP $P$ and a rule $r$, deciding $P \models_u r$ is (i) $\Pi_2^P$-complete in general, (ii)* coNP*-complete if $P$ is either positive or head-cycle free, and (iii) polynomial if $P$ is Horn.*

*Proof.* (Sketch) The complementary problem, $P \not\models_u r$, is in $\Sigma_2^P$ 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 9 be verified with a call to a NP-oracle resp. in polynomial time. In case of a positive $P$, by Theorem 5, $P \models_u r$ iff $P \models r$, which is in coNP for general $P$ and polynomial for Horn $P$. The hardness results can be obtained by adapting the constructions in hardness proofs of previous results. $\square$

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 [14], 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 [16].

## 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 [12, 23, 24, 19], i.e., we get the same characterizations and the same complexity ($\Pi_2^P$).

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 set of all literals using strong negation over $\mathcal{A}$.

*Example 11.* 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. 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, 12, 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 $Y \models P \wedge X \models P^Y$.*

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

**Theorem 13.** *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.* $P$*).*

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

**Theorem 14.** *Let* $P$ *and* $Q$ *be positive, extended DLPs. Then* $P \equiv^s Q$ *iff* $P \equiv^u Q$*.*

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 $\Pi_2^P$-hard (resp. coNP-hard).

However, not all properties do carry over. As Example 11 reveals, in general a head-cycle free extended $DLP$ $P$ is no longer equivalent, and hence not uniformly equivalent, to its shift $P^{\leftarrow}$. However, under the condition below $P \equiv^u P^{\leftarrow}$ holds as well. Call any $DLP$ *contradiction-free*, if $Lit_{\mathcal{A}}$ is not an answer set of it.

**Proposition 5.** *Let* $P$ *be a head-cycle free and contradiction-free extended* $DLP$*. Then* $P \equiv^u P^{\leftarrow}$ *iff for each* $A \subseteq Lit_{\mathcal{A}}$*, the program* $P \cup A$ *is contradiction-free if the program* $\{r \in P \mid |H(r)| \leq 1\} \cup A$ *is contradiction-free.*

As shown in [6], "$A \subseteq Lit_{\mathcal{A}}$" can be equivalently replaced by "$A \subseteq Lit_{\mathcal{A}}$ such that $D_H(P) \not\subseteq A$," where $D_H(P) = \{L \mid L \in H(r), |H(r)| > 1, r \in P\}$.

We finally note that Pearce and Valverde have given, inspired by our work, a generalization of our results on UE-models to equilibrium logic [20].

# 7 Conclusion

Uniform equivalence of logic programs, which has been considered earlier for datalog and general Horn logic programs [22, 16], 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 [12]), 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 need not 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. Researching this issue is part of our ongoing work.

Another direction of research is to investigate the usage of uniform equivalence for program replacement and program rewriting in optimization. To this end, in follow-up works [7, 8] we analyzed compliances with current optimization techniques and gave characterizations of programs possessing equivalent programs belonging to syntactic subclasses of disjunctive logic programs under both, uniform and strong equivalence. Furthermore we gave encodings of our characterizations in answer-set programming and investigated the computational complexity of program simplification and determining semantical equivalence.

**Acknowledgments.** We thank the anonymous referees for their valuable comments.

## 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 *Proc. KR'94*, pp. 39–50, 1994.
3. P. Cabalar. A three-valued characterization for strong equivalence of logic programs. In *Proc. AAAI '02*, pp. 106–111, 2002.
4. 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.
5. T. Eiter, G. Gottlob, and H. Mannila. Disjunctive datalog. *ACM TODS*, 22(3):364–417, 1997.
6. T. Eiter and M. Fink. Uniform equivalence of logic programs under the stable model semantics. Tech. Rep. INFSYS RR-1843-03-08, Inst. für Informationssysteme, TU Wien, 2003.
7. T. Eiter, M. Fink, H. Tompits, and S. Woltran. Simplifying logic programs under uniform and strong equivalence. Manuscript, submitted, July 2003.
8. T. Eiter, M. Fink, H. Tompits, and S. Woltran. Eliminating disjunction from propositional logic programs under stable model preservation. Manuscript, submitted, August 2003.
9. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Logic Programming: Proc. Fifth Int'l Conference and Symposium*, pp. 1070–1080, 1988.
10. M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365–385, 1991.
11. 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. Tech. Rep. INFSYS RR-1843-02-14, Inst. für Informationssysteme, TU Wien, 2002.
12. V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Trans. on Computational Logic*, 2(4):526–541, 2001.
13. V. Lifschitz and H. Turner. Splitting a logic program. In *Proc. ICLP-94*, pp. 23–38, 1994.
14. F. Lin. Reducing strong equivalence of logic programs to entailment in classical propositional logic. In *Proc. KR-2002*, pp. 170–176, 2002.
15. F. Lin and Y. Zhao. ASSAT: Computing answer sets of a logic program by SAT solvers. In *Proc. AAAI-2002*, pp. 112–117, 2002.
16. M. J. Maher. Equivalences of logic programs. In Minker [17], pp. 627–658.
17. J. Minker, editor. *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann, 1988.
18. I. Niemelä, P. Simons, and T. Syrjänen. Smodels: A system for answer set programming. In *Proc. 8th Int'l Workshop on Non-Monotonic Reasoning* (*NMR'2000*), 2000.
19. D. Pearce, H. Tompits, and S. Woltran. Encodings for equilibrium logic and logic programs with nested expressions. In *Proc. EPIA 2001*, LNCS 2258, pp. 306–320, 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 [17], pp. 659–698.
23. H. Turner. Strong equivalence for logic programs and default theories (made easy). In *Proc. LPNMR-01*, LNCS 2173, pp. 81–92, 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.