INFSYS

RESEARCH

REPORT

INSTITUT FÜR INFORMATIONSSYSTEME

ARBEITSBEREICH WISSENSBASIERTE SYSTEME

SEMANTICAL CHARACTERIZATIONS AND COMPLEXITY OF EQUIVALENCES IN ANSWER SET PROGRAMMING

Thomas Eiter Michael Fink Stefan Woltran

INFSYS RESEARCH REPORT 1843-05-01

FEBRUARY 2005

Institut für Informationssysteme
Ab. 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 1843-05-01, FEBRUARY 2005

SEMANTICAL CHARACTERIZATIONS AND COMPUTATIONAL ASPECTS OF EQUIVALENCES IN STABLE LOGIC PROGRAMMING

Thomas Eiter[1] and Michael Fink[2] and Stefan Woltran[3]

Abstract. In recent research on non-monotonic 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 answer sets for any other program R. This property strengthens equivalence of P and Q with respect to answer sets (which is the particular case for $R = \emptyset$), and has its applications in program optimization, verification, and modular logic programming. In this paper, we consider more liberal notions of strong equivalence, in which the actual form of R may be syntactically restricted. On the one hand, we consider uniform equivalence, where R is a set of facts rather than a set of rules. This notion, which is well known in the area of deductive databases, is particularly useful for assessing whether programs P and Q are equivalent as components of a logic program which is modularly structured. On the other hand, we consider relativized notions of equivalence, where R ranges over rules over a fixed alphabet, and thus generalize our results to relativized notions of strong and uniform equivalence. For all these notions, we consider disjunctive logic programs in the propositional (ground) case, as well as some restricted classes, provide semantical characterizations and analyze the computational complexity. Our results, which naturally extend to answer set semantics for programs with strong negation, 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: answer set semantics, stable models, computational complexity, program optimization, uniform equivalence, strong equivalence.

---

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

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

[3]Institute of Information Systems, Knowledge-Based Systems Group, TU Vienna, Favoritenstraße 9-11, A-1040 Vienna, Austria. E-mail: stefan@kr.tuwien.ac.at.

Acknowledgements: This work was partially supported by the Austrian Science Fund (FWF) under project P18019-N04, as well as by the European Commission under projects FET-2001-37004 WASP, IST-2001-33570 COLOGNET, and IST-2001-33570 INFOMIX.

This report extends INFSYS RR-1843-03-08. The contents were partially published in Proceedings 19th International Conference on Logic Programming (ICLP 2003), LNCS, Springer, 2003; and Proceedings 9th European Conference on Logics in Artificial Intelligence (JELIA 2004), LNCS, Springer, 2004; see [16, 58].

Contents

1 Introduction **1**

2 Preliminaries **5**

3 Uniform Equivalence **7**
3.1 A Characterization for Uniform Equivalence 7
3.2 Introducing UE-Models 9
3.3 Consequence under Uniform Equivalence 12

4 Relativized Notions of Strong and Uniform Equivalence **14**
4.1 A Characterization for Relativized Strong Equivalence 15
4.2 A Characterization for Relativized Uniform Equivalence 17
4.3 Properties of Relativized Equivalences 20

5 Restricted Classes of Programs **21**
5.1 Positive Programs 21
5.2 Head-cycle free programs 24

6 Computational Complexity **26**
6.1 Complexity of Uniform Equivalence 29
6.2 Complexity of Relativized Equivalence 34
6.3 Complexity of Bounded Relativization 40

7 Language Variations **43**
7.1 Extensions in the Propositional Case 43
7.2 DATALOG programs 44

8 Conclusion and Further Work **45**

A Proofs **50**
A.1 Proof of Lemma 5 50
A.2 Proof of Theorem 11 51
A.3 Proof of Theorem 23 51
A.4 Proof of Theorem 24 51
A.5 Proof of Theorem 30 52
A.6 Proof of Theorem 32 53
A.7 Proof of Theorem 33 54

1 Introduction

In the last decade, the approach to reduce finding solutions of a problem to finding "models" of a logical theory has gained increasing importance as a declarative problem solving method. The idea is that a problem at hand is encoded to a logical theory, such that the models of this theory correspond to the solutions of the problem, in a way such that from an arbitrary model of the theory, the corresponding solution can be extracted efficiently. Given that the mappings can be computed in polynomial time, this facilitates polynomial time problem solving modulo the computation of a model of the constructed logical theory, for which an efficient solver may be used. An example of a fruitful application of this approach is [33], which showed that planning problems can be competitively solved by encodings to the classical propositional satisfiability problem (SAT) and running efficient SAT solvers. Encodings of planning problems to nonclassical logics, in particular to non-monotonic logic programs, have later been given in [55, 12, 36, 15]. Because of the features of non-monotonic negation, such programs allow for a more natural and succinct encoding of planning problems than classical logic, and thus are attractive from a declarative point of view.

Given this potential, encoding problems to non-monotonic logic programs under the answer set semantics [24, 25], which is now known as *Answer-Set Programming (ASP)* [50], has been considered in the recent years for a broad range of other applications including knowledge-base updates [59, 30, 1, 18], linguistics [23], security requirements engineering [26], or symbolic model checking [28] as well, to mention some of them. Many of these applications are realized via dedicated languages (see, for instance, [14]) using ASP solvers as back-ends in which a specified reasoning task is translated into a corresponding logic program. Thus, an ever growing number of programs is *automatically generated*, leaving the burden of optimizations to the underlying ASP system.

Despite the high sophistication of current ASP-solvers like [54, 35, 41, 2], their current support for optimizing the programs is restricted in the sense that optimizations are mainly geared towards on-the-fly model generation. In an ad-hoc manner, program optimization aims at simplifying an input program in a way such that the resulting program has the same answer sets. This is heavily exploited in the systems Smodels [54] and DLV [35], for instance, when variables are eliminated from programs via grounding.

However, such optimization can only be applied to the entire program. Local simplifications in parts of the program may not be correct at the global level, since by the non-monotonicity of answer set semantics, adding the same rules to equivalent programs may lead to programs with different models. This in particular hampers an offline optimization of programs to which at run-time further rules are added, which is important in different respects. Regarding code reuse, for instance, a program may be used as a "subprogram" or "expanded macro" within the context of another program (for example, to nondeterministically choose an element from a set), and thus be utilized in many applications. On the other hand, a problem encoding in ASP usually consists of two parts: a generic problem specification and instance-specific input (for example, 3-colorability of a graph in general and a particular graph); here, an offline simplification of the generic part is desirable, regardless of the concrete input at run-time.

As pointed out by several authors [37, 16, 45], this calls for stronger notions of equivalence. As discussed below, there are different ways to access this problem, depending on the actual context of application and optimization. Accordingly, different notions of equivalence may serve as a theoretical basis for optimization procedures. In this paper, we present a first systematic and thorough exploration of different notions of equivalence for answer set semantics with respect to semantical characterizations and computational complexity. It provides a theoretical underpinning for advanced methods of program optimization and for enhanced ASP application development, as well as a potential basis for the development of ASP debugging tools. In the following, we recall some notions of equivalence that have been considered for answer set semantics, illustrated with some examples.

Notions of Equivalence. A notion of equivalence which is feasible for the issues discussed above is *strong equivalence* [37, 56]: 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 answer set semantics, i.e., have the same answer sets. 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.

Example 1 *The programs $P_1 = \{a \vee b\}$ and $Q_1 = \{a \vee b;\ a \leftarrow not\, b\}$ are strongly equivalent. Intuitively, the rule $a \leftarrow not\, b$ in Q is redundant since under answer set semantics, a will be derived from the disjunction $a \vee b$ if b is false. On the other hand, the programs $P_2 = \{a \vee b\}$ and $Q_2 = \{a \leftarrow not\, b;\ b \leftarrow not\, a\}$ are not strongly equivalent: $P_2 \cup \{a \leftarrow b;\ b \leftarrow a\}$ has the answer set $\{a, b\}$, which is not an answer set of $Q_2 \cup \{a \leftarrow b;\ b \leftarrow a\}$.*

Note that strong equivalence is, in general, suitable as a theoretical basis for local optimization. However, it is a very restrictive concept. There are two fundamental options to weaken it and obtain less restrictive notions. On the one hand, one can restrict the syntax of possible program extensions R, or one can restrict the set of atoms occurring in R.

The first approach leads us to to the well known notion of *uniform equivalence* [52, 43]. 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 answer sets. That strong equivalence and uniform equivalence are different concepts is illustrated by the following simple example.

Example 2 *It can be checked that the programs P_2 and Q_2 from Example 1, while not strongly equivalent, are uniformly equivalent. We note that by adding the constraint $\leftarrow a, b$ to them, the resulting programs $P_3 = \{a \vee b;\ \leftarrow a, b\}$ and $Q_3 = \{a \leftarrow not\, b;\ b \leftarrow not\, a;\ \leftarrow a, b\}$, which both express exclusive disjunction of a and b, are strongly equivalent (and hence also uniformly equivalent).*

This example may suggest that disjunction is an essential feature to make a difference between strong and uniform equivalence. In fact this is not the case, as shown by the following example.

Example 3 *Let $P_4 = \{a \leftarrow not\, b;\ a \leftarrow b\}$ and $Q_4 = \{a \leftarrow not\, c;\ a \leftarrow c\}$. Then, it is easily verified that P_4 and Q_4 are uniformly equivalent. However, they are not strongly equivalent: For $P_4 \cup \{b \leftarrow a\}$ and $Q_4 \cup \{b \leftarrow a\}$, we have that $S = \{a, b\}$ is a answer set of $Q_4 \cup \{b \leftarrow a\}$ but not of $P_4 \cup \{b \leftarrow a\}$.*

As for program optimization, compared to strong equivalence, uniform equivalence is more 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 may output facts to a higher layer [39, 22]. In particular, the applies to the typical ASP setting outlined above, in which a generic problem specification component receives problem-specific input as a set of facts.

However, as mentioned before, a different way to obtain weaker equivalence notions than strong equivalence is to restrict the alphabet of possible program extensions. This is of particular interest, whenever one wants to *exclude* dedicated atoms from program extensions. Such atoms may play the role of internal atoms in program components and are considered not to appear anywhere else in the complete program P. This notion of equivalence was originally suggested by Lin in [40] but not further investigated. We will formally define *strong equivalence relative to a given set of atoms* A of two programs P and Q as the test whether, for all sets of rules S over a given set of atoms A, $P \cup S$ and $Q \cup S$ have the same answer sets.

Finally, we introduce the notion of *uniform equivalence relative to a given set of atoms* A, as the property that for two programs P and Q and for all sets $F \subseteq A$ of facts, $P \cup F$ and $Q \cup F$ have the same answer sets. Note that relativized uniform equivalence generalizes the notion of equivalence of DATALOG programs in deductive databases [53]. There, DATALOG programs are called equivalent, if it holds that they compute the same outputs on any set of external atoms (which are atoms that do not occur in any rule head) given as input. The next example illustrates that relativization weakens corresponding notions of equivalence.

Example 4 *Let* $P_5 = \{a \vee b\}$ *and* $Q_5 = \{a \leftarrow not\, b;\ b \leftarrow not\, a;\ c \leftarrow a, b;\ \leftarrow c\}$. *The programs* P_5 *and* Q_5 *have the same answer set, but are neither uniformly equivalent nor strongly equivalent. In particular, it is sufficient to add the fact* c. *Then,* $P_5 \cup \{c\}$ *has* $\{a, b, c\}$ *as an answer set, while* $Q_5 \cup \{c\}$ *has no answer set. However, if we exclude* c *from the alphabet of possible program extensions, uniform equivalence holds. More specifically,* P *and* Q *are uniformly equivalent relative to for any set of atoms* A *such that* $c \notin A$. *On the other hand,* P *and* Q *are not strongly equivalent relative to any* A *which includes both* a *and* b. *The reason is that adding* $a \leftarrow b$ *and* $b \leftarrow a$ *leads to different answer sets (cf. Example 1).*

Main Contributions. In this paper, we study semantical and complexity properties of the above notions of equivalence, where we focus on the propositional case (to which first-order logic programs reduce by instantiation). Our main contributions are briefly summarized as follows.

- We provide characterizations of uniform equivalence of logic programs. To this aim, we build on the concept of *strong-equivalence models (SE-models),* which have been introduced for characterizing strong equivalence [56, 57] in logic programming terms, resembling an earlier characterization of strong equivalence in terms of equilibrium logic which builds on the intuitionistic logic of here and there [37]. A strong equivalence model of a program P is a pair (X, Y) of (Herbrand) interpretations such that $X \subseteq Y$, Y is a classical model of P, and X is a model of the Gelfond-Lifschitz reduct P^Y of P with respect to Y [24, 25]. Our characterizations of uniform equivalence will elucidate the differences between strong and uniform equivalence, as illustrated in the examples above, such that they immediately become apparent.

- For the finitary case, we provide a mathematical 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. Informally, those SE-models (X, Y) of a program P are UE-models, such that either X equals Y or is a maximal proper subset of Y. On the other hand, we show that uniform equivalence of infinite programs cannot be captured by any class of SE-models in general. Furthermore, the notion of logical consequence from UE-models, $P \models_u Q$, turns out to be interesting since programs P and Q are uniformly equivalent if and only if $P \models_u Q$ and $Q \models_u P$ holds. Therefore, logical consequence (relative to UE-models) can be fruitfully used to determine redundancies under uniform equivalence.

- By suitably generalizing the characterizations of strong and uniform equivalence, and in particular SE-models and UE-models, we also provide suitable *semantical characterizations* for both relativized strong and uniform equivalence. Our new characterizations thus capture all considered notions of equivalence (including ordinary equivalence) in a uniform way. Moreover, we show that relativized strong equivalence shares an important property with strong equivalence: constraining possible program extensions to sets of rules of the form $A \leftarrow B$, where A and B are atoms, does not lead to a different concept (Corollary 3). The observation of Pearce and Valverde [49] that uniform and strong equivalence are essentially the only concepts of equivalence obtained by varying the *logical* form of the program extensions therefore generalizes to relative equivalence.

• Besides the general case, we consider various major syntactic subclasses of programs, in particular Horn programs, positive programs, disjunction-free programs, and head-cycle free programs [4], and consider how these notions of equivalence relate among each other. For instance, we establish that for positive programs, all these notions coincide, and therefore only the classical models of the programs have to be taken into account for equivalence testing. Interestingly, for head-cycle free programs, eliminating disjunctions by shifting atoms from rule heads to the respective rule bodies preserves (relativized) uniform equivalence, while it affects (relativized) strong equivalence in general.

• We thoroughly analyze the computational complexity of deciding (relativized) uniform equivalence and relativized strong equivalence, as well as the complexity of model checking for the corresponding model-theoretic characterizations. We show that deciding uniform equivalence of programs P and Q is Π_2^P-complete in the general propositional case, and thus harder than deciding strong equivalence of P and Q, which is coNP-complete [47, 40, 57]. The relativized notions of equivalence have the same complexity as uniform equivalence in general (Π_2^P-completeness). These results reflect the intuitive complexity of equivalence checking using the characterizations we provide. Furthermore, we consider the problems for subclasses and establish coNP-completeness results for important fragments, including positive and head-cycle free programs, and thus obtain a complete picture of the complexity-landscape, which is summarized in Table 2. Some of the results obtained are surprising; for example, checking relativized uniform equivalence of head-cycle free programs, is *easier* than deciding relativized strong equivalence. For an overview and discussion of the complexity results, we refer to Section 6.

• Finally, we address extensions of our results w.r.t. modifications in the language of propositional programs, viz. addition of strong negation or nested expressions, as well as disallowing constraints. Moreover, we briefly discuss the general DATALOG-case.

Our results extend recent results on strong equivalence of logic programs, and pave the way for optimization of logic programs under answer set semantics by exploiting either strong equivalence, uniform equivalence, or relativized notions thereof.

Related Work. While strong equivalence of logic programs under answer set semantics has been considered in a number of papers [7, 11, 40, 37, 45, 47, 56, 57, 46, 48], investigations on uniform equivalence just started with preliminary parts of this work [16]. Recent papers on program transformations [20, 19] already take both notions into account. In the case of DATALOG, uniform equivalence is a well-known concept, however. Sagiv [52], 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 the undecidability of equivalence testing for DATALOG programs [53]. Also Maher [43] considered uniform equivalence for definite general Horn programs (with function symbols), and reported undecidability. Moreover, both [52, 43] showed that uniform equivalence coincides for the respective programs with Herbrand logical equivalence. Maher also pointed out that for DATALOG programs, this result has been independently established by Cosmadakis and Kanellakis [10]. Finally, a general notion of equivalence has also been introduced by Inoue and Sakama [31]. In their framework, called *update equivalence*, one can exactly specify a set of arbitrary rules which may be added to the programs under consideration and, furthermore, a set of rules which may be deleted. However, for such an explicit enumeration of rules for program extension, respectively modification, it seems to be much more complicated to obtain simple semantical characterizations.

The mentioned papers on strong equivalence mostly concern logical characterizations. In particular, the seminal work by Lifschitz *et al.* [37] showed that strong equivalence corresponds to equivalence in the non-classical logic of here-and-there. De Jongh and Hendriks [11] generalized this result by showing that strong equivalence is characterized by equivalence in all intermediate logics lying between here-and-there (upper bound) and the logic KC of weak excluded middle [34] (lower bound) which is axiomatized by intuitionistic logic together with the schema $\neg\varphi \vee \neg\neg\varphi$. In addition, [7] presents another multi-valued logic known as L_3 which can be employed to decide strong equivalence in the same manner. However, the most popular semantical characterization was introduced by Turner [56, 57]. He abstracts from the Kripke-semantics as used in the logic of here-and-there, resulting in the above mentioned *SE-models*. Approaches to implement strong equivalence can be found in [20, 32, 47]. Complexity characterizations of strong equivalence were given by several authors [47, 40, 57]. Our work refines and generalizes this work by considering (relativized) strong equivalence also for syntactic fragments, which previous work did not pay much attention to. As well, we present a new syntactical criterion to retain strong equivalence when transforming head-cycle free programs to disjunction-free ones, complementing work on program transformations [19, 20, 45, 49]. The recent work by Pearce and Valverde [49] addresses strong equivalence of programs over disjoint alphabets which are synonymous under structurally defined mappings.

Structure of the paper. The remainder of this paper is organized as follows. The next section recalls important concepts and fixes notation. After that, in Section 3, we present our characterizations of uniform equivalence. We also introduce the notions of UE-model and UE-consequence and relate the latter to other notions of consequence. Then, Section 4 introduces the relativized notions of equivalence, and we present our generalized characterizations in model-theoretic terms. Section 5 considers two important classes of programs, in particular positive and head-cycle free logic programs, which include Horn and normal logic programs, respectively. The subsequent Section 6 is devoted to a detailed analysis of complexity issues, while Section 7 considers possible extensions of our results to nested logic programs and answer set semantics for programs with strong negation (also allowing for inconsistent answer sets), as well as to DATALOG programs. The final Section 8 concludes the paper and outlines issues for further research.

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 *disjunctive fact* if $B(r) = \emptyset$, and r is a (non-disjunctive) *fact* if $B(r) = \emptyset$ and $l = 1$, both 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 definite Horn rule is called *unary* iff its body contains at most one atom.

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, Horn, or unary), if all rules in P are normal (resp., definite, positive, Horn, unary). Furthermore, a program P is *head-cycle free* (HCF) [4], if each each $r \in P$ is head-cycle free (in P), i.e., if the dependency graph of P (which is defined as usual) where literals of form $not\, A$ are disregarded, has no directed cycle that contains two atoms belonging to $H(r)$.

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. The set of all atoms occurring in a program P is denoted by $Atm(P)$.

We shall deal with further variations of the syntax, where either *strong* negation is available or constraints are disallowed in Section 7. There we shall also briefly discuss how to apply our results to programs with nested expressions [38] or to non-ground programs directly.

We recall the answer set semantics for DLPs [25], which generalizes the answer set semantics for NLPs [24]. 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 $A \in B^-(r)$ is false in I, i.e., $A \notin I$. Furthermore, I models rule r, i.e., $I \models r$ 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 $r \in P$. If $I \models P$ (resp. $I \models r$), I is called a *model* of P (resp. r).

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 $I \cap B^-(r) = \emptyset$; otherwise r^I is void. Note that a void rule has any interpretation as its model. The *Gelfond-Lifschitz reduct* P^I, of a program P is $P^I = \{r^I \mid r \in P\}$. An interpretation I is an *answer set* (or a *stable model* [51]) of a program P iff I is a minimal model (under inclusion $\subseteq$) of P^I. By $\mathcal{AS}(P)$ we denote the set of all answer sets of P.

Several notions for equivalence of logic programs have been considered, cf. [37, 43, 52]. In answer set programming, two DLPs P and Q are regarded as equivalent, denoted $P \equiv Q$, iff $\mathcal{AS}(P) = \mathcal{AS}(Q)$.

The more restrictive form of strong equivalence [37] is as follows.

Definition 1 *Let P and Q be two DLPs. Then, 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., $P \cup R \equiv Q \cup R$.*

One of the main results of [37] 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 [56, 57]:

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. For a single rule r, we write $SE(r)$ instead of $SE(\{r\})$.*

Strong equivalence can be characterized as follows.

Proposition 1 ([56, 57]) *For every DLPs P and Q, $P \equiv_s Q$ iff $SE(P) = SE(Q)$.*

To check strong equivalence of two programs P and Q, it is obviously sufficient to consider SE-interpretations (X, Y) over $Atm(P \cup Q)$, i.e., with $X \subseteq Y \subseteq Atm(P \cup Q)$. We implicitly make use of this simplification when convenient.

Example 5 *Reconsider the examples from the introduction. First take programs $P = \{a \vee b\}$ and $Q = \{a \leftarrow not\, b;\; b \leftarrow not\, a\}$. We have*[1]

$$SE(P) \;=\; \{(a, a);\; (b, b);\; (a, ab);\; (b, ab);\; (ab, ab)\};$$

[1]To ease notation, we write abc instead of $\{a, b, c\}$, a instead of $\{a\}$, etc.

$$SE(Q) \;=\; \{(\emptyset, ab);\ (a,a);\ (b,b);\ (a,ab);\ (b,ab);\ (ab,ab)\}.$$

Thus, $(\emptyset, ab)$ *is SE-model of* Q *but not of* P*. This is due to the fact that* $P^{\{a,b\}} = \{a \vee b\}$ *and* $Q^{\{a,b\}}$ *is the empty program. The latter is modelled by the empty interpretation, while the former is not. Hence, we derive* $P \not\equiv_s Q$*.*

Example 6 *For the second example,* $P = \{a \leftarrow not\, b;\ a \leftarrow b\}$ *and* $Q = \{a \leftarrow not\, c;\ a \leftarrow c\}$*, we also get* $P \not\equiv_s Q$*. In this case, we have:*

$$SE(P) \;=\; \{(\emptyset, ab);\ (\emptyset, abc);\ (c, abc)\} \cup S;$$
$$SE(Q) \;=\; \{(\emptyset, ac);\ (\emptyset, abc);\ (b, abc)\} \cup S;$$

with $S = \{(X,Y) \mid \{a\} \subseteq X \subseteq Y \subseteq \{a,b,c\}\}$*. This shows* $P \not\equiv_s Q$*.*

Note that from the proofs of the results in [37, 57], it appears that for strong equivalence, only the addition of unary rules is crucial. That is, by constraining the rules in the set R in the definition of strong equivalence to normal rules having at most one positive atom in the body does not lead to a different concept. This is encountered by restriction to facts (i.e., empty rule bodies), however.

As well, answer sets of a program can be characterized via its SE-models as follows:

Proposition 2 *For any DLP* P*,* $Y \in \mathcal{AS}(P)$ *iff* $(Y,Y) \in SE(P)$ *and* $(X,Y) \in SE(P)$ *implies* $X = Y$*, for any* X*.*

Finally, we define a consequence relation associated to SE-models.

Definition 3 *Let* P *be a DLP and* r *a rule. Then,* r *is a SE-consequence of* P*, denoted* $P \models_s r$*, iff for each* $(X,Y) \in SE(P)$*, it holds that* $(X,Y) \in SE(r)$*. Furthermore, we write* $P \models_s Q$ *iff* $P \models_s r$*, for every* $r \in Q$*.*

Proposition 3 *For any DLP* P *and* Q*,* $P \equiv_s Q$ *iff* $P \models_s Q$ *and* $Q \models_s P$*.*

Thus, the notion of SE-consequence captures strong equivalence of logic programs.

3 Uniform Equivalence

After the preliminary definitions, we now turn to the issue of uniform equivalence of logic programs. We follow the definitions of uniform equivalence in [52, 43].

Definition 4 *Let* P *and* Q *be two DLPs. Then,* 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.,* $P \cup F \equiv Q \cup F$*.*

3.1 A Characterization for Uniform Equivalence

We proceed by characterizing uniform equivalence of logic programs in model-theoretic terms. As restated above, strong equivalence can be captured by the notion of SE-model (equivalently, HT-model [37]) 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 (X', Y), $X \subset X' \subset Y$, 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 $P \cup Y$, $Y \in \mathcal{AS}(P \cup Y)$, while $Y \notin \mathcal{AS}(Q \cup Y)$. This contradicts our assumption $P \equiv_u Q$. Hence, (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 X', $X \subset X' \subset Y$, it holds that (X', 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, $Y \in \mathcal{AS}(Q \cup X)$ in this case, while $Y \notin \mathcal{AS}(P \cup X)$ ($X \models P^Y$ 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 X' such that $X \subset X' \subset Y$, (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 (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 F and X, such that w.l.o.g., $X \in \mathcal{AS}(P \cup F) \setminus \mathcal{AS}(Q \cup F)$. Since $X \in \mathcal{AS}(P \cup F)$, we have that $F \subseteq X$, and, moreover, $X \models P$. Consequently, (X, X) is an SE-model of P. Since $X \notin \mathcal{AS}(Q \cup F)$, either $X \not\models (Q \cup F)^X$, or there exists $Z \subset X$ such that $Z \models (Q \cup F)^X$.

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

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 F and Z exists such that Z is an answer set of exactly one of $P \cup F$ and $Q \cup F$. That is, $P \equiv_u Q$ holds. □

From Lemma 1 we immediately obtain the following characterization of uniform equivalence of logic programs.

Theorem 1 *Two DLPs, P and Q are uniformly equivalent, $P \equiv_u Q$, iff, for interpretations X, Y,*

(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 X', such that $X \subseteq X' \subset Y$, and (X', Y) is an SE-model of Q (respectively P).

Example 7 *Reconsider the programs $P = \{a \vee b\}$ and $Q = \{a \leftarrow not\, b;\ b \leftarrow not\, a\}$. By Theorem 1, we can easily verify that P and Q are uniformly equivalent: Their SE-models differ only in $(\emptyset, ab)$, 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, ab) is an SE-model of P, and thus item (ii) also holds for $(\emptyset, ab)$.

Recall that P and Q are strongly equivalent after adding the constraint $\leftarrow a, b$, which enforces exclusive disjunction (see Example 2). Uniform equivalence does not require such an addition.

From Theorem 1 we can derive the following characterization of uniform equivalence.

Theorem 2 *Two DLPs P and Q, such that at least one of them is finite, are uniformly equivalent, i.e., $P \equiv_u Q$, iff the following conditions hold:*

- (i) *for every X, (X, X) is an SE-model of P iff it is an SE-model of Q, and*
- (ii) *for every SE-model $(X, Y) \in SE(P) \cup SE(Q)$ such that $X \subset Y$, there exists an SE-model $(X', Y) \in SE(P) \cap SE(Q)$ (=$SE(P \cup Q)$) such that $X \subseteq X' \subset Y$.*

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 (X_1, Y), $X \subseteq X_1 \subset Y$, such that (X_1, Y) is in $SE(P) \cup SE(Q)$. By repeating this argument, we obtain a chain of SE-models $(X, Y) = (X_0, Y), (X_1, Y), \ldots, (X_i, Y), \ldots$ such that $(X_i, Y) \in SE(P) \cup SE(Q)$ and $X_i \subseteq X_{i+1}$, for all $i \geq 0$. Furthermore, we may choose X_1 such that X_1 coincides with Y on all atoms which do not occur in $P \cup Q$ (and hence all X_i, $i \geq 1$, do so). Since one of P and Q is fi nite, it follows that $X_i = X_{i+1}$ must hold for some $i \geq 0$ and hence $(X_i, Y) \in SE(P) \cap SE(Q)$ must hold. This proves the result. □

3.2 Introducing UE-Models

In the light of this result, we can capture uniform equivalence of fi nite programs by the notion of UE-models defi ned as follows.

Definition 5 (UE-model) *Let P be a DLP. Then, $(X, Y) \in SE(P)$ 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 total SE-models (Y, Y) of a DLP plus all its *maximal* non-total SE-models (X, Y), with $X \subset Y$. Formally,

$$UE(P) \quad = \quad \{(Y, Y) \in SE(P)\} \;\cup\; max_{\geq}\{(X, Y) \in SE(P) \mid X \subset Y\};$$

where $(X', Y') \geq (X, Y)$ iff jointly $Y' = Y$ and $X \subseteq X'$.

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

Theorem 3 *Let P and Q be DLPs. Then,*

- (a) *$P \equiv_u Q$ implies $UE(P) = UE(Q)$;*
- (b) *$UE(P) = UE(Q)$ implies $P \equiv_u Q$, whenever at least one of the programs P, Q is finite.*

Proof. For proving (a), let $P \equiv_u Q$. Then, by Theorem 1 (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 Theorem 1 (ii), there exists (X', Y), $X \subseteq X' \subset Y$, which is an SE-model of Q, and by a further application, the existence of (X'', Y), $X' \subseteq X'' \subset Y$, which is an SE-model of P follows. Since $X \subset X''$ contradicts $(X, Y) \in UE(P)$, let $X'' = X' = X$, i.e., (X, Y) is an SE-model of Q as well, but it is not in $UE(Q)$. Hence, there exists $(Z, Y) \in SE(Q)$, $X \subset Z \subset Y$ and, again by Theorem 1 (ii), there exists (Z', Y), $Z \subseteq Z' \subset Y$, which is an SE-model of P. This again contradicts $(X, Y) \in UE(P)$. Hence, $UE(P) = UE(Q)$ must hold.

For (b), assume $UE(P) = UE(Q)$, and w.l.o.g. let P be finite. Since $UE(P) = UE(Q)$ implies Theorem 1 (i), towards a contradiction, suppose that Theorem 1 (ii) is not satisfied, i.e., there exists $X \subset Y$, such that either (1) $(X, Y) \in SE(P)$ and not exists $X \subseteq X' \subset Y$, $(X', Y) \in SE(Q)$, or vice versa (2) $(X, Y) \in SE(Q)$ and not exists $X \subseteq X' \subset Y$, $(X', Y) \in SE(P)$.
Case (1): We show the existence of a set Z, $X \subseteq Z \subset Y$, such that $(Z, Y) \in UE(P)$. If $(X, Y) \in UE(P)$, or Y is finite, this is trivial. So let $(X, Y) \notin UE(P)$ and Y infinite. Then $Y_P = Y \cap Atm(P)$ and $X_P = X \cap Atm(P)$ are finite, $(X_P, Y_P) \in SE(P)$, and $X_P \subset Y_P$. (To see the latter, observe that otherwise we end up in a contradiction by the fact that then $X_P \models P$, hence $X \models P$, and thus $(X, X) \in UE(P) = UE(Q)$, which implies $(X, Y) \in SE(Q)$, since $(Y, Y) \in UE(Q) = UE(P)$ holds.) Since Y_P is finite, there exists a set Z_P, $X_P \subseteq Z_P \subset Y_P$, such that $(Z_P, Y_P) \in UE(P)$. Now, let $Z = Z_P \cup (Y \setminus Y_P)$. Then $X \subseteq Z \subset Y$ holds by construction. Furthermore $(Z, Y) \in UE(P)$, since $Y \setminus Z = Y_P \setminus Z_P$, $P^Y = P^{Y_P}$, and $(Z_P, Y_P) \in UE(P)$. By our assumption $(Z, Y) \in UE(Q)$ follows. Contradiction.
Case (2): We show the existence of a set Z, $X \subseteq Z \subset Y$, such that $(Z, Y) \in UE(Q)$. If $(X, Y) \in UE(Q)$, or Y is finite, this is trivial. So let $(X, Y) \notin UE(Q)$, and Y infinite. Futhermore, $Y \setminus X \subseteq Atm(P)$ must hold. (To see the latter, observe that otherwise we end up in a contradiction by taking any atom $a \in Y \setminus X$, such that $a \notin Atm(P)$, and considering $Z = Y \setminus \{a\}$. Then $X \subseteq Z \subset Y$ holds by construction and since $(Y, Y) \in UE(P) = UE(Q)$, $Y \models P$ and so does Z, i.e., $(Z, Y) \in SE(P)$, a contradiction.) However, since $Atm(P)$ is finite, this means that $Y \setminus X$ is finite, i.e., there cannot exist an infinite chain of SE-models $(X, Y) = (X_0, Y), (X_1, Y), \ldots, (X_i, Y), \ldots$, such that $X_i \subset X_j \subset Y$, for $i < j$, and $(X_i, Y) \in SE(Q)$. Thus, there exists a maximal model $(Z, Y) \in UE(Q)$. By our assumption $(Z, Y) \in UE(P)$ follows. Contradiction. Thus, Theorem 1 (ii) holds as well, proving $P \equiv_u Q$ in Case (b). $\Box$

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.

Corollary 1 *Two finite DLPs P and Q are uniformly equivalent, i.e., $P \equiv_u Q$, if and only if $UE(P) = UE(Q)$.*

Example 8 *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 $(\emptyset, ab)$, and all of its SE-models except this one are UE-models of Q. Thus,*

$$UE(P) = UE(Q) = \{(a, a);\ (b, b);\ (a, ab);\ (b, ab);\ (ab, ab)\}.$$

Note that the strong equivalence of P and Q fails because $(\emptyset, ab)$ 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 $(X_1 \cap X_2, Y) \in SE(P)$). This intersection property is satisfied by the Horn program Q^Y, but violated by the disjunctive program P^Y (=P). The maximality condition of UE-models eliminates this intersection property.

Example 9 *Reconsider* $P = \{a \leftarrow not\, b;\ a \leftarrow b\}$*, which has classical models (over* $\{a, b, c\}$*) of form* $\{a\} \subseteq Y \subseteq \{a, b, c\}$*. Its UE-models are* (X, 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.*

The following example shows why the characterization via UE-models fails if both compared programs are infinite. The crucial issue here is the expression of an "infinite chain" resulting in an infinite number of non-total SE-models. In this case, the concept of maximal non-total SE-models does not capture the general characterization from Theorem 1.

Example 10 *Consider the programs* P *and* Q *over* $\mathcal{A} = \{a_i \mid i \geq 1\}$*, defined by*

$$P = \{a_i \leftarrow \mid i \geq 1\}, \quad \text{and} \quad Q = \{a_i \leftarrow not\, a_i,\ a_i \leftarrow a_{i+1} \mid i \geq 1\}.$$

Both P *and* Q *have the single classical model* $\mathcal{A} = \{a_i \mid i \geq 1\}$*. Furthermore,* P *has no "incomplete" SE-model* $(X, \mathcal{A})$ *such that* $X \subset \mathcal{A}$*, while* Q *has the incomplete SE-models* $(X_i, \mathcal{A})$*, where* $X_i = \{a_1, \ldots, a_i\}$ *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 an answer set while* Q *has obviously not. Note that this is caught by our Theorem 1, item (*ii*): for* $(X_0, \mathcal{A})$*, which is an SE-model of* Q *but not of* P*, we cannot find an SE-model* $(X, \mathcal{A})$ *of* P *between* $(X_0, \mathcal{A})$ *and* $(\mathcal{A}, \mathcal{A})$*.*

In fact, uniform equivalence of infinite programs P and Q cannot be captured by a selection of SE-models:

Theorem 4 *Let* P *and* Q *be infinite DLPs. There is no selection of SE-models,* $\sigma(SE(\cdot))$*, such that* P *and* Q *are uniformly equivalent,* $P \equiv_u Q$*, if and only if* $\sigma(SE(P)) = \sigma(SE(Q))$*.*

Proof. Consider programs over $\mathcal{A} = \{a_i \mid i \geq 1\}$ as follows. The program $P = \{a_i \leftarrow \mid i \geq 1\}$ in Example 10, as well as

$$\begin{aligned} Q &= \{a_i \leftarrow not\, a_i,\ a_i \leftarrow a_{i+1},\ a_{2i} \leftarrow a_{2i-1} \mid i \geq 1\}, \\ R &= \{a_i \leftarrow not\, a_i,\ a_i \leftarrow a_{i+1},\ a_{2i+1} \leftarrow a_{2i},\ a_1 \leftarrow \mid i \geq 1\}, \text{ and} \\ S &= \{a_i \leftarrow,\ \leftarrow a_1 \mid i \geq 1\}. \end{aligned}$$

Considering corresponding SE-models, it is easily verified that $SE(P) = \{(\mathcal{A}, \mathcal{A})\}$, $SE(S) = \emptyset$, as well as

$$\begin{aligned} SE(Q) &= \{(\emptyset, \mathcal{A}), (a_1 a_2, \mathcal{A}), \ldots, (a_1 a_2 \cdots a_{2i}, \mathcal{A}), \ldots, (\mathcal{A}, \mathcal{A}) \mid i \geq 0\}, \text{ and} \\ SE(R) &= \{(a_1, \mathcal{A}), (a_1 a_2 a_3, \mathcal{A}), \ldots, (a_1 a_2 \cdots a_{2i+1}, \mathcal{A}), \ldots, (\mathcal{A}, \mathcal{A}) \mid i \geq 0\}. \end{aligned}$$

Hence, we have that $SE(Q) \cap SE(R) = \{(\mathcal{A}, \mathcal{A})\}$. Observe also that $Q \cup X$ and $R \cup X$ do not have an answer set for any proper subset $X \subset \mathcal{A}$, while $\mathcal{A}$ is (the only) answer set of both $Q \cup \mathcal{A}$ and $R \cup \mathcal{A}$. Thus, $Q \equiv_u R$. However, $S \cup \mathcal{A}$ does not have an answer set and we get $Q \not\equiv_u S$ and $R \not\equiv_u S$. Since P has the answer set $\mathcal{A}$, we finally conclude that $P \not\equiv_u Q$, $P \not\equiv_u R$, and $P \not\equiv_u S$.

Towards a contradiction, let us assume that there exists a selection function $\sigma(SE(\cdot))$, such that $P_i \equiv_u P_j$ iff $\sigma(SE(P_i)) = \sigma(SE(P_j))$, for $P_i, P_j \in \{P, Q, R, S\}$. Then, $\sigma(SE(S)) = \emptyset$ and, since $P \not\equiv_u S$, $\sigma(SE(P)) = \{(\mathcal{A}, \mathcal{A})\}$. Furthermore, $Q \equiv_u R$ implies $\sigma(SE(Q)) = \sigma(SE(R))$ and by $SE(Q) \cap SE(R) = \{(\mathcal{A}, \mathcal{A})\}$ we conclude either $\sigma(SE(Q)) = \sigma(SE(R)) = \emptyset$, or $\sigma(SE(Q)) = \sigma(SE(R)) = \{(\mathcal{A}, \mathcal{A})\}$. From $P \not\equiv_u Q$, the former follows, i.e., $\sigma(SE(Q)) = \sigma(SE(R)) = \emptyset$. However, then $\sigma(SE(Q)) = \sigma(SE(S))$ while $Q \not\equiv_u S$, which is a contradiction. □

3.3 Consequence under Uniform Equivalence

Based on UE-models, we define an associated notion of consequence under *uniform equivalence*.

Definition 6 (UE-consequence) *A rule, r, is an* UE-consequence *of a program P, denoted $P \models_u r$, if $(X, Y) \in SE(r)$, 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 the UE-models of a program remain invariant under addition of UE-consequences.

Proposition 4 *For any program P and rule r, if $P \models_u r$ then $UE(P) = UE(P \cup \{r\})$.*

Proof. Let $P \models_u r$, we show that $UE(P) = UE(P \cup \{r\})$.
"$\subseteq$": 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 $(X', Y) \models (P \cup \{r\})^Y$. 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 $(X, Y) \notin UE(P)$. Then, some $(X', Y) \in UE(P)$ exists such that $X \subset X' \subset Y$. By hypothesis, $(X', Y) \in SE(r)$ (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 $(X, Y) \in UE(P)$. □

As usual, we write $P \models_u R$ for any set of rules R if $P \models_u r$ for all $r \in R$. As a corollary, taking Theorem 3 (b) into account, we get the following.

Corollary 2 *For any finite program P and set of rules R, if $P \models_u R$ then $P \cup R \equiv_u P$.*

From this proposition, we also obtain an alternative characterization of uniform equivalence in terms of UE-consequence.

Theorem 5 *Let P and Q be DLPs. Then,*

(a) *$P \equiv_u Q$ implies $P \models_u Q$ and $Q \models_u P$;*

(b) *$P \models_u Q$ and $Q \models_u P$ implies $P \equiv_u Q$, whenever at least one of the programs P, Q is finite.*

Proof. In Case (a), we have $UE(P) = UE(Q)$ if $P \equiv_u Q$ by Theorem 3 (a), 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 $P \models_u Q$. For (b), we apply Proposition 4 repeatedly and obtain $UE(P) = UE(P \cup Q) = UE(Q)$. By Theorem 3 (b) $P \equiv_u Q$. □

Rewriting this result in terms of SE- and UE-models gives the following characterization (which has also been derived for finite programs in [19]; Proposition 5).

Proposition 5 *Let P and Q be DLPs. Then,*

(a) *$P \equiv_u Q$ implies $UE(P) \subseteq SE(Q)$ and $UE(Q) \subseteq SE(P)$;*

(b) *$UE(P) \subseteq SE(Q)$ and $UE(Q) \subseteq SE(P)$ implies $P \equiv_u Q$, whenever at least one of the programs P, Q is finite.*

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 built on UE-consequence, P resp. P^* may be transformed into smaller uniformally equivalent programs; we leave this for further study.

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

Proposition 6 *For any finite program P and rule r, (i) $P \models_u r$ implies $P \cup F \models_c r$, for each set of facts F; (ii) $P \cup F \models_c r$, for each set of facts F, implies $P \models_c r$; and (iii) $P \models_c r$ implies $P \models r$.*

Proof. Since each answer set is a classical model, it remains to show (i). Suppose $P \models_u r$. Then, $P \equiv_u P \cup \{r\}$ by Corollary 2, i.e., $\mathcal{AS}(P \cup F) = \mathcal{AS}(P \cup \{r\} \cup F)$, for each set of facts F. Since $X \models r$ for each $X \in \mathcal{AS}(P \cup \{r\} \cup F)$, it follows that $P \cup F \models_c r$, for each set of facts F. □

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 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 $X \subseteq Y$, we have $r^Y = r^X$. In both cases $X \models r^Y$ follows, which proves $(X, Y) \in SE(r)$. Thus, $P \models_u r$. □

Theorem 6 *Let P be any 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.
$(ii) \Rightarrow (i)$. Suppose (ii) holds. The only-if direction in (i) holds immediatly by Lemma 2. The if direction in (i) holds in gerenal, since $P \models_u r$ iff $UE(P) \subseteq SE(r)$. The latter clearly implies that each total SE-model of P is a total SE-model of r. Consequently, $P \models r$.
$(i) \Rightarrow (ii)$. 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, 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 Section 5.1, though.

4 Relativized Notions of Strong and Uniform Equivalence

In what follows, we formally introduce the notions of relativized strong equivalence (RSE) and relativized uniform equivalence (RUE).

Definition 7 *Let P and Q be programs and let A be a set of atoms. Then,*

(i) *P and Q are* strongly equivalent relative to A*, denoted $P \equiv_s^A Q$, iff $P \cup R \equiv Q \cup R$, for all programs R over A;*

(ii) *P and Q are* uniformly equivalent relative to A*, denoted $P \equiv_u^A Q$, iff $P \cup F \equiv Q \cup F$, for all $($non-disjunctive$)$ facts $F \subseteq A$.*

Observe that the range of applicability of these notions covers ordinary equivalence (by setting $A = \emptyset$) of two programs P, Q, and *general* strong (resp. uniform) equivalence (whenever $Atm(P \cup Q) \subseteq A$). Also the following relation holds: For any set A of atoms, let $A' = A \cap Atm(P \cup Q)$. Then, $P \equiv_e^A Q$ holds, iff $P \equiv_e^{A'} Q$ holds, for $e \in \{s, u\}$.

Our first main result lists some properties for relativized strong equivalence. Among them, we show that RSE shares an important property with general strong equivalence: In particular, from the proofs of the results in [37, 57], it appears that for strong equivalence, only the addition of unary rules is crucial. That is, by constraining the rules in the set R in Definition 7 to unary ones does not lead to a different concept.

Lemma 3 *For programs P, Q, and a set of atoms A, the following statements are equivalent:*

(1) *there exists a program R over A, such that $\mathcal{AS}(P \cup R) \not\subseteq \mathcal{AS}(Q \cup R)$;*

(2) *there exists a unary program U over A, such that $\mathcal{AS}(P \cup U) \not\subseteq \mathcal{AS}(Q \cup U)$;*

(3) *there exists an interpretation Y, such that (a) $Y \models P$; (b) for each $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$, $Y' \not\models P^Y$ holds; and (c) $Y \models Q$ implies existence of an $X \subset Y$, such that $X \models Q^Y$ and, for each $X' \subset Y$ with $(X' \cap A) = (X \cap A)$, $X' \not\models P^Y$ holds.*

Proof. (1) $\Rightarrow$ (3): Suppose an interpretation Y and a set R of rules over A, such that $Y \in \mathcal{AS}(P \cup R)$ and $Y \notin \mathcal{AS}(Q \cup R)$. From $Y \in \mathcal{AS}(P \cup R)$, we get $Y \models P \cup R$ and, for each $Z \subset Y$, $Z \not\models P^Y \cup R^Y$. Thus (a) holds, and since $Y' \models R^Y$ holds, for each Y' with $(Y' \cap A) = (Y \cap A)$, (b) holds as well. From $Y \notin \mathcal{AS}(Q \cup R)$, we get that either $Y \not\models Q \cup R$ or there exists an interpretation $X \subset Y$, such that $X \models Q^Y \cup R^Y$. Note that $Y \not\models Q \cup R$ implies $Y \not\models Q$, since from above, we have $Y \models R$. Thus, in the case of $Y \not\models Q \cup R$, (c) holds; otherwise we get that $X \models Q^Y$. Now since $X \models R^Y$, we know that, for each $X' \subset Y$ with $(X' \cap A) = (X \cap A)$, $X' \not\models P^Y$ has to hold, otherwise $Y \notin \mathcal{AS}(P \cup R)$. Hence, (c) is satisfied.

(3) $\Rightarrow$ (2): Suppose an interpretation Y, such that Conditions (a–c) hold. We have two cases: First, if $Y \not\models Q$, consider the unary program $U = (Y \cap A)$. By Conditions (a) and (b), it is easily seen that $Y \in \mathcal{AS}(P \cup U)$, and from $Y \not\models Q$, $Y \notin \mathcal{AS}(Q \cup U)$ follows. So suppose, $Y \models Q$. By (c), there exists an $X \subset Y$, such that $X \models Q^Y$. Consider the program $U = (X \cap A) \cup \{p \leftarrow q \mid p, q \in (Y \setminus X) \cap A\}$. Again, U is unary over A. Clearly, $Y \models Q \cup U$ and $X \models Q^Y \cup U$. Thus $Y \notin \mathcal{AS}(Q \cup U)$. It remains to show that $Y \in \mathcal{AS}(P \cup U)$. We have $Y \models P \cup U$. Towards a contradiction, suppose a $Z \subset Y$, such that $Z \models P^Y \cup U$. By definition of U, $Z \supseteq (X \cap A)$. If $(Z \cap A) = (X \cap A)$, Condition (c) is violated; if

$(Z \cap A) = (Y \cap A)$, Condition (b) is violated. Thus, $(X \cap A) \subset (Z \cap A) \subset (Y \cap A)$. But then, $Z \not\models U$, since there exists at least one rule $p \leftarrow q$ in U, such that $q \in Z$ and $p \notin Z$. Contradiction.

(2) $\Rightarrow$ (1) is obvious. □

The next result is an immediate consequence of the fact that Propositions (1) and (2) from above result are equivalent.

Corollary 3 *For programs P, Q, and a set of atoms A, $P \equiv_s^A Q$ holds iff, for each unary program U over A, $P \cup U \equiv Q \cup U$ holds.*

We emphasize that therefore also for relatived equivalences, it holds that restricting the syntax of the added rules, RSE and RUE are the only concepts which differ. Note that this generalizes an observation reported in [49] to relativized notions of equivalence, namely that uniform and strong equivalence are the only forms of equivalence obtained by varying the logical form of expressions in the extension.

4.1 A Characterization for Relativized Strong Equivalence

In this section, we provide a semantical characterization of RSE by generalizing the notion of SE-models. Hence, our aim is to capture the problem $P \equiv_s^A Q$ in model-like terms. We emphasize that the forthcoming results are also applicable to infinite programs. Moreover, having found a suitable notion of *relativized SE-models*, we expect that a corresponding pendant for relativized uniform equivalence can be derived in the same manner as general UE-models are defined over general SE-models. As in the case of UE-models, we need some restrictions concerning the infinite case, i.e., if infinite programs are considered.

We introduce the following notion.

Definition 8 *Let A be a set of atoms. A pair of interpretations (X, Y) is a* (relativized) A-SE-interpretation *iff either $X = Y$ or $X \subset (Y \cap A)$. The former are called total and the latter non-total A-SE-interpretations.*

Moreover, an A-SE-interpretation (X, Y) is a (relativized) A-SE-model *of a program P iff*

(i) $Y \models P$;

(ii) for all $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$, $Y' \not\models P^Y$; and

(iii) $X \subset Y$ implies existence of a $X' \subseteq Y$ with $(X' \cap A) = X$, such that $X' \models P^Y$ holds.

The set of A-SE-models of P is given by $SE^A(P)$.

Compared to SE-models, this definition is more involved. This is due to the fact, that we have to take care of two different effects when relativizing strong equivalence. The first one is as follows: Suppose a program P has among its SE-models the pairs (Y, Y) and (Y', Y) with $(Y' \cap A) = (Y \cap A)$ and $Y' \subset Y$. Then, Y never becomes an answer set of a program $P \cup R$, regardless of the rules R over A we add to P. This is due to the fact that either $Y' \models (P \cup R)^Y$ still holds for some $Y' \subset Y$, or, $Y \not\models (P \cup R)^Y$ (the latter is a consequence of finding an R such that $Y' \not\models (P \cup R)^Y$, for $(Y' \cap A) = (Y \cap A)$, $Y' \subset Y$ modelling P). In other words, for the construction of a program R over A, such that $\mathcal{AS}(P \cup R) \neq \mathcal{AS}(Q \cup R)$, it is not worth to to pay attention to any original SE-model of P of the form $(\cdot, Y)$, whenever there exists a $(Y', Y) \in SE(P)$ with $(Y' \cap A) = (Y \cap A)$. This motivates Condition (ii). Condition (iii) deals with a different effect: Suppose P has SE-models (X, Y) and (X', Y), with $(X \cap A) = (X' \cap A) \subset (Y \cap A)$.

Here, it is not possible to eliminate just one of these two SE-models by adding rules over A. Such SE-models which do not differ with respect to A, are collected into a single A-SE-model $((X \cap A), Y)$.

The different role of these two independent conditions becomes even more apparent in the following cases. On the one hand, setting $A = \emptyset$, the A-SE-models of a program P collapse with the answer sets of P. More precisely, all such $\emptyset$-SE-models have to be of the form (Y, Y), and it holds that (Y, Y) is an $\emptyset$-SE-model of a DLP P iff Y is an answer set of P. This is easily seen by the fact that under $A = \emptyset$, Conditions (i) and (ii) in Definition 8 exactly coincide with the characterization of answer sets, following Proposition 2. Therefore, A-SE-model-checking for DLPs is not possible in polynomial time in the general case; otherwise we get that checking whether a DLP has some answer set is NP-complete; which is in contradiction to known results [21], provided the polynomial hierarchy does not collapse. On the other hand, if each atom from P is contained in A, then the A-SE-models of P coincide with the SE-models (over A) of P. The conditions in Definition 8 are hereby instantiated as follows: A pair (X, Y) is an A-SE-interpretation iff $X \subseteq Y$, and by (i) we get $Y \models P$, (ii) is trivially satisfied, and (iii) states $X \models P^Y$.

The central result is as follows. In particular, we show that A-SE-models capture the notion of $\equiv_s^A$ in the same manner as SE-models capture $\equiv_s$.

Theorem 7 *For programs P, Q, and a set of atoms A, $P \equiv_s^A Q$ holds iff $SE^A(P) = SE^A(Q)$.*

Proof. First suppose $P \not\equiv_s^A Q$ and wlog consider for some R over A, $\mathcal{AS}(P \cup R) \not\subseteq \mathcal{AS}(Q \cup R)$. By Lemma 3, there exists an interpretation Y, such that (a) $Y \models P$; (b) for each $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$, $Y' \not\models P^Y$; and (c) $Y \not\models Q$ or there exists an interpretation $X \subset Y$, such that $X \models Q^Y$ and, for each $X' \subset Y$ with $(X' \cap A) = (X \cap A)$, $X' \not\models P^Y$. First suppose $Y \not\models Q$, or $Y \models Q$ and $(X \cap A) = (Y \cap A)$. Then (Y, Y) is an A-SE-model of P but not of Q. Otherwise, i.e., $Y \models Q$ and $(X \cap A) \subset (Y \cap A)$, $((X \cap A), Y)$ is an A-SE-model of Q. But, by Condition (c), $((X \cap A), Y)$ is not an A-SE-model of P.

For the converse direction of the theorem, suppose a pair (Z, Y), such that wlog (Z, Y) is an A-SE-model of P but not of Q. First, let $Z = Y$. We show that $\mathcal{AS}(P \cup R) \not\subseteq \mathcal{AS}(Q \cup R)$ for some program R over A. Since (Y, Y) is an A-SE-model of P, we get from Definition 8, that $Y \models P$ and, for each $Y' \subset Y$ with $(Y \cap A) = (Y' \cap A)$, $Y' \not\models P^Y$. Thus, Conditions (a) and (b) in Part (3) of Lemma 3 are satisfied for P by Y. On the other hand, (Y, Y) is not an A-SE-model of Q. By Definition 8, either $Y \not\models Q$, or there exists a $Y' \subset Y$, with $(Y' \cap A) = (Y \cap A)$, such that $Y' \models Q^Y$. Therefore, Condition (c) from Lemma 3 is satisfied by either $Y \not\models Q$ or, if $Y \models Q$, by setting $X = Y'$. We apply Lemma 3 and get the desired result. Consequently, $P \not\equiv_s^A Q$. So suppose, $Z \neq Y$. We show that then $\mathcal{AS}(Q \cup R) \not\subseteq \mathcal{AS}(P \cup R)$ holds, for some program R over A. First, observe that whenever (Z, Y) is an A-SE-model of P, then also (Y, Y) is an A-SE-model of P. Hence, the case where (Y, Y) is not an A-SE-model of Q is already shown. So, suppose (Y, Y) is an A-SE-model of Q. We have $Y \models Q$ and, for each $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$, $Y' \not\models Q^Y$. This satisfies Conditions (a) and (b) in Lemma 3 for Q. However, since (Z, Y) is not an A-SE-model of Q, for each $X' \subset Y$ with $(X' \cap A) = Z$, $X' \not\models Q^Y$ holds. Since (Z, Y) in turn is an A-SE-model of P, there exists an $X \subset Y$ with $(X \cap A) = Z$, such that $X \models P^Y$. These observations imply that (c) holds in Lemma 3. We apply the lemma and finally get $P \not\equiv_s^A Q$. □

Altough A-SE-models are more involved than SE-models, they share some fundamental properties with general SE-models. On the other hand, some properties do not generalize to A-SE-models. We shall discuss these issues in detail in Section 4.3. For the moment, we list some observations, concerning the relation between SE-models and A-SE-models, in order to present some examples.

A	A-SE-models of Q	A-SE-models of Q'
$\{a,b,c\}$	$(abc,abc),(a,abc),(b,abc)$	$(abc,abc),(a,abc),(b,abc),(\emptyset,abc)$
$\{a,b\}$	$(abc,abc),(a,abc),(b,abc)$	$(abc,abc),(a,abc),(b,abc),(\emptyset,abc)$
$\{a,c\}$	$(abc,abc),(a,abc),(\emptyset,abc)$	$(abc,abc),(a,abc),(\emptyset,abc)$
$\{b,c\}$	$(abc,abc),(\emptyset,abc),(b,abc)$	$(abc,abc),(b,abc),(\emptyset,abc)$
$\{a\}$	-	-
$\{b\}$	-	-
$\{c\}$	(abc,abc), $(\emptyset,abc)$	(abc,abc), $(\emptyset,abc)$
$\emptyset$	-	-

Table 1: Comparing the A-SE-models for Example Programs Q and Q'.

Lemma 4 *Let P be a program and A be a set of atoms. We have the following relations between A-SE-models and SE-models.*

(i) *If $(Y,Y) \in SE^A(P)$, then $(Y,Y) \in SE(P)$.*

(ii) *If $(X,Y) \in SE^A(P)$, then $(X',Y) \in SE(P)$, for some $X' \subseteq Y$ with $(X' \cap A) = X$.*

Example 11 *Consider the programs*

$$Q = \{a \vee b \leftarrow;\ a \leftarrow c;\ b \leftarrow c;\ \leftarrow not\, c;\ c \leftarrow a, b\};$$
$$Q' = \{a \leftarrow not\, b;\ b \leftarrow not\, a;\ a \leftarrow c;\ b \leftarrow c;\ \leftarrow not\, c;\ c \leftarrow a, b\}.$$

Thus, Q' results from Q by replacing the disjunctive rule $a \vee b \leftarrow$ by the two rules $a \leftarrow not\, b$; $b \leftarrow not\, a$.

Table 1 lists, for each $A \subseteq \{a,b,c\}$, the A-SE-models of Q and Q', respectively. The first row of the table gives the SE-models (over $\{a,b,c\}$) for Q and Q'. From this row, we can by Definition 8 and Lemma 4, obtain the other rows quite easily. Observe that we have $Q \not\equiv_s Q'$. The second row shows that, for $A = \{a,b\}$, $Q \not\equiv_s^A Q'$, as well. Indeed, adding $R = \{a \leftarrow b;\ b \leftarrow a\}$ yields $\{a,b,c\}$ as answer set of $Q \cup R$, whereas $Q' \cup R$ has no answer set. For all other $A \subset \{a,b,c\}$, the A-SE-models of Q and Q' coincide. Basically, there are two different reasons. First, for $A = \{a,c\}$, $A = \{b,c\}$, or $A = \{c\}$, Condition (iii) from Definition 8 comes into play. In those cases, at least one of the SE-interpretations (a,abc) or (b,abc) is "switched" to $(\emptyset,abc)$, and thus the original difference between the SE-models disappears when considering A-SE-models. In the remaining cases, i.e., $A \subset \{a,b\}$, Condition (ii) prevents any $(\cdot,abc)$ to be an A-SE-model of Q or Q'. Then, neither Q nor Q' possesses any A-SE-model.

4.2 A Characterization for Relativized Uniform Equivalence

In what follows, we consider the problem of checking relativized uniform equivalence. Therefore, we shall make use of the newly introduced A-SE-models in the same manner as Section 3 provided characterizations for uniform equivalence using SE-models.[2]

We start with a generalization of Lemma 1. The proof is similar to the proof of Lemma 1 and thus relegated to the Appendix.

[2]For a slightly different way to prove the main results on RUE, we refer to [58].

Lemma 5 *Two DLPs P and Q are uniformly equivalent wrt to a set of atoms A, i.e. $P \equiv_u^A Q$, iff for every A-SE-model (X, Y), such that (X, Y) is an A-SE-model of exactly one of the programs P and Q, it holds that (i) $(Y, Y) \in SE^A(P) \cap SE^A(Q)$, and (ii) there exists an A-SE-model (X', Y), $X \subset X' \subset Y$, of the other program.*

From Lemma 5 we immediately obtain the following characterization of relativized uniform equivalence.

Theorem 8 *Two programs, P and Q are uniformly equivalent wrt to a set of atoms A, $P \equiv_u^A Q$, iff*

(i) for each Y, $(Y, Y) \in SE^A(P)$ iff $(Y, Y) \in SE^A(Q)$, i.e., the total A-SE-models of P and Q coincide;

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

In contrast to uniform equivalence, we can obtain further characterizations for $\equiv_u^A$ also for infinite programs, provided that A is finite.

Theorem 9 *Let P and Q be programs, A a set of atoms, such that P, Q, or A is finite. Then $P \equiv_u^A Q$, iff the following conditions hold:*

(i) for each Y, $(Y, Y) \in SE^A(P)$ iff $(Y, Y) \in SE^A(Q)$, i.e., the total A-SE-models of P and Q coincide;

(ii) for each $(X, Y) \in SE^A(P) \cup SE^A(Q)$ such that $X \subset Y$, there exists an $(X', Y) \in SE^A(P) \cap SE^A(Q)$ such that $X \subseteq X' \subset Y$.

The result is proved by the same argumentation as used in the proof of Theorem 2. The only additional argumentation is needed for the cases that P and Q are both infinite, but A is finite. Recall that in this case there is also only a finite number of non-total A-SE-interpretations (X, Y) for fixed Y, since $X \subseteq A$ holds by definition of A-SE-interpretation. Therefore, any chain (as used in the proof of Theorem 2) of different A-SE-models (X, Y) with fixed Y is finite.

As mentioned before, we aim at defining relativized A-UE-models over A-SE-models in the same manner as general UE-models are defined over general SE-models, following Definition 5.

Definition 9 *Let A be a set of atoms and P be a program. A pair (X, Y) is a* (relativized) A-UE-model *of P iff it is an A-SE-model of P and, for every A-SE-model (X', Y) of P, $X \subset X'$ implies $X' = Y$. The set of A-UE-models of P is given by $UE^A(P)$.*

An alternative characterization of A-UE-models, which will be useful later, is immediately obtained from Definitions 8 and 9 as follows.

Proposition 7 *An A-SE-interpretation (X, Y) is an A-UE-model of a program P iff*

(i) $Y \models P$;

(ii) for each $X'' \subset Y$ with either $(X \cap A) \subset (X'' \cap A)$ or $(X'' \cap A) = (Y \cap A)$, $X'' \not\models P^Y$; and

(iii) if $X \subset Y$, there exists a $X' \subseteq Y$ with $(X' \cap A) = (X \cap A)$, such that $X' \models P^Y$.

Next, we derive the desired characterization for relativized uniform equivalence, generalizing the results in Theorem 3.

Theorem 10 *Let P and Q be DLPs, and A a set of atoms. Then,*

(a) $P \equiv_u^A Q$ *implies* $UE^A(P) = UE^A(Q)$;

(b) $UE^A(P) = UE^A(Q)$ *implies* $P \equiv_u^A Q$*, whenever at least one of* P*,* Q*, or* A *is finite.*

Proof. Proving (a) is basically done as for Theorem 3, applying Theorem 8 instead of Theorem 1.

We proceed with the more interesting part (b). First assume that P or A is finite. The case where Q (or A) is finite is analogous. Assume $UE^A(P) = UE^A(Q)$. Then Property (i) of Theorem 8 holds, and towards a contradiction, suppose that Theorem 8 (ii) is not satisfied, i.e., there exists $X \subset Y$, such that either (1) $(X,Y) \in SE^A(P)$ and not exists $X \subseteq X' \subset Y$, $(X',Y) \in SE^A(Q)$, or vice versa (2) $(X,Y) \in SE^A(Q)$ and not exists $X \subseteq X' \subset Y$, $(X',Y) \in SE^A(P)$.
Case (1): We show the existence of a set Z, $X \subseteq Z \subset Y$, such that $(Z,Y) \in UE^A(P)$. If $(X,Y) \in UE^A(P)$, or either Y or A is finite, this is trivial. So let $(X,Y) \notin UE^A(P)$ and both Y and A be infinite. Then $Y_P = Y \cap Atm(P)$ and $X_P = X \cap Atm(P)$ are finite, and $(X_P, Y_P) \in SE^A(P)$. The latter holds by the observations that (i) $Y \models P$ implies $Y_P \models P$; (ii) for each $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$, $Y' \not\models P^Y$ implies that, for each $Y'' \subset Y_P$ with $(Y'' \cap A) = (Y_P \cap A)$, $Y'' \not\models P^Y$; and (iii) $X' \models P^Y$ for some $(X' \cap A) = X$ implies that $(X' \cap Atm(P)) \models P^Y = P^{Y_P}$. Moreover, $X_P \subset Y_P$, otherwise we end up in a contradiction by the fact that then $(X'', X'') \in UE^A(P) = UE^A(Q)$ for some $(X'' \cap A) = X_P$, implying $(X,Y) \in SE^A(Q)$, since $(Y,Y) \in UE^A(Q) = UE^A(P)$ holds. Since Y_P is finite, there exists a set Z_P, $X_P \subseteq Z_P \subset Y_P$, such that $(Z_P, Y_P) \in UE^A(P)$. Now, let $Z = A \cap (Z_P \cup (Y \setminus Y_P))$. Then $X \subseteq Z \subset Y$ holds by construction. Furthermore $(Z,Y) \in UE^A(P)$, since $Y \setminus Z = Y_P \setminus Z_P$, $P^Y = P^{Y_P}$, and $(Z_P, Y_P) \in UE^A(P)$. By our assumption $(Z,Y) \in UE^A(Q)$ follows. Contradiction.
Case (2): We show the existence of a set Z, $X \subseteq Z \subset Y$, such that $(Z,Y) \in UE^A(Q)$. If $(X,Y) \in UE^A(Q)$, or one of A, Y is finite, this is trivial. So let $(X,Y) \notin UE(Q)$, and both Y and A infinite. If $(X,Y) \notin UE^A(Q)$, $((Y \cap A) \setminus X) \subseteq Atm(P)$ must hold; otherwise we end up in a contradiction by taking any atom $a \in (Y \cap A) \setminus X$. (Consider $Z = (Y \cap A) \setminus \{a\}$. Then $X \subseteq Z \subset Y$ holds by construction and since $(Y,Y) \in UE^A(P) = UE^A(Q)$, as well as some Z' with $(Z' \cap A) = Z$ models $P^{Z'} = P^Y$ we get $(Z,Y) \in SE^A(P)$, a contradiction). Now, since $Atm(P)$ is finite, this means that $(Y \cap A) \setminus X$ is finite, i.e., there cannot exist an infinite chain of SE-models $(X,Y) = (X_0, Y), (X_1, Y), \ldots, (X_i, Y), \ldots$, such that $X_i \subset X_j \subset (Y \cap A)$, for $i < j$, and $(X_i, Y) \in SE^A(Q)$. Thus, there exists a maximal model $(Z,Y) \in UE^A(Q)$. By our assumption $(Z,Y) \in UE^A(P)$ follows. Contradiction. Thus, Theorem 8 (ii) holds as well, proving $P \equiv_u^A Q$ in Case (b). □

Example 12 *Recall our example programs Q and Q' from above. Via the first row in the table (i.e., for $A = \{a,b,c\}$, yielding the respective SE-models), it is easily checked by Proposition 3 that Q and Q' are uniformly equivalent. In fact, the SE-model $(\emptyset, abc)$ of Q' is not a UE-model of Q', due to the presence of the SE-model (a, abc), or alternatively because of (b, abc). Note that $Q \equiv_u Q'$ implies $Q \equiv_u^A Q'$ for any A. Inspecting the remaining rows in the table, it can be seen that for any A, the sets of A-UE-models of Q and Q' are equal, as expected.*

We conclude this section, with remarking that we do not have a directly corresponding result to Theorem 5 for relativized uniform equivalence (see also next subsection). A generalization of Proposition 5 is possible, however. The proof is in the Appendix.

Theorem 11 *Let P and Q be DLPs, and A a set of atoms. Then,*

(a) *$P \equiv_u^A Q$ implies $UE^A(P) \subseteq SE^A(Q)$ and $UE^A(Q) \subseteq SE^A(P)$;*

(b) *$UE^A(P) \subseteq SE^A(Q)$ and $UE^A(Q) \subseteq SE^A(P)$ implies $P \equiv_u^A Q$, whenever at least one of P, Q, or A is finite.*

4.3 Properties of Relativized Equivalences

This section collects a number of properties of A-SE-models and A-UE-models, respectively. Note that there are situations where A-SE-models and A-UE-models are the same concepts.

Proposition 8 *For any program P, and a set of atoms A with $card(A) < 2$, $SE^A(P) = UE^A(P)$ holds.*

Corollary 4 *For programs P,Q and a set of atoms A with $card(A) < 2$, $P \equiv_s^A Q$ iff $P \equiv_u^A Q$.*

The following results are only given in terms of A-SE-models; the impact of the results on properties of A-UE-models is in most cases obvious, and thus not explicitly mentioned.

First, we are able to generalize Proposition 2 to relativized SE-models.

Lemma 6 *An interpretation Y is an answer set of a program P iff $(Y, Y) \in SE^A(P)$ and, for each $X \subset Y$, $(X, Y) \notin SE^A(P)$.*

One drawback of A-SE-models is that they are not closed under program composition. Formally, $SE^A(P \cup Q) = SE^A(P) \cap SE^A(Q)$ does not hold in general; however, it holds whenever A contains all atoms occurring in P or Q. However, the fact that, in general, $SE^A(P \cup Q) \neq SE^A(P) \cap SE^A(Q)$, is not a surprise, since for $A = \emptyset$, A-SE-models capture answer sets; and if this closure property would hold, answer set semantics would be monotonic.

Proposition 9 *For programs P, Q, and a set of atoms A, we have the following relations:*

(i) *$(Y, Y) \in SE^A(P) \cap SE^A(Q)$ implies $(Y, Y) \in SE^A(P \cup Q)$;*

(ii) *for $X \subset Y$, $(X, Y) \in SE^A(P \cup Q)$ implies $(X, Y) \in SE^A(R)$, whenever $(Y, Y) \in SE^A(R)$, for $R \in \{P, Q\}$;*

(iii) *the converse directions of (i) and (ii) do not hold in general.*

Proof. ad (i): Suppose $(Y, Y) \notin SE^A(P \cup Q)$; then either (a) $Y \not\models P \cup Q$; or (b) there exists a $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$, such that $Y' \models (P \cup Q)^Y$. If $Y \not\models P \cup Q$, then either $Y \not\models P$ or $Y \not\models Q$. Consequently, $(Y, Y) \notin SE^A(P)$ or $(Y, Y) \notin SE^A(Q)$. So, suppose $Y \models P \cup Q$ and (b) holds. Then neither, $(Y, Y) \in SE^A(P)$ nor $(Y, Y) \in SE^A(Q)$.

ad (ii): Let $R \in \{P, Q\}$. Suppose $(Y, Y) \in SE^A(R)$ and $(X, Y) \notin SE^A(R)$. The latter implies that no $X' \subset Y$ with $(X' \cap A) = (X \cap A)$, satisfies $X' \models P^Y$. Consequently, no such X' satisfies $X' \models (P \cup Q)^Y$, and thus $(X, Y) \notin SE^A(P \cup Q)$.

ad (iii): Take the following example programs. Consider programs over $V = \{a, b, c\}$ containing rules $R = \{\leftarrow not\, a;\ \leftarrow not\, b;\ \leftarrow not\, c\}$. Note that $SE(R) = \{(X, V) \mid X \subseteq V\}$. Let

$$\begin{aligned} P_a &= R \cup \{a \leftarrow;\ b \leftarrow c;\ c \leftarrow b\}; \\ P_b &= R \cup \{b \leftarrow;\ a \leftarrow c;\ c \leftarrow a\}; \\ P_c &= R \cup \{c \leftarrow;\ a \leftarrow b;\ b \leftarrow a\}. \end{aligned}$$

Then, the SE-models of P_v are given by (v, abc) and (abc, abc), for $v \in V$.

Set now, for instance, $A = \{c\}$. Then, we have $SE^A(P_a) = SE^A(P_b) = \{(\emptyset, abc), (abc, abc)\}$, while $SE^A(P_c) = \emptyset$. However, $SE^A(P_a \cup P_b) = SE^A(P_a \cup P_c) = SE^A(P_b \cup P_c) = \{(abc, abc)\}$. This shows that for both, (i) and (ii) in Proposition 9, the converse direction does not hold. □

The above result crucially influences the behavior of relativized consequence operators, i.e., generalizations of $\models_e$ as introduced in Definitions 3 and 6, respectively, to the relativized notions of equivalence.

To check rule redundancy in the context of relatived strong equivalence, we give the following result.

Definition 10 *A rule, r, is an A-relativized* SE-consequence *of a program P, denoted $P \models_s^A r$, if $(X, Y) \in SE^A(\{r\})$, for all $(X, Y) \in SE^A(P)$.*

Lemma 7 *For any set of atoms A, program P, and rule r with $(B^+(r) \cup H(r)) \subseteq A$, it holds that if $P \models_s^A r$ then $P \cup \{r\} \equiv_s^A P$.*

Proof. We show $SE^A(P \cup \{r\}) = SE^A(P)$, given $P \models_s^A r$.
"$\subseteq$": Let $(X, Y) \in SE^A(P \cup \{r\})$. We show $(X, Y) \in SE^A(P)$. First let $X = Y$. Then, $Y \models P \cup \{r\}$ and, for each $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$, $Y' \not\models (P \cup r)^Y$. Since $Atm(r^Y) \subseteq A$, for each such Y', $Y' \models r^Y$, and therefore, $Y' \not\models P^Y$. Consequently, $(Y, Y) \in SE^A(P)$. So suppose, $X \subset Y$. Then, $(Y, Y) \in SE^A(P \cup \{r\})$. We already know that then $(Y, Y) \in SE^A(P)$. We apply Proposition 9, and get $(X, Y) \in SE^A(P)$.
"$\supseteq$": Let $(X, Y) \in SE^A(P)$. Then, $(Y, Y) \in SE^A(P)$ and by assumption $(Y, Y) \in SE^A(\{r\})$. By Proposition 9, we get $(Y, Y) \in SE^A(P \cup \{r\})$. Moreover, from $(X, Y) \in SE^A(P)$ and $P \models_s^A r$, we get $(X, Y) \in SE^A(\{r\})$. Hence, there exist X', X'' with $(X' \cap A) = (X'' \cap A) = (X \cap A)$ such that $X' \models P^Y$ and $X'' \models r^Y$. By assumption $Atm(r^Y) \subseteq A$. Since X' and X'' agree on A, we get $X' \models r^Y$; and thus $X' \models (P \cup r)^Y$. Consequently, $(X, Y) \in SE^A(P \cup \{r\})$. □

The result similarly applies to the notion of UE-consequence relative to A, i.e., the restriction $(H(r) \cup B^+(r)) \subseteq A$ is also necessary in that case. However (as in Proposition 4), the result has to be slightly rephrased for A-UE-models in order to handle the case of infinite programs properly.

In general, checking rule-redundancy with respect to relativized equivalences is a more involved task; we leave it for further study.

5 Restricted Classes of Programs

So far, we discussed several forms of equivalence for propositional programs, in general. This section is devoted to two prominent subclasses of disjunctive logic programs, namely positive and head-cycle free programs. Notice that these classes include the Horn logic programs and the disjunction-free logic programs, respectively.

5.1 Positive Programs

While for programs with negation, strong equivalence and uniform equivalence are different, the notions coincide for positive programs, also in the relativized cases. We start with some technical results.

Lemma 8 *Let P be a program, and A, $X \subset Y$ be sets of atoms. We have the following relations:*

1. *If* $(Y, Y) \in SE^A(P)$ *and* $(X, X) \in SE^A(P)$*, then* $((X \cap A), Y) \in SE^A(P)$*.*
2. *If* $(X, Y) \in SE^A(P)$*, then* $(Y, Y) \in SE^A(P)$ *and, whenever* P *is positive, there exists an* $X' \subseteq Y$ *with* $(X' \cap A) = X$*, such that* $(X', X') \in SE^A(P)$*.*

Proof. (1) First, observe that $(X \cap A) \subset (Y \cap A)$ holds. Otherwise, we get from $X \models P^X$, $X \models P^Y$ (since $P^Y \subseteq P^X$, whenever $X \subseteq Y$), and thus $(Y, Y) \notin SE^A(P)$, by definition. Moreover, since $X \models P^Y$ and $(Y, Y) \in SE^A(P)$, we derive $((X \cap A), Y) \in SE^A(P)$.

(2) Let $(X, Y) \in SE^A(P)$. Then, $(Y, Y) \in SE^A(P)$ is an immediate consequence of the definition of A-SE-models. From $(X, Y) \in SE^A(P)$ we get that there exists an $X' \subseteq Y$ with $(X' \cap A) = X$, such that $X' \models P^Y$. Take X' as the minimal interpretation satisfying this condition. For positive P, we have $P^{X'} = P^Y = P$ and we get $X' \models P^{X'} = P$. Moreover, since we chose X' minimal, there does not exist an $X'' \subset X'$ with $(X'' \cap A) = (X' \cap A)$, such that $X'' \models P^{X'} = P$. Hence, $(X', X') \in SE^A(P)$. □

In other words, the set of all A-SE-models of a positive program P is determined by its total A-SE-models. An important consequence of this result is the following.

Proposition 10 *Let* P*,* Q *be programs,* P *be positive, and suppose the total* A*-SE-models of* P *and* Q *coincide. Then,* $SE^A(P) \subseteq SE^A(Q)$*.*

Proof. Towards a contradiction, assume there exists an A-SE-interpretation satisfying $(X, Y) \in SE^A(P)$ and $(X, Y) \notin SE^A(Q)$. Since P is positive, by Lemma 8 we get that there exists some $X' \subseteq Y$ with $(X' \cap A) = X$, such that $(X', X') \in SE^A(P)$. By assumption, the total A-SE-models coincide, and thus we have $(X', X') \in SE^A(Q)$. Moreover, since $(X, Y) \in SE^A(P)$, we get $(Y, Y) \in SE^A(P)$ and furthermore $(Y, Y) \in SE^A(Q)$. Hence, $(X', X') \in SE^A(Q)$ and $(Y, Y) \in SE^A(Q)$. By Lemma 8, we get that $((X' \cap A), Y) = (X, Y)$ is A-SE-model of Q, which is in contradiction to our assumption. □

From this result, we get that deciding relativized strong and uniform equivalence of positive programs collapses to checking whether total A-SE-models coincide.

Theorem 12 *Let* P *and* Q *be positive DLPs, and* A *a set of atoms. The following propositions are equivalent:*

(i) $P \equiv_s^A Q$*;*

(ii) $P \equiv_u^A Q$*;*

(iii) $(Y, Y) \in SE^A(P)$ *iff* $(Y, Y) \in SE^A(Q)$*, for each interpretation* Y*.*

Proof. (i) implies (ii) by definition; (ii) implies (iii) by Theorem 10. We show (iii) implies (i). Applying Proposition 10 in case of two positive programs immediately yields that (iii) implies $SE^A(P) = SE^A(Q)$. Hence, $P \equiv_s^A Q$. □

Therefore, RSE and RUE are the same concepts for positive programs; we thus sometimes write generically $\equiv_e$ for $\equiv_s$ and $\equiv_u$.

An important consequence of this result, is that A-UE-models (and thus UE-models) are capable to deal with infinite programs as well, provided they are positive.

Corollary 5 *Let* A *be a (possibly infinite) set of atoms, and* P*,* Q *(possibly infinite) positive program. Then,* $P \equiv_u^A Q$ *holds iff* $UE^A(P) = UE^A(Q)$*.*

Proof. The only-if direction has already been obtained in Theorem 10. For the if direction, note that $UE^A(P) = UE^A(Q)$ implies (iii) from Theorem 12, and since P and Q are positive we derive $P \equiv_u^A Q$ immediately from that Theorem. □

Concerning strong equivalence and uniform equivalence, Lemma 8 generalizes some well known observations for positive programs.

Proposition 11 *For any positive program P, and sets of atoms $X \subseteq Y$, $(X, Y) \in SE(P)$ iff $(X, X) \in SE(P)$ and $(Y, Y) \in SE(P)$.*

In other words, the set of all SE-models of a program P is determined by its total SE-models (i.e., by the classical models of P). As known and easy to see from main results [37, 56, 57], on the class of positive programs classical and strong equivalence coincide. Using Theorem 12, we can extend this result:

Theorem 13 *For positive programs P, Q, $P \equiv_e Q$ ($e \in \{s, u\}$) iff P and Q have the same classical models.*

Note that Sagiv [52] showed that uniform equivalence of DATALOG programs Π and Π' coincides with equivalence of Π' and Π over Herbrand models; this implies the above result for definite Horn programs. Maher [43] 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 [10].

Example 13 *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 14 *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 Theorem 12).*

Concerning the relativized notions, a result corresponding directly to Theorem 13 is not achievable. However, this is not surprising, otherwise we would have that in case of empty A, $P \equiv_s^A Q$ (or $P \equiv_u^A Q$) collapses to classical equivalence. This, of course, cannot be the case since for positive programs, $P \equiv Q$ denotes the equivalence of the *minimal* classical models of P and Q, rather than classical equivalence.

Thus, while for strong and uniform equivalence total models (Y, Y) for a positive program P coincide with the classical models Y of P, the relativized variants capture a more specific relation, viz. minimal models. We therefore define as follows.

Definition 11 *An A-minimal model of a program P is a classical model Y of P, such that, for each $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$, Y' is not a classical model of P.*

Then, we can generalize Theorem 13 in the following manner:

Theorem 14 *Let P and Q be positive DLPs, and A a set of atoms. Then, $P \equiv_e^A Q$ ($e \in \{s, u\}$) iff P and Q have the same A-minimal models.*

Proof. By Theorem 12 it is sufficient to show that the total A-SE-models of a program P equal its A-minimal models. This relation holds for positive programs, since $P^Y = P$ for any positive program P and any interpretation Y. In this case the conditions for $(Y, Y) \in SE^A(P)$ are the same as for Y being A-minimal for P. □

Note that for $A = \emptyset$ the theorem states that $P \equiv_e^A Q$ iff the minimal classical models of P and Q coincide, reflecting the minimal model semantics of positive programs. On the other hand, for $A = U$, the theorem states that $P \equiv_e^A Q$ iff all classical models of P and Q coincide, as stated above.

5.2 Head-cycle free programs

The class of head-cycle free programs generalizes the class of normal logic programs by permitting a restricted form of disjunction. Still, it is capable of expressing nondeterminism such as, e.g., a guess for the value of an atom a, which does not occur in the head of any other rule. For a definition of head-cycle freeness, we refer to Section 2. As shown by Ben-Eliyahu and Dechter [4], each head-cycle free program can be rewritten to an ordinary equivalent normal program, which is obtained by shifting atoms from the head to the body.

More formally, let us define the following notations.

Definition 12 *For any rule r, let*

$$r^{\rightarrow} = \begin{cases} \{a \leftarrow B^+(r), not\,(B^-(r) \cup (H(r) \setminus \{a\})) \mid a \in H(r)\} & \text{if } H(r) \neq \emptyset, \\ \{r\} & \text{otherwise} \end{cases}$$

For any DLP P, let $P_r^{\rightarrow} = (P \setminus \{r\}) \cup r^{\rightarrow}$; and $P^{\rightarrow} = \bigcup_{r \in P} r^{\rightarrow}$.

It is well-known that for any head-cycle free program P, it holds that $P \equiv P^{\rightarrow}$ (cf. [4]). This result can be strengthened to uniform equivalence as well as to its relativized forms.

Theorem 15 *For any head-cycle free program P, and any set of atoms A, it holds that $P \equiv_u^A P^{\rightarrow}$.*

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

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 $b \leftarrow not\, a$ (the 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 [39, 22].

The following result provides a characterization of arbitrary programs which are relativized strongly equivalent to their shift variant. A more detailed discussion of eliminating disjunction under different notions of equivalences was recently published in [19].

First, we state a simple technical result.

Lemma 9 *For any rule r, $SE(r) \subseteq SE(r^{\rightarrow})$.*

Proof. Indirect. Suppose $(X, Y) \in SE(r)$ and $(X, Y) \notin SE(r^{\rightarrow})$. Then, $Y \models r$ and either $Y \cap B^{-}(r) \neq \emptyset$, $X \not\models B^{+}(r)$, or $X \cap H(r) \neq \emptyset$. By classical logic, $Y \models r$ iff $Y \models r^{\rightarrow}$. By assumption $(X, Y) \notin SE(r^{\rightarrow})$, there exists a rule in $r^{\rightarrow}$ with a as the only atom in its head, such that $a \notin X$, $Y \cap B^{-}(r) = \emptyset$, $X \models B^{+}(r)$, and $Y \cap (H(r) \setminus \{a\}) = \emptyset$. Hence, from the above conditions for $(X, Y) \in SE(r)$, only $X \cap H(r) \neq \emptyset$ applies. Then, some b from $H(r)$ is contained in X. If $a = b$ we get a contradiction to $a \notin X$; otherwise we get a contradiction to $Y \cap (H(r) \setminus \{a\}) = \emptyset$, since $Y \supseteq X$ and thus $b \in Y$. □

Next, we define the following set, which characterizes the exact difference between r and $r^{\rightarrow}$ in terms of SE-models.

Definition 13 *For any rule r, define*

$$S_r = \{(X, Y) \mid X \subseteq Y,\ X \models B^{+}(r),\ Y \cap B^{-}(r) = \emptyset,\ card(H(r) \cap Y) \geq 2,\ H(r) \cap X = \emptyset\}.$$

Proposition 12 *For any disjunctive rule r, $SE(r^{\rightarrow}) \setminus SE(r) = S_r$.*

A proof for this result can be found in [19]. Hence, together with Lemma 9, we get that, for any disjunctive rule r, S_r characterizes exactly the difference between r and $r^{\rightarrow}$ in terms of SE-models.

Theorem 16 *Let P be a program, and $r \in P$. Then, $P \equiv_s^A P_r^{\rightarrow}$ iff for each SE-model $(X, Y) \in SE(P_r^{\rightarrow}) \cap S_r$, exists a $X' \subset Y$, with $X' \neq X$ and $(X' \cap A) = (X \cap A)$, such that $(X', Y) \in SE(P)$.*

Proof. Suppose $P \not\equiv_s^A P_r^{\rightarrow}$. First, assume there exists an A-SE-interpretation $(Z, Y) \in SE^A(P)$ such that $(Z, Y) \notin SE^A(P_r^{\rightarrow})$. By definition of A-SE-models, Lemma 9 and the fact that $Y \models P$ iff $Y \models P_r^{\rightarrow}$, we get that $Z = Y$. Since $(Y, Y) \notin SE^A(P_r^{\rightarrow})$ but $Y \models P_r^{\rightarrow}$, there exists an X such that (X, Y) is SE-model of $P_r^{\rightarrow}$. Moreover, by Proposition 12, $(X, Y) \in S_r$. On the other hand, from $(Y, Y) \in SE^A(P)$, we get that, for each $X' \subset Y$ with $(X' \cap A) = (Y \cap A) = (X \cap A)$, (X', Y) is not SE-model of P. Second, assume there exists an A-SE-interpretation $(Z, Y) \in SE^A(P_r^{\rightarrow})$, such that $(Z, Y) \notin SE^A(P)$. One can verify that using Lemma 9 this implies $Z \subset Y$. Hence, there exists some $X \subseteq Y$ with $(X \cap A) = (Z \cap A)$ such that (X, Y) is SE-model of $P^{\rightarrow}$ but no X' with $(X' \cap A) = (X \cap A)$ is SE-model of P. Moreover, $(X, Y) \in S_r$. This shows the claim. The converse direction is by exactly the same arguments. □

As an immediate consequence of this result, we obtain the following characterization for general strong equivalence.

Corollary 6 *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 $(X, Y) \in S_r$ (i.e., $SE(P^{\rightarrow}) \cap S_r = \emptyset$).*

Example 15 *Reconsider $P = \{a \vee b \leftarrow\}$. Then $P^{\rightarrow} = \{a \leftarrow not\, b, b \leftarrow not\, a\}$ has the SE-model $(\emptyset, ab)$ which satisfies the conditions for $S_{a \vee b \leftarrow}$. 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, $(\emptyset, ab)$ is also an SE-model of $P'^{\rightarrow}$. Furthermore, P' is also not uniformly equivalent to $P'^{\rightarrow}$, since $(\emptyset, ab)$ is moreover a UE-model of $P'^{\rightarrow}$, but P' has the single SE-model (and thus UE-model) (ab, ab).*

We already have seen that shifting is possible if the disjunction is made exclusive with an additional constraint (see also Example 2).

Example 16 *Let P be a program containing the two rules $r = a \vee b \leftarrow$ and $r' = \leftarrow a, b$. The rule r' guarantees that no SE-model (X, Y) of P or of $P_r^{\rightarrow}$ with $\{a, b\} \subseteq Y$ exists. But then, S_r does not contain an element from $SE(P_r^{\rightarrow})$, and we get by Corollary 6, $P \equiv_s P_r^{\rightarrow}$.*

So far, we have presented a general semantic criterion for deciding whether shifting is invariant under $\equiv_s^A$. We close this section, with a syntactic criterion generalizing the concept of head-cycle freeness.

Definition 14 *For a set of atoms A, a rule r is A-head-cycle free (A-HCF) in a program P, iff the dependency graph of P augmented with the clique over A, does not contain a cycle going through two atoms from $H(r)$. A program is A-HCF, iff all its rules are A-HCF.*

In other words, the considered augmented graph of P as used in the definition is given by the pair $(A \cup Atm(P), E)$ with

$$E = \bigcup_{r \in P} \{(p, q) \mid p \in B^+(r), q \in H(r), p \neq q\} \cup \{(p, q),\ (q, p) \mid p, q \in A,\ p \neq q\}$$

and obviously coincides with the (ordinary) dependency graph of the program $P \cup R$, where R is the set of all unary rules over A. Recall that following Corollary 3, unary rules characterize relativized strong equivalence sufficiently. From this observation, the forthcoming results follow in a straight-forward manner.

Theorem 17 *For any program P, $r \in P$, and a set of atoms A, $P \equiv_s^A P_r^{\rightarrow}$, whenever r is A-HCF in P.*

Note that if r is A-HCF in P, then r is HCF in $P \cup R$, where R is the set of unary rules over A. In turn, r then is HCF in all programs $P \cup R'$, with $R' \subseteq R$. Thus, $P \cup R' \equiv P_r^{\rightarrow} \cup R'$ holds for all R' by known results. Consequently, $P \equiv_s^A P_r^{\rightarrow}$.

Corollary 7 *For any program P, and a set of atoms A, $P \equiv_s^A P^{\rightarrow}$ holds, whenever P is A-HCF.*

6 Computational Complexity

In this section, we address the computational complexity of checking various notions of equivalence for logic programs. We start with uniform equivalence also taking the associated consequence operator into account. Then, we generalize these results and consider the complexity of relativized equivalence. Finally, we consider *bounded* relativization, i.e., the problem of deciding $P \equiv_e^A Q$ ($e \in \{s, u\}$), such that the number of atoms *missing* in A is bounded by a constant k, denoted $P\ {}^k{\equiv}_e^A\ Q$. For all three groups of problems we provide a fine-grained picture of their complexity by taking different classes of programs into account.

Recall that $\Pi_2^P = \text{coNP}^{\text{NP}}$ 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 $\Sigma_2^P = \text{NP}^{\text{NP}}$. As well, the class D^P consists of all problems expressible as the conjunction of a problem in NP and a problem in coNP. Moreover, any problem in D^P can be solved with a fixed number of NP-oracle calls, and is thus intuitively easier than a problem complete for Δ_2^P.

Our results are summarized in Table 2. More precisely, the table shows the complexity of the considered problems $P \equiv_s^A Q$ and $P \equiv_u^A Q$ in the general case; as well as in the bounded case ($P\ {}^k{\equiv}_s^A\ Q$ and $P\ {}^k{\equiv}_u^A\ Q$). Moreover, we explicitly list the problem of uniform equivalence, $P \equiv_u Q$. Depending on the program classes P and Q belong to, the corresponding entry shows the complexity (in terms of a completness result) for all five equivalence problems with respect to these classes. In fact, the table has to be read as follows.

$P \equiv^A_s Q$ / $P \equiv^A_u Q$ / $P \,{}^k\!\!\equiv^A_s Q$ / $P \,{}^k\!\!\equiv^A_u Q$ / $P \equiv_u Q$	DLP	positive	HCF	normal	Horn
Horn	Π^P_2 coNP	coNP coNP	coNP coNP	coNP coNP	coNP P
normal	Π^P_2 coNP	Π^P_2 coNP	Π^P_2/coNP coNP	coNP coNP	
HCF	Π^P_2 coNP	Π^P_2 coNP	Π^P_2/coNP coNP		
positive	Π^P_2 coNP/Π^P_2/Π^P_2	Π^P_2 coNP			
DLP	Π^P_2 coNP/Π^P_2/Π^P_2				

Table 2: Complexity of Equivalence Checking in Terms of Completeness Results.

For instance, the complexity of equivalence checking for DLPs in general is given by the entry in the last line and the first column of Table 2. The entry's first line refers to the problems $P \equiv^A_s Q$ and $P \equiv^A_u Q$ (which are both Π^P_2-complete), and the entry's second line refers to the problems $P \,{}^k\!\!\equiv^A_s Q$, $P \,{}^k\!\!\equiv^A_u Q$, and $P \equiv_u Q$, respectively. The latter two show Π^P_2-completenes while $P \,{}^k\!\!\equiv^A_s Q$ is coNP-complete. As another example, the complexity of deciding equivalence of a head-cycle free program and a normal program is reported by the entry in the second line of the third column.

We now highlight the most interesting entries of Table 2.

- (Unrelativized) uniform equivalence is *harder* than (unrelativized) strong equivalence; and this result carries over to the case of bounded relativization. This difference in complexity is only obtained if both programs involved contain head-cycles and at least one of them contains default negation.

- For the case of relativization, uniform equivalence is in some cases *easier* to decide than relativized strong equivalence. This effect occurs only, if both programs are head-cycle free, whereby one of them may be normal (but not Horn).

- Another interesting case amounts if two Horn programs are involved. Hereby, relativized equivalence is harder than in the bounded case, but it is also harder than *ordinary* equivalence (see Theorem 33 in Section 6.2 below). In each other case, relativization is never harder than ordinary equivalence.

- Finally, we list those cases where bounded relativizations decreases the complexity: As already mentioned for both RSE and RUE, this holds for comparing Horn programs. Additionally, in the case of RSE, there is a proper decrease whenever one program is disjunctive and the other is not Horn, or P contains negation as well as head-cycles and Q is Horn. In the latter situation, we also observe a descrease in the case of RUE. Additionaly, such a decrease for RUE is present, if P is normal or HCF and Q is disjunctive and contains headcycles, or if two positive DLPs containing headcycles are compared.

	A-SE-models				A-UE-models		UE-models
	A bounded	general	$card(A) = 1$	$A = \emptyset$	general	A bounded	
DLP/positive	in P	D^P	D^P	coNP	D^P	coNP	coNP
HCF	in P	NP	P	P	P	P	P
normal/Horn	in P	P	P	P	P	P	P

Table 3: Complexity of Model Checking.

Some of the effects can be explained by inspecting the underlying decision problem of model checking. For a set of atoms A, the problem of A-SE-model checking (resp. A-UE-model checking) is defined as follows: Given sets of atoms X, Y, and a program P, decide whether $(X, Y) \in SE^A(P)$ (resp. $(X, Y) \in UE^A(P)$). We compactly summarize our results on A-SE-model checking, resp. A-UE-model checking, in Table 3. This table has to be read as follows. The lines determine the class of programs dealed with and the columns refer to model checking problems in different settings. From left to right we have: (i) bounded A-SE-model checking of a program P, i.e., it is assumed that $Atm(P) \setminus A$ contains a fixed number of atoms; (ii) the general A-SE-model checking problem; (iii) the special case of $card(A) = 1$, where A-SE-model checking and A-UE-model checking coincide; (iv) the special case of $card(A) = 0$, where both A-SE-model checking and A-UE-model checking coincide with answer set checking; (v) the general A-UE-model checking problem; (vi) bounded A-UE-model checking (analogously to bounded A-SE-model checking); and finally, we explicitly list the results for (vii) UE-model checking. All results from Table 3 are proven in detail in the subsequent sections, as well. All entries except the ones in the first column are completeness results. Some interesting observations, which also intuitively explain the different results for $\equiv_s^A$ and $\equiv_u^A$ include: (1) A-SE-model checking is easier than A-UE-model checking in the case of DLPs and bounded A; Roughly spoken, in this case the additional test for maximality in A-UE-model checking is responsible for the higher complexity; (2) for the case of head-cycle free programs, A-SE-model checking is harder than A-UE-model checking, viz. NP-complete. This result is a consequence of Theorem 16, which guarantees that in terms of uniform equivalence, shifted HCF (and thus normal) programs can be employed; recall that this simplification is not possible in the context of strong equivalence.

Towards showing all results in detail, we introduce the following notions used throughout this section. We often reduce propositional formulas to logic programs using, for a set of propositional atoms V, an additional set of atoms $\bar{V} = \{\bar{v} \mid v \in V\}$ within the programs to refer to negative literals. Consequently, we associate to each interpretation $I \subseteq V$, an extended interpretation $\sigma_V(I) = I \cup \{\bar{v} \mid v \in V \setminus I\}$, usually dropping subscript V if clear from the context. The classical models of a formula ϕ are denoted by M_ϕ. Furthermore, we have a mapping $(\cdot)^*$ defined as $v^* = v$, $(\neg v)^* = \bar{v}$, and $(\phi \circ \psi)^* = \phi^* \circ \psi^*$, with v an atom, ϕ and ψ formulas, and $\circ \in \{\vee, \wedge\}$. A further mapping $(\bar{\cdot})$ is defined as $\overline{v} = \bar{v}$, $\overline{\neg v} = v$, $\overline{\phi \vee \psi} = \overline{\phi} \wedge \overline{\psi}$, and $\overline{(\phi \wedge \psi)} = \overline{\phi} \vee \overline{\psi}$. To use these mappings in logic programs, we denote rules also by $a_1 \vee \ldots \vee a_l \leftarrow a_{l+1} \wedge \ldots \wedge a_m \wedge not\, a_{m+1} \wedge \ldots \wedge not\, a_n$.

Finally, we define, for a set of atoms $Y \subseteq U$, the following sets of Horn rules.

$$
\begin{aligned}
Y_{\subseteq}^U &= \{\leftarrow y \mid y \in U \setminus Y\} \\
Y_{\subset}^U &= Y_{\subseteq}^U \cup \{\leftarrow y_1, \ldots, y_n\} \\
Y_{=}^U &= Y_{\subseteq}^U \cup Y
\end{aligned}
$$

Sometimes we do not write the superscript U which refers to the universe. We assume that, unless stated

$P \equiv_u Q$	DLP	positive	HCF	normal	Horn
Horn	coNP	coNP	coNP	coNP	P
normal	coNP	coNP	coNP	coNP	
HCF	coNP	coNP	coNP		
positive	Π_2^P	coNP			
DLP	Π_2^P				

Table 4: Complexity of Uniform Equivalence in Terms of Completeness Results.

otherwise, U refers all the atoms occurring in the programs under consideration.

6.1 Complexity of Uniform Equivalence

In this section, we address the computational complexity of uniform equivalence. While our main interest is with the problem of deciding uniform equivalence of two given programs, we also consider the related problems of UE-model checking and UE-consequence. Our complexity results for deciding uniform equivalence of two given programs are collected from Table 2 into Table 4, for the matter of presentation. The table has to be read as Table 2. Note that in general, uniform equivalence is complete for class Π_2^P, and therefore more complex than deciding strong equivalence, which is in coNP [47, 40, 57]. 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 as strong equivalence.

In what follows, we prove all the results in Table 4. Towards these results, we start with the problem of UE-model checking. Let $\|\alpha\|$ denote the size of an object α.

Theorem 18 *Given a pair of sets (X, Y) and a program P, the problem of 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 $X \models P^Y$, i.e., $(X, Y) \in SE(P)$, for given interpretations X, 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 via checking

$$P^Y \cup X \cup Y_{\subset} \quad \models \quad X_{=}. \tag{1}$$

In fact, each model, X', of $P^Y \cup X \cup Y_{\subset}$ gives a non-total SE-model (X', Y) of P with $X \subseteq X' \subset Y$. On the other hand, the only model of $X_{=}$ is X itself. Hence, (1) holds iff no X' with $X \subset X' \subset Y$ exists such that $(X', Y) \in SE(P)$, i.e., iff $(X, Y) \in UE(P)$. In general, deciding (1) is in coNP witnessed by the membership part of (i).

If P is normal then the involved programs in (1) are Horn and, since classical consequence can be decided in polynomial time for Horn programs, the overall check proceeds in polynomial time. Finally, if P is head-cycle free, then also P^Y is. Moreover, by Theorem 15 we have $P \equiv_u P^{\rightarrow}$. Hence, in this case, (1) holds iff $(P^{\rightarrow})^Y \cup X \cup Y_{\subset} \models X_{=}$. Since $P^{\rightarrow}$ is normal, the latter test can be done in polynomial time (with respect to $\|P\| + \|X\| + \|Y\|$). This shows (ii).

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

$$\begin{array}{rllll} P = & \{ \; x_i \vee \bar{x}_i \leftarrow x_j. & x_i \vee \bar{x}_i \leftarrow \bar{x}_j. & \mid & 1 \leq i \neq j \leq n \,\} \cup \\ & \{ \; x_i \leftarrow x_j, \bar{x}_j. & \bar{x}_i \leftarrow x_j, \bar{x}_j. & \mid & 1 \leq i \neq j \leq n \,\} \cup \\ & \{ \; x_i \leftarrow D_k^*. & \bar{x}_i \leftarrow D_k^*. & \mid & 1 \leq k \leq m, 1 \leq i \leq n \,\}, \end{array}$$

where D_k^* results from D_k by replacing literals $\neg x_i$ by $\bar{x}_i$.

Since P is positive, the SE-models of P are determined by its classical models, which are given by $\emptyset$, $X \cup \bar{X}$, and $\sigma(I)$, for each interpretation $I \subseteq X$ making F false. Hence, $(\emptyset, X \cup \bar{X})$ is an SE-model of P and $(\emptyset, X \cup \bar{X}) \in UE(P)$ iff F is a tautology. This proves coNP-hardness. □

In fact, also those UE-model checking problems which are feasible in polynomial time, are hard for the class P.

Theorem 19 *Given a pair of sets (X, Y) and a head-cycle free program P, the problem of deciding whether $(X, Y) \in UE(P)$ is P-complete. Hardness holds, even if P is definite Horn.*

Proof. Membership has already been shown in Theorem 18. We show hardness via a reduction from the P-complete problem HORNSAT to UE-model checking for Horn programs. Hence, let $\phi = \phi_f \wedge \phi_r \wedge \phi_c$ a Horn formula over atoms V, where $\phi_f = a_1 \wedge \cdots \wedge a_n$; $\phi_r = \bigwedge_{i=1}^{m} (b_{i,1} \wedge \cdots \wedge b_{i,k_i} \rightarrow b_i)$; and $\phi_c = \bigwedge_{i=1}^{l} \neg(c_{i,1} \wedge \cdots \wedge c_{i,k_i})$. Wlog suppose $n \geq 1$ (otherwise ϕ would be trivially satisfiable by the empty interpretation). Let u, w be new atoms, and take the program

$$\begin{array}{rcl} P & = & \{a_i \leftarrow u \mid 1 \leq i \leq n\} \cup \\ & & \{b_i \leftarrow b_{i,1}, \ldots, b_{i,k_i} \mid 1 \leq i \leq m\} \cup \\ & & \{w \leftarrow c_{i,1}, \ldots, c_{i,k_i} \mid 1 \leq i \leq l\} \cup \\ & & \{u \leftarrow v; \; v \leftarrow w \mid v \in V\} \cup \{u \leftarrow w\}. \end{array}$$

We show that ϕ is unsatisfiable iff $(\emptyset, V \cup \{u, w\})$ is UE-model of P. Note that both $\emptyset$ and $V \cup \{u, w\}$ are classical models of P for any ϕ. Since P is positive, it is sufficient to show that ϕ is satisfiable iff a model M of P exists, such that $\emptyset \subset M \subset (V \cup \{u, w\})$.

Suppose ϕ is satisfiable, and M is a model of ϕ; then it is easily checked that $M \cup \{u\}$ is a model of P. So suppose ϕ is unsatisfiable, and towards a contradiction let some M with $\emptyset \subset M \subset (V \cup \{u, w\})$ be a model of P. From the rules $\{v \leftarrow w \mid v \in V\} \cup \{u \leftarrow w\}$, we get $w \notin M$. Hence, the constraints ϕ_c are true under M. Since M is not empty, either $u \in M$ or some $v \in V$ is in M. However, the latter implies that $u \in M$ as well (by rules $\{u \leftarrow v \mid v \in V\}$). Recall that ϕ_f is not empty by assumption, hence all a_i's from ϕ_f are in M. Then, it is easy to see that $M \setminus \{u\}$ satisfies ϕ, which contradicts our assumption that ϕ is unsatisfiable. □

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 10 *Given two DLPs P and Q, deciding whether $P \equiv_u Q$ is in the class Π_2^P.*

Proof. To show that two DLPs 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 one of the programs P and Q. By Theorem 18, 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 20 *Given two DLPs P and Q, deciding whether $P \equiv_u Q$ is Π_2^P-complete. Hardness holds even if one of the programs is positive.*

Proof. Membership in Π_2^P has already been established in Lemma 10. 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 $QBF_{2,\forall}$ of form $F = \forall X \exists Y \phi$ with $\phi = \bigwedge_{i=1}^{i=m} C_i$, where each C_i is a disjunction of literals over the boolean variables in $X \cup Y$. Deciding whether a given such F is true is well known to be Π_2^P-complete.

For the moment, let us assume that $X = \emptyset$, i.e., the QBF amounts to a SAT-instance F over Y. More precisely, in what follows we reduce the satisfiability problem of the quantifier-free formula ϕ to the problem of deciding uniform equivalence of two programs P and Q. Afterwards, we take the entire QBF F into account.

Let a and b be fresh atoms and define

$$
\begin{aligned}
P \;=\;& \{y \vee \bar{y} \leftarrow \mid y \in Y\} \cup && (2)\\
& \{b \leftarrow y, \bar{y};\; y \leftarrow b;\; \bar{y} \leftarrow b \mid y \in Y\} \cup && (3)\\
& \{b \leftarrow \overline{C}_i \mid 1 \leq i \leq m\} \cup && (4)\\
& \{a \leftarrow\}. && (5)
\end{aligned}
$$

Note that P is positive. The second program is defined as follows:

$$
\begin{aligned}
Q \;=\;& \{y \vee \bar{y} \leftarrow z \mid y \in Y;\; z \in Y \cup \bar{Y} \cup \{a\}\} \cup && (6)\\
& \{b \leftarrow y, \bar{y};\; y \leftarrow b;\; \bar{y} \leftarrow b \mid y \in Y\} \cup && (7)\\
& \{b \leftarrow \overline{C}_i \mid 1 \leq i \leq m\} \cup && (8)\\
& \{a \leftarrow b;\; a \leftarrow not\, b;\; a \leftarrow not\, a\}. && (9)
\end{aligned}
$$

The only differences between the two programs P and Q are located in the rules (2) compared to (6) as well as (5) compared to (9). Note that (9) also contains default negation.

Let us first compute the SE-models of P. Since P is positive it is sufficient to consider classical models. Let $\mathcal{A} = Y \cup \bar{Y} \cup \{a, b\}$. First, $\mathcal{A}$ is clearly a classical model of P, and so is $\sigma(I) \cup \{a\}$, for each classical model $I \in M_\phi$. In fact, these are the only models of P. This can be seen as follows. By rules (2), at least one y or $\bar{y}$ must be contained in a model, for each $y \in Y$. By (3), if both y and $\bar{y}$ are contained in a candidate-model for some $y \in Y$ or b is contained in the candidate, then the candidate is spoiled up to $Y \cup \bar{Y} \cup \{b\}$. Hence the classical models of (2–3) are given by $\{\sigma(I) \mid I \subseteq Y\}$ and $Y \cup \bar{Y} \cup \{b\}$. Now, (4) eliminates those candidates which make ϕ false by "lifting" them to $Y \cup \bar{Y} \cup \{b\}$. By (5) we finally have to add a to the remaining candidates.

Hence, the SE-models of P are given by

$$\{(\sigma(I) \cup \{a\}, \sigma(I) \cup \{a\}) \mid I \in M_\phi\} \cup \{(\sigma(I) \cup \{a\}, \mathcal{A}) \mid I \in M_\phi\} \cup (\mathcal{A}, \mathcal{A}).$$

Obviously, each SE-model of P is also UE-model of P.

We now analyze Q. First observe that the classical models of P and Q coincide. This is due the fact that (5) is classically equivalent to (9) and thus classically derives a, making (6) and (2) do the same job in this context. However, since Q is not positive we have to consider the respective reducts of Q to compute the SE-models. We start with SE-models of the form $(X, \mathcal{A})$. In fact, $(X, \mathcal{A}) \in SE(Q)$ iff $X \in \{\emptyset, \mathcal{A}\} \cup \{\sigma(I) \mid I \in M_\phi\} \cup \{\sigma(I) \cup \{a\} \mid I \in M_\phi\}$. The remaining SE-models of Q are all total and, as for P, given by $\{(\sigma(I) \cup \{a\}, \sigma(I) \cup \{a\}) \mid I \in M_\phi\}$.

Hence, the set of all SE-models of Q is

$$\{(\sigma(I) \cup \{a\}, \sigma(I) \cup \{a\}) \mid I \in M_\phi\} \;\cup\; \{(\sigma(I) \cup \{a\}, \mathcal{A}) \mid I \in M_\phi\} \;\cup\; (\mathcal{A}, \mathcal{A}) \;\cup$$
$$\{(\sigma(I), \mathcal{A}) \mid I \in M_\phi\} \;\cup\; (\emptyset, \mathcal{A});$$

having additional SE-models compared to P, namely $(\emptyset, \mathcal{A})$ and $\{(\sigma(I), \mathcal{A}) \mid I \in M_\phi\}$. Note however, that the latter SE-models never are UE-models of Q, since clearly $\sigma(I) \subset (\sigma(I) \cup \{a\})$, for all $I \in M_\phi$.

Thus, if M_ϕ is not empty, the UE-models of P and Q coincide; otherwise there is a single non-total UE-model of Q, namely $(\emptyset, \mathcal{A})$. Note that the latter is not UE-model of Q in the case $M_\phi \neq \emptyset$ since, for each $I \in M_\phi$, $\sigma(I) \neq \emptyset$. Consequently, the UE-models of P and Q coincide iff M_ϕ is not empty, i.e., iff ϕ is satisfi able.

So far we have shown how to construct programs P and Q, such that uniform equivalence encodes SAT. To complete the reduction for the QBF, we now also take X into account.

We add in both P and Q the set of rules

$$\{x \vee \bar{x} \leftarrow;\; \leftarrow x, \bar{x} \mid x \in X\}$$

where the $\bar{x}$'s are fresh atoms. The set $\mathcal{A}$ remains as before, i.e., without any atom of the form x or $\bar{x}$.

This has the following effects. First the classical models of both P and Q are now given by $\sigma_{X \cup Y}(I) \cup \{a\}$, for each $I \in M_\phi$, and $(\sigma_{X \cup Y}(J) \cup \mathcal{A}) = (\sigma_X(J) \cup \mathcal{A})$, for each $J \subseteq X$. Therefore, the SE-models of P are given by

$$\{(\sigma_{X \cup Y}(I) \cup \{a\}, \sigma_{X \cup Y}(I) \cup \{a\}) \mid I \in M_\phi\} \cup \tag{10}$$
$$\{(\sigma_{X \cup Y}(I) \cup \{a\}, \sigma_X(I) \cup \mathcal{A}) \mid I \in M_\phi\} \cup \tag{11}$$
$$\{(\sigma_X(J) \cup \mathcal{A}, \sigma_X(J) \cup \mathcal{A}) \mid J \subseteq X\}. \tag{12}$$

Again, each SE-model of P is also UE-model of P. For Q the argumentation from above is used analogously. In particular, for each $J \subseteq X$, we get an additional SE-model $\{(\sigma_X(J), \sigma_X(J) \cup \mathcal{A})\}$ for Q. Thus, the UE-models of P and Q coincide iff, none of these additional SE-models $\{(\sigma_X(J), \sigma_X(J) \cup \mathcal{A})\}$ of Q is an UE-model of Q, as well. This is the case iff, for each $J \subseteq X$, there exists a truth assignment to Y making ϕ true, i.e., iff the QBF $\forall X \exists Y \phi$ is true.

Since P and Q are obviously constructible in polynomial time, our result follows. □

For the construction of P and Q in above proof we used—for matters of presentation—two additional atoms a and b. However, one can resign on b; by replacing rules (3) and (4) in both programs by $\{y \leftarrow \overline{C};\; \bar{y} \leftarrow \overline{C}_i \mid y \in Y;\; 1 \leq i \leq m\}$; and additionally rules (9) in Q by $\{a \leftarrow \overline{C};\; \bar{a} \leftarrow \overline{C}_i \mid 1 \leq i \leq m\} \cup \{\leftarrow \mathit{not}\, a\}$. Hence, already a single occurrence of default negation in one of the compared programs makes the problem harder. Note that equivalence of two positive disjunctive programs is among the coNP-problems discussed in the following.

Theorem 21 *Let P and Q be positive DLPs. Then, deciding whether $P \equiv_u Q$ is* coNP*- complete, where* coNP*-hardness holds even if one of the programs is Horn.*

Proof. By Theorem 12, uniform equivalence and strong equivalence are the same concepts for positive programs. Since strong equivalence is in coNP in general, the membership part of the theorem follows immediately.

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 atoms X, let

$$\begin{aligned} P &= \{C_i^* \vee a \leftarrow \mid 1 \leq i \leq m\} \cup \{\leftarrow x, \bar{x} \mid x \in X\}; \quad \text{and} \\ Q &= \{a \leftarrow\} \cup \{\leftarrow x, \bar{x} \mid x \in X\}. \end{aligned}$$

By Theorem 13, $P \equiv_u Q$ iff P and Q have the same classical models. The latter holds iff each model of P contains the atom a. But then, F is unsatisfiable. □

We now turn to head-cycle free programs.

Theorem 22 *Let P and Q be DLPs, and P head-cycle free. Then, deciding $P \equiv_u Q$ is* coNP*- complete, where* coNP*-hardness holds even if P is normal and Q is Horn.*

Proof. For the membership part, by Theorem 5, $P \equiv_u Q$ iff $P \models_u Q$ and $Q \models_u P$. Both tasks are in coNP (see Theorem 24 below). Since the class coNP is closed under conjunction, it follows that deciding $P \equiv_u Q$ is in coNP.

To show coNP-hardness consider the programs from the proof of Theorem 21. Indeed, P is HCF and, therefore, $P \equiv_u P^{\rightarrow}$ by Theorem 15. Using the same argumentation as above, yields $P^{\rightarrow} \equiv_u Q$ iff F is unsatisfiable. This shows the coNP-hardness result for comparing normal and Horn programs. □

Note that Sagiv showed [52] that deciding $P \equiv_u Q$ for given definite Horn programs P and Q is polynomial, which easily follows from his result that the property of uniform containment (whether the least model of $P \cup R$ is always a subset of $Q \cup R$) can be decided in polynomial time. As pointed out by Maher [43], Buntine [5] has like Sagiv provided an algorithm for deciding uniform containment.

Sagiv's result clearly generalizes to arbitrary Horn programs, since by Theorem 13, deciding $P \equiv_u Q$ reduces to checking classical equivalence of Horn theories, which is known to be P-complete.

Corollary 8 *Deciding uniform equivalence of Horn programs is* P*-complete.*

This concludes our analysis on the complexity of checking uniform equivalence. Our results cover all possible combinations of the classes of programs considered, i.e., DLPs, positive programs, normal programs, head-cycle free programs, as well as Horn programs, as already highlighted in Table 4.

Finally, we complement the results on uniform equivalence and UE-model checking with addressing the complexity of UE-consequence. The proofs of these results can be found in the Appendix.

Theorem 23 *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.*

Theorem 24 *Let P, Q be DLPs. Then, $P \models_u Q$ is* coNP*-complete, whenever one of the programs is head-cycle free.* coNP*-hardness holds, even if P is normal and Q is Horn.*

$P \equiv^A_s Q$ / $P \equiv^A_u Q$	DLP	positive	HCF	normal	Horn
Horn	Π^P_2	coNP	coNP	coNP	coNP
normal	Π^P_2	Π^P_2	Π^P_2/coNP	coNP	
HCF	Π^P_2	Π^P_2	Π^P_2/coNP		
positive	Π^P_2	Π^P_2			
DLP	Π^P_2				

Table 5: Complexity of Relativized Equivalences in Terms of Completeness Results.

6.2 Complexity of Relativized Equivalence

We now generalize the complexity results to relativized forms of equivalence. In particular, we investigate the complexity of A-SE/UE-model checking as well as of the equivalence problems $\equiv^A_s$ and $\equiv^A_u$, respectively. Like in the previous section, we also consider different classes of programs. Our results are summarized in Table 5 for both RSE and RUE at a glance by just highlighting where the complexity differs. Note that the only differences between RSE and RUE stem from the entries Π^P_2/coNP in the column for head-cycle free programs. Here we have that in the cases HCF/HCF and HCF/normal, checking $\equiv^A_s$ is in general harder for RSE than for RUE. Another issue to mention is that already for uniform equivalence, the concept of relativization make things more difficult. One just needs to compare the first two columns of Tables 4 and 5, respectively. Even worse for strong equivalence, which is in coNP in its unrelativized version and now jumps up to Π^P_2-completeness in several cases. Finally, also the comparison of two Horn programs becomes intractable, viz. coNP-complete, compared to the polynomial-time result in the cases of unrelativized strong and uniform equivalence.

To summarize, RSE and RUE (i) are harder to decide than in their unrelativized versions in several cases, and (ii) both are generally of the same complexity except head-cycle free programs are involved. Note that Observation (ii), on the one hand, contrasts the current view that notions of strong equivalence have milder complexity than notions like uniform equivalence. On the other hand, the intuition behind this gap becomes apparent if one takes into account that for HCF programs P, $P \equiv^A_u P^{\rightarrow}$ holds, while $P \equiv^A_s P^{\rightarrow}$ does not.

For an even more fine-grained picture, note that problems associated with equivalence tests relative to an atom set A call for further distinctions between several cases concerning the concrete instance A. We identify the following ones:

- $card(A) = 0$: In this case, both A-SE and A-UE-model checking collapse to answer set checking; correspondingly, RSE and RUE collapse to ordinary equivalence;

- $card(A) < 2$: By Proposition 8 and Corollary 4, A-SE-models and A-UE-models coincide, and thus, RSE and RUE are the same concepts.

Our results for $\equiv^A_e$, $e \in \{s, u\}$, given in the following, consider arbitrary fixed A unless stated otherwise. Moreover, we consider that A contains only atoms which also occur in the programs under consideration. In some cases the hardness-part of the complexity results is obtained only if $card(A) > k$ for some constant k. We shall make these cases explicit.

Another special case for A is to consider *bounded relativization*. This denotes the class of problems where the cardinality of $(V \setminus A)$ is less or equal than a fixed constant k, with V being the atoms occurring in the two programs compared. Note that this concepts contains strong and uniform equivalence, respectively, as special cases, i.e., if $(V \setminus A) = \emptyset$. We deal with bounded relativization explicitly in the subsequent section.

Towards deriving the results from Table 5, we first consider model checking problems. Formally, for a set of atoms A, the problem of A-SE-model checking (resp. A-UE-model checking) is defined as follows: Given sets of atoms X, Y, and a program P, decide whether $(X, Y) \in SE^A(P)$ (resp. $(X, Y) \in UE^A(P)$). We start with the following tractable cases.

Theorem 25 *Given a pair of sets (X, Y), a set of atoms A, and a program P, the problem of deciding whether $(X, Y) \in SE^A(P)$ (resp. $(X, Y) \in UE^A(P)$) is feasible in polynomial time with respect to $\|P\| + \|X\| + \|Y\|$, whenever P is normal (resp. whenever P is HCF).*

Proof. We start with the test whether (X, Y) is A-SE-model of a normal program P. Note that P^Y is Horn, and that Y is a model of P^Y iff Y is a model of P. Consider the following algorithm

1. Check whether Y is a model of P^Y.
2. Check whether $P_Y = P^Y \cup (Y \cap A) \cup Y_{\subset}$ is unsatisfiable.
3. If $X \subset Y$, check whether $P_X = P^Y \cup (X \cap A) \cup \{\leftarrow x \mid x \in (A \setminus X)\} \cup Y_{\subseteq}$ is satisfiable.

Note that each step is feasible in polynomial time, especially since both P_X and P_Y are Horn. Hence, it remains to proof that above algorithm holds, exactly if (X, Y) is A-SE-model of P. This is seen as follows: each step exactly coincides with one of the conditions of checking whether (X, Y) is an A-SE-model, i.e., (1) $Y \models P$; (2) for all $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$, $Y' \not\models P^Y$; and (3) $X \subset Y$ implies existence of a $X' \subseteq Y$ with $(X' \cap A) = X$, such that $X' \models P^Y$.

For the result on A-UE-model checking we use a similar argumentation. First suppose that P is normal and consider the algorithm from above but replacing the second step by

2a. Check whether $P^Y \cup (X \cap A) \cup Y_{\subset} \models (X \cap A) \cup \{\leftarrow x \mid x \in (A \setminus X)\}$.

The desired algorithm then corresponds to the respective conditions for A-UE-model checking following Proposition 7. To be more specific, the models of $P^Y \cup (X \cap A) \cup Y_{\subset}$ are those X' with $(X \cap A) \subseteq X' \subset Y$ such that $X' \models P^Y$. The set of models of the right-hand side is given by $\{Z \mid (Z \cap A) = (X \cap A)\}$. Hence, the test in [2a.] is violated iff there exists an X' with $(X \cap A) \subset (X' \cap A)$ and $X' \subset Y$ such that $X' \models P^Y$, i.e., iff $(X, Y) \notin UE^A(P)$. Moreover, for HCF programs, $P^{\rightarrow}$ is A-UE-equivalent to P, following Theorem 15, i.e., the A-UE-models for P and $P^{\rightarrow}$ coincide. Applying $P^{\rightarrow}$ to the presented procedure thus shows that A-UE-model checking is feasible in polynomial time also for HCF programs. □

Without a formal proof, we mention that these tractable model checking problems are complete for the class P. Indeed, one can re-use the argumentation from the proof of Theorem 19 and take, for instance, $A = \{u\}$. Then, $(\emptyset, V \cup \{u, w\}) \in SE^A(P) = UE^A(P)$ iff the encoded Horn formula is satisfiable. Note that P-hardness holds also for answer set checking (i.e., $A = \emptyset$) by the straightforward observation that a Horn program P has an answer set iff P is satisfiable.

Next, we consider the case of A-SE-model checking for head-cycle free programs. Recall that for $card(A) < 2$, A-SE-model checking coincides with A-UE-model checking, and thus in these cases A-SE-model checking is feasible in polynomial time, as well. However, in general, A-SE-model checking is harder than A-UE-model checking for head-cycle free programs.

Theorem 26 *Let (X, Y) be a pair of interpretations, and P a head-cycle free program. Deciding whether $(X, Y) \in SE^A(P)$ is* NP*-complete. Hardness holds for any fixed A with $card(A) \geq 2$.*

Proof. For the membership result we argue as follows. First we check whether $(Y, Y) \in SE^A(P)$. Note that $(Y, Y) \in SE^A(P)$ iff $(Y, Y) \in UE^A(P)$. By Theorem 25 the latter test is feasible in polynomial time. It remains to check whether there exists a $X' \subseteq Y$ with $(X' \cap A) = X$, such that $X' \models P^Y$. This task is in NP, and therefore, the entire test is in NP.

For the corresponding NP-hardness, consider the problem of checking satisfiability of a formula $\psi = \bigwedge_{j=1}^{m} C_j$ in CNF given over a set of atoms V. This problem is NP-complete. We reduce it to A-SE-model checking for a HCF program. Consider the following program with additional atoms a_1, a_2, $\bar{V} = \{\bar{v} \mid v \in V\}$, and let $A = \{a_1, a_2\}$.

$$P = \{v \vee \bar{v} \leftarrow \mid v \in V\} \tag{13}$$
$$\{v \leftarrow a_1;\ \bar{v} \leftarrow a_1 \mid v \in V\} \tag{14}$$
$$\{a_2 \leftarrow \overline{C}_j \mid 1 \leq j \leq m\} \tag{15}$$
$$\{a_2 \leftarrow v, \bar{v} \mid v \in V\}. \tag{16}$$

Note that P is HCF. Let $Y = V \cup \bar{V} \cup A$. We show that $(\emptyset, Y) \in SE^A(P)$ iff ψ is satisfiable. It is clear that $Y \models P$ and no $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$ satisfies $Y' \models P^Y = P$ due to Rules (14). This shows $(Y, Y) \in SE^A(P)$. Now, $(\emptyset, Y) \in SE^A(P)$ iff there exists a $X \subseteq (V \cup \bar{V})$ such that $X \models P^Y = P$. Suppose $X \models P$. Since $a_2 \notin X$, X must represent a consistent guess due to Rules (13) and (16). Moreover, X has to represent a model of ψ due to Rules (15). Finally, $X \models (14)$ holds by trivial means, i.e., since $a_1 \notin X$. The converse direction is by analogous arguments. Hence, $(\emptyset, Y) \in SE^A(P)$ iff there exists a model of ψ, i.e., iff ψ is satisfiable.

This shows hardness for $card(A) = 2$. To obtain coNP-hardness for any A with $k = card(A) > 2$ and, such that all $a \in A$ are also occurring in the program, consider P as above augmented by rules $\{a_{i+1} \leftarrow a_i \mid 2 \leq i < k\}$ and $A = \{a_i \mid 1 \leq i \leq k\}$. By analogous arguments as above, one can show that then $(\emptyset, (V \cup \bar{V} \cup A)) \in SE^A(P)$ iff ψ is satisfiable. □

The next result concerns A-SE-model checking and A-UE-model checking of disjunctive logic programs in general and positive DLPs. For $A = \emptyset$, these tasks coincide with answer set checking which is known to be coNP-complete (see, for instance, [21]). Already a single element in A yields a mild increase of complexity.

Theorem 27 *Let (X, Y) be a pair of interpretations, and P a DLP. Deciding whether $(X, Y) \in SE^A(P)$ (resp. $(X, Y) \in UE^A(P)$) is D^P-complete. Hardness holds for any fixed A with $card(A) \geq 1$ even for positive programs.*

Proof. We first show D^P-membership. By Definition 8, a pair of interpretations (X, Y) is an A-SE-model of P iff (1) (X, Y) is a valid A-SE-interpretation; (2) $Y \models P$; (3) for all $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$, $Y' \not\models P^Y$; and (4) $X \subset Y$ implies existence of a $X' \subseteq Y$ with $(X' \cap A) = X$, such that $X' \models P^Y$ holds. Obviously, (1) and (2) can be verified in polynomial time. The complementary problem of (3) can be verified by a guess for Y' and a derivability check. As well, (4) can be verified by a guess for X' and a derivability check. Hence, (3) is in coNP and (4) is in NP, which shows D^P-membership. Similar in the case of A-UE-models. By Proposition 7, $(X, Y) \in UE^A(P)$ iff (1) (X, Y) is a valid A-SE-interpretation; (2) $Y \models P$; (3) for each $X'' \subset Y$ with $(X \cap A) \subset (X'' \cap A)$ or $X'' = (Y \cap A)$, $X'' \not\models P^Y$ holds; and (4) $X \subset Y$ implies that there exists a $X' \subseteq Y$ with $(X' \cap A) = X$, such that $X' \models P^Y$. Similar as before one can verify that the first two conditions are feasible in polynomial time, whereas checking (3) is a coNP-test, and checking (4) a NP-test.

For the matching lower bound, we consider the case $card(A) = 1$. Therefore, D^P-hardness of both A-SE-model checking and A-UE-model checking are captured at once. We consider the problem of jointly checking whether

(a) a formula $\phi = \bigvee_{i=1}^{n} D_i$ in DNF is a tautology; and

(b) a formula $\psi = \bigwedge_{j=1}^{m} C_j$ in CNF is satisfiable.

This problem is D^P-complete, even if both formulas are given over the same set of atoms V. Consider the following positive program

$$P = \{v \vee \bar{v} \leftarrow \mid v \in V\} \quad (17)$$
$$\{v \leftarrow a_1, D_i^*;\ \bar{v} \leftarrow a_1, D_i^* \mid v \in V,\ 1 \leq i \leq n\} \quad (18)$$
$$\{a_1 \leftarrow \overline{C}_j \mid 1 \leq j \leq m\} \quad (19)$$
$$\{a_1 \leftarrow v, \bar{v} \mid v \in V\}; \quad (20)$$

where a_1 is a fresh atom. Let $Y = \{a_1\} \cup V \cup \bar{V}$ and $A = \{a_1\}$. We show that $(\emptyset, Y)$ is A-SE-model of P iff (a) and (b) jointly hold. Since P is positive, we can argue via classical models (over Y). Rules (17) have classical models $\{X \mid \sigma(I) \subseteq X \subseteq Y, I \subseteq V\}$. By (18) this set splits into $S = \{X \mid \sigma(I) \subseteq X \subseteq (Y \setminus \{a_1\})\}$ and $T = \{\sigma(I) \cup \{a_1\} \mid I \notin M_\phi\} \cup \{Y\}$. By (20), S reduces to $\{\sigma(I) \mid I \subseteq V\}$, and by (19) only those elements $\sigma(I)$ survive with $I \in M_\psi$. To summarize, the models of P are given by

$$\{\sigma(I) \mid I \subseteq V, I \in M_\psi\} \cup \{\sigma(I) \cup \{a_1\} \mid I \subseteq V, I \notin M_\phi\} \cup \{Y\}.$$

From this the A-SE-models are easily obtained. We want to check whether $(\emptyset, Y) \in SE^A(P)$. We have $Y \models P$. Further we have that no $Y' \subset Y$ with $a_1 \in Y'$ exists such that $Y' \models P = P^Y$ iff there exists no $I \subseteq V$ making ϕ false, i.e., iff ϕ is a tautology. Finally, to show that $(\emptyset, Y) \in SE^A(P)$, there has to exist an $X \subseteq (V \cup \bar{V})$, such that $X \models P = P^Y$. This holds exactly if ψ is satisfiable. Since P is always polynomial in size of ϕ plus ψ, we derive D^P-hardness.

This shows the claim for $card(A) = 1$. For $card(A) > 1$, we apply a similar technique as in the proof of Theorem 26. However, since we deal here with both A-SE-models and A-UE-models we have to be a bit more strict. Let $k = card(A) > 1$. We add to P the following rules $\{a_{i+1} \leftarrow a_i;\ a_i \leftarrow a_{i+1} \mid 1 \leq i < k\}$ and set $A = \{a_i \mid 1 \leq i \leq k\}$. One can show that then, for $Y = A \cup V \cup \bar{V}$, $(\emptyset, Y) \in SE^A(P)$ iff $(\emptyset, Y) \in UE^A(P)$ iff (a) and (b) jointly hold. □

With these results for model checking at hand, we obtain numerous complexity results for deciding relativized equivalence.

Theorem 28 *For programs P, Q, a set of atoms A, and $e \in \{s, u\}$, $P \equiv_e^A Q$ is in Π_2^P.*

Proof. We guess an A-SE-interpretation (X, Y). Then, by virtue of Theorem 27, we can verify that (X, Y) is A-SE-model (resp. A-UE-model) of exactly one of the programs P, Q in polynomial time with four calls to an NP-oracle (since the two model-checking tasks are in D^P). Hence, the complementary problem of deciding relativized equivalence is in Σ_2^P. This shows Π_2^P-membership. □

Theorem 29 *Let P, Q be DLPs, A a set of atoms, and $e \in \{s, u\}$. Then, $P \equiv_e^A Q$ is Π_2^P-complete. Π_2^P-hardness holds even if Q is Horn.*

Proof. Membership is already shown in Theorem 28.

For the hardness part, we reduce the Σ_2^P-complete problem of deciding truth of a QBF $\exists X \forall Y \phi$ with $\phi = \bigvee_{i=1}^n D_i$ a DNF to the complementary problem $P \not\equiv_s^A Q$. We define

$$\begin{aligned} P \;=\; & \{x \vee \bar{x} \leftarrow;\ \leftarrow x, \bar{x} \mid x \in X\} \cup \\ & \{y \vee \bar{y} \leftarrow;\ y \leftarrow a;\ \bar{y} \leftarrow a;\ a \leftarrow y, \bar{y} \mid y \in Y\} \cup \\ & \{a \leftarrow D_i^* \mid 1 \leq i \leq n\} \cup \\ & \{\leftarrow not\, a\}; \end{aligned}$$

and take $Q = \{\bot\}$. Note that $\{\bot\}$ has no A-SE-model, for any A. It thus remains to show that P has an A-SE-model iff the QBF $\exists X \forall Y \phi$ is true.

P has an answer set (i.e., an $\emptyset$-SE-model) iff $\exists X \forall Y \phi$ is true (see the Σ_2^P-hardness proof for the program consistency problem in [21]). From this we get that ordinary equivalence is Π_2^P-hard. This shows the claim for $card(A) = 0$. For A of arbitrary cardinality k it is sufficient to add "dummy" rules $a_i \leftarrow a_i$, for each $1 \leq i \leq k$, to P. These rules do not have any effect on our argumentation. Whence, for any fixed A, $\equiv_s^A$ and $\equiv_u^A$ are Π_2^P-hard as well. □

A slight modification (see Appendix for details) of this proof gives us the following result.

Theorem 30 *Let P be a positive program, A a set of atoms, and $e \in \{s, u\}$. Then, deciding whether $P \equiv_e^A Q$ is Π_2^P-complete, where Π_2^P-hardness holds even if Q is either positive or normal.*

For head-cycle free programs, RSE and RUE have different complexities. We first consider RSE.

Theorem 31 *Let P and Q be head-cycle free programs, and A be a set of atoms. Then, deciding whether $P \equiv_s^A Q$ is Π_2^P-complete, where Π_2^P-hardness holds even if Q is normal, and fixed A with $card(A) \geq 2$.*

Proof. As before, we reduce the problem of deciding truth of a QBF of the form $\exists X \forall Y \phi$, with ϕ a DNF, to the complementary problem of $P \equiv_s^A Q$ using for P a head-cycle free program and for Q a normal program. We use similar building blocks as in the proofs of the previous results, but the argumentation is more complex here. We need a further new atom b, and define

$$\begin{aligned} P \;=\; & \{x \vee \bar{x} \leftarrow;\ \leftarrow x, \bar{x} \mid x \in X\} \cup \\ & \{y \vee \bar{y} \leftarrow;\ y \leftarrow a;\ \bar{y} \leftarrow a \mid y \in Y\} \cup \\ & \{b \leftarrow D_i^* \mid 1 \leq i \leq n\} \cup \\ & \{b \leftarrow y, \bar{y} \mid y \in Y\} \cup \{b \leftarrow a\}. \end{aligned}$$

Note that P is head-cycle free. For the matter of presentation, suppose first $X = \emptyset$. We show that ϕ is valid iff $P \not\equiv_s^A P^{\rightarrow}$ holds, for $A = \{a, b\}$. Afterwards, we generalize the claim to arbitrary X and show that $P \not\equiv_s^A P^{\rightarrow}$ iff $\exists X \forall Y \phi$ is true holds, for any A of the form $\{a, b\} \subseteq A \subseteq (X \cup \bar{X} \cup \{a, b\})$.

Let us first compute the A-SE-models of P under the assumption that $X = \emptyset$. Since P is positive, this is best accomplished by first considering the classical models of P. These are given as follows:

(a) $\sigma(I)$ for each $I \subseteq Y$ making ϕ false;

(b) $\sigma(I) \cup \{b\}$ for each $I \subseteq Y$;

(c) all M satisfying $(\sigma(I) \cup \{b\}) \subset M \subseteq (Y \cup \bar{Y} \cup \{b\})$ for some $I \subseteq Y$; and

(d) $\mathcal{A} = Y \cup \bar{Y} \cup \{a, b\}$.

Note that (a), (b), and (d) become total A-SE-models of P; while the elements in (c) do not. In fact, for each element M in (c) there exists a corresponding element M' from (b), such that $M' \subset M$ and $(M' \cap A) = (M \cap A) = \{b\}$. It remains to consider non-total A-SE-models of P, by combining the elements from (a), (c), (d). If there exists an element in (a) (i.e., ϕ is not valid), then we get $(\emptyset, \sigma(I) \cup \{b\}) \in SE^A(P)$, for each $I \subseteq V$; as well we then have also $(\emptyset, \mathcal{A}) \in SE^A(P)$. Combining (b) and (c), yields $(\{b\}, \mathcal{A}) \in SE^A(P)$. Hence,

$$\begin{aligned} SE^A(P) \;=\; & \{(\sigma(I), \sigma(I)) \mid I \subseteq V : \phi \text{ is false under } I\} \cup \\ & \{(\sigma(I) \cup \{b\}, \sigma(I) \cup \{b\}) \mid I \subseteq V\} \cup \\ & \{(\emptyset, \sigma(I) \cup \{b\}) \mid I \subseteq V, \text{ if } \phi \text{ is not valid}\} \cup \\ & \{(\emptyset, \mathcal{A}) \mid \text{if } \phi \text{ is not valid}\} \cup \\ & \{(\{b\}, \mathcal{A}), (\mathcal{A}, \mathcal{A})\}. \end{aligned}$$

For $P^{\rightarrow}$ we get a (possibly) additional A-SE-model, viz. $(\emptyset, \mathcal{A})$, since $(\emptyset, \mathcal{A}) \in SE(P^{\rightarrow})$ holds in any case, also if ϕ is valid. Hence, the A-SE-models of P and $P^{\rightarrow}$ coincide iff ϕ is not valid.

The extension to $X \neq \emptyset$ and deciding truth of QBF $\exists X \forall Y \phi$ via the complementary problem $\equiv_s^A$ is similar to the argumentation in the proof of Theorem 20. In particular, we then can use any A with $\{a, b\} \subseteq A \subseteq (X \cup \bar{X} \cup \{a, b\})$. Recall that deciding $\exists X \forall Y \phi$ is Σ_2^P-complete, and thus we get that $P \equiv_s^A Q$ is Π_2^P-hard for P a HCF program, Q normal. □

This concludes the collection of problems which are located at the second level of the polynomial hierarchy. Note that in the hardness part of the proof of Theorem 31, we used at least two elements in A. In fact, for HCF programs and $card(A) \leq 1$ the complexity is different. Since for $card(A) \leq 1$, $\equiv_s^A$ and $\equiv_u^A$ are the same concepts, this special case is implicitly considered in the next theorem. Another issue is to decide $P \equiv_s^A Q$ if both P and Q are A-HCF as introduced in Definition 14. In this case, we can employ $P^{\rightarrow} \equiv_s^A Q^{\rightarrow}$, and thus the complexity coincides with the complexity of $\equiv_s^A$ for normal programs. This is also part of the next theorem.

Theorem 32 *Deciding $P \equiv_e^A Q$ is* coNP*-complete in the following settings:*

(i) $e \in \{s, u\}$*, P positive, Q Horn;*

(ii) $e = s$*, P head-cycle free and Q Horn;*

(iii) $e \in \{s, u\}$*, P and Q normal;*

(iv) $e = u$*, P and Q head-cycle free.*

coNP*-hardness of $P \equiv_e^A Q$ $(e \in \{s, u\})$ holds even if P is normal or positive and Q is Horn.*

Proof. We start with the coNP-membership results. The cases (iii) and (iv) follow immediately from Theorem 25, since A-SE/UE-model checking for the programs involved is feasible in polynomial time. The more complicated cases (i) and (ii) are addressed in the Appendix.

It remains to show the coNP-hardness part of the theorem. We use UNSAT of a formula $F = \bigwedge_{i=1}^{n} C_i$ in CNF over atoms X. Take

$$P = \{x \vee \bar{x} \leftarrow;\ \leftarrow x, \bar{x} \mid x \in X\} \cup \{\leftarrow \overline{C}_i \mid 1 \leq i \leq n\}$$

Note that this program is positive and HCF. The program has a classical model iff F is satisfiable, i.e., iff it is not equivalent to the Horn program $Q = \{\bot\}$. In other words, $SE^A(P) \neq \emptyset$ (or, resp. $UE^A(P) \neq \emptyset$) iff ϕ is satisfiable. Note that A can thus be of any form. Since the rules $\leftarrow x, \bar{x}$ are present in P, we have $P \equiv_s^A P^{\rightarrow}$. This proves coNP-hardness also for the case where one program is normal and the other is Horn. □

A final case remains open, namely checking relativized equivalence of Horn programs. Unfortunately, this task is coNP-complete. However, whenever the cardinality of A is fixed by a constant the problem gets tractable. This is in contrast to the hardness results proved so far, which even hold in the case where $card(A)$ is fixed. The proof of the theorem is given in the Appendix.

Theorem 33 *Deciding $P \equiv_e^A Q$, for $e \in \{s, u\}$, is coNP-complete for Horn programs P, Q. Hardness holds whenever $card(A)$ is not fixed by a constant, and even for definite Horn programs.*

Whenever the cardinality of A is bounded, we can decide this problem in polynomial time.

Theorem 34 *Let P, Q be Horn programs and A be a set of atoms such that $card(A) \leq k$ with a fixed constant k. Then, deciding $P \equiv_e^A Q$ is feasible in polynomial time with respect to $\|P\| + \|Q\| + k$.*

Proof. It is sufficient to show the claim for $e = u$. By explicitly checking whether $(P \cup S) \equiv (Q \cup S)$ holds for any $S \subseteq A$. we obtain a polynomial-time algorithm, since checking ordinary equivalence of Horn programs is polynomial and we need at most 2^k such checks. □

6.3 Complexity of Bounded Relativization

In this section, we pay attention to the special case of tests $\equiv_s^A$ and $\equiv_u^A$ where the number of atoms from the considered programs *missing* in A, is bounded by some constant k (in symbols $P\ {}^k\!\!\equiv_s^A Q$, and resp., $P\ {}^k\!\!\equiv_u^A Q$). Hence, the respective problem classes apply to programs P, Q, only if $card(Atm(P \cup Q) \setminus A) \leq k$. Apparently, this class of problems contains strong and uniform equivalence in its unrelativized versions ($k = 0$). The complexity results are summarized in Table 6. In particular, we get that in the case of RSE all entries (except Horn/Horn) reduce to coNP-completeness. This generalizes results on strong equivalence. Previous work reported some of these results but not in form of this exhaustive list.

In what follows, we first give the respective results for model checking, and then we prove the entries in Table 6.

Lemma 11 *For a program P, and a set of atoms A, such that $card(Atm(P) \setminus A) \leq k$, with k a fixed constant, A-SE-model checking is feasible in polynomial time with respect to to $\|P\| + k$.*

Proof. By the conditions in Definition 8, deciding $(X, Y) \in SE^A(P)$ can be done as follows: (i) checking $Y \models P$; (ii) checking whether for all $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$, $Y \not\models P^Y$ holds; and (iii) if $X \subset Y$, checking existence of a $X' \subseteq Y$ with $(X' \cap A) = X$, such that $X' \models P^Y$ holds. Test (i) can be done in polynomial time; test (ii) is a conjunction of at most $2^k - 1$ independent polynomial tests (for each such Y'),

$P\,{}^{k}{\equiv}^{A}_{s}\,Q$ / $P\,{}^{k}{\equiv}^{A}_{u}\,Q$	DLP	positive	HCF	normal	Horn
Horn	coNP	coNP	coNP	coNP	P
normal	coNP	coNP	coNP	coNP	
HCF	coNP	coNP	coNP		
positive	coNP/Π_2^P	coNP			
DLP	coNP/Π_2^P				

Table 6: Complexity of Equivalences with Bounded Relativization in Terms of Completeness Results.

while (iii) is a disjunction of at most 2^k polynomial tests (for each X'). Since we have fixed k the entire test is feasible in polynomial time. □

Compared to the model checking problems discussed so far, the polynomial-time decidable problems of A-SE-model checking in the bounded case do not belong to the class of P-complete problems, but are easier. This is best illustrated by SE-model checking, which obviously reduces to two (ordinary) independent model checking tests; which in turn are in ALOGTIME [6] (see also [3, 29]). For bounded A-SE-model checking the situation is basically the same, since it is sufficient to employ a fixed number of independent model checking tests.

Concerning UE-model checking we already established some P-hardness results in Theorem 19 which generalize to the relativized case for arbitrary bound A. In general, for A-UE-model checking the decrease of complexity is in certain cases only moderate compared to the corresponding decrease in the case of A-SE-model checking.

Lemma 12 *For a program P and a set of atoms A, such that $card(Atm(P) \setminus A) \leq k$, with k a fixed constant, A-UE-model checking is* coNP*-complete. Hardness holds even for positive programs.*

Proof. We show NP-membership for the complementary problem, i.e., checking whether a given pair (X, Y) is not in $UE^A(P)$. We first check whether (X, Y) is A-SE-model of P. This can be done in polynomial time, by Lemma 11. If this is not the case we are done; otherwise, we guess an X' with $X \subset X' \subset (Y \cap A)$ and check whether (X', Y) is A-SE-model of P. This guess for (X', Y) can be verified in polynomial time using an NP oracle. Therefore, the entire problem is in NP. The correctness of the procedure is given by its direct reflection of Definition 9. This yields coNP-membership for bounded A-UE-model checking.

Hardness is obtained via the case $card(Atm(P) \setminus A) = 0$, i.e., ordinary UE-model checking and the respective result in Theorem 18. □

Theorem 35 *For programs P, Q and a set of atoms A, such that $card(Atm(P \cup Q) \setminus A) \leq k$ with k a fixed constant, $P \equiv^A_s Q$ is* coNP*-complete. Hardness holds provided P and Q are not Horn.*

Proof. By Lemma 11, A-SE-model checking is feasible in polynomial time in the bounded case. Hence, coNP-membership for $P \equiv^A_s Q$ is an immediate consequence. The hardness result is easily obtained by the hardness part from Theorem 32. □

For RUE some cases remain on the second level, however. This is not a surprise, since as we have seen in Theorem 20, (unrelativized) uniform equivalence is Π_2^P-complete in general.

Theorem 36 *For programs P, Q and a set of atoms A, such that $card(Atm(P \cup Q) \setminus A) \leq k$ with k a fixed constant, $P \equiv^A_u Q$ is Π^P_2-complete. Π^P_2-hardness holds even if one of the programs is positive.*

Proof. Membership is obtained by the fact that A-UE-model checking with A bounded is coNP-complete (see Lemma 12). Hardness comes from the Π^P_2-hardness of uniform equivalence. □

For all other cases, RUE for bounded A is in coNP.

Theorem 37 *For programs P, Q and a set of atoms A, such that $card(Atm(P \cup Q) \setminus A) \leq k$ with k a fixed constant, $P \equiv^A_u Q$ is coNP-complete, if either (i) both programs are positive; or (ii) at least one program is head-cycle free. Hardness holds, even if P is normal or positive and Q is Horn.*

Proof. We start with coNP-membership. For (i) this is an immediate consequence of the fact that for positive programs, RSE and RUE are the same concepts and since RSE is coNP-complete as shown in Theorem 35.

For (ii) we argue as follows. Consider P is HCF. By Theorem 11 it is sufficient to check (a) $UE^A(P) \subseteq SE^A(Q)$ and (b) $UE^A(Q) \subseteq SE^A(P)$. We show that both tasks are in coNP. ad (a): For the complementary problem we guess a pair (X, Y) and check whether $(X, Y) \in UE^A(P)$ and $(X, Y) \notin SE^A(Q)$. Both checks are already shown to be feasible in polynomial time. ad (b): We consider the complementary problem and show that this reduces to the disjunction of two NP problems. First, we consider total A-SE-interpretations. By guessing Y and check whether $(Y, Y) \in SE^A(Q)$ and $(Y, Y) \notin SE^A(P)$, we get obtain NP-membership. If this holds, we secondly we consider non-total A-SE-interpretations. We claim that existence of a $(X, Y) \in SE^A(Q)$, such that, for each $X \subseteq X' \subset (Y \cap A)$, $(X', Y) \notin UE^A(P^{\rightarrow})$, implies $UE^A(Q) \not\subseteq SE^A(P)$. This can be seen as follows. Given X, Y, suppose no $X \subseteq X' \subset (Y \cap A)$ satisfies $(X', Y) \in UE^A(P^{\rightarrow})$. Then, no such (X', Y) is A-UE-model of the original P (by Theorem 15). By definition, no such (X', Y) is A-SE-model of P. On the other hand, either $(X, Y) \in UE^A(Q)$ or for some such X', $(X', Y) \in UE^A(Q)$. Hence, $UE^A(Q) \not\subseteq SE^A(P)$. Therefore, we guess a pair (X, Y) and check $(X, Y) \in SE^A(Q)$ and whether $T = (P^{\rightarrow})^Y \cup X \cup Y_{\subset}$ is unsatisfiable. Both can be done in polynomial time. It remains to show that T is unsatisfiable iff, for each $X \subseteq X' \subset (Y \cap A)$, $(X', Y) \notin UE^A(P^{\rightarrow})$. Suppose T is satisfiable and let X' be a maximal interpretation making T true. Then $(X' \cap A) \subset (Y \cap A)$ holds, since $(Y, Y) \in UE^A(P)$ (and thus $(Y, Y) \in SE^A(P^{\rightarrow})$) by assumption that the total A-SE-models of P and Q coincide. But then $((X' \cap A), Y) \in UE^A(P^{\rightarrow})$, since X' is a maximal model of T. On the other hand, if T is unsatisfiable, no (X', Y) with $X \subseteq X' \subset (Y \cap A)$ can be A-SE-model of P, and thus no such (X', Y) is A-UE-model of P and thus of $P^{\rightarrow}$. This gives membership for NP. Since NP is closed under disjunction, the entire complementary problem is shown to be in NP.

The matching lower bound is obtained from the hardness result in Theorem 32. □

One final case remains to be considered.

Theorem 38 *Let P, Q be Horn programs and let A be a set of atoms such that $card(Atm(P \cup Q) \setminus A) \leq k$ with a fixed constant k. Then, deciding $P \equiv^A_e Q$ is feasible in polynomial time with respect to $\|P\| + \|Q\| + k$.*

Proof. We use the following characterization which can be derived from Theorem 14: For positive programs P, Q, $P \equiv^A_e Q$ holds, iff, for each model Y of P, there exists a $X \subseteq Y$ with $(X \cap A) = (Y \cap A)$ being model of Q; and vice versa. This can be done as follows. We show one direction, i.e., whether, for each interpretation Y, $Y \models P$ implies $X \models Q$ for some $X \subseteq Y$, such that $(X \cap A) = (Y \cap A)$. Let $V = (Atm(P \cup Q) \setminus A)$. We test, for every $U \subseteq V$ and each $W \subseteq U$, whether

$$P'_V \cup (U^V_=)' \cup (W^V_=) \models Q; \tag{21}$$

where P' results from P by replacing each $v \in V$ occurring in P by v' and $(U^V_=)'$ is the set $\{v' \mid v \in U^V_=\}$ with $U^V_=$ as defined in the beginning of the section. Observe that both sides in the derivability test (21) are Horn programs.

$P'_V \cup (U^V_=)'$ has a model $R \cup S'$ iff there exists a $R \subseteq A$ and a $S' \subseteq V'$ such that $R \cup S'$ is a model of P'_V, i.e., iff $R \cup S$ is a model of P. Then, we check whether for one $W \subseteq U$, $R \cup W$ is model of Q. This matches the test whether for each model $R \cup S$ of P, there exists a $R \cup W$ with $((R \cup W) \cap A) = ((R \cup S) \cap A) = R$, such that $R \cup W$ models Q, i.e., the property to be tested. This yields $O(2^k \times 2^k) = O(2^{k+1})$ Horn-derivability tests. The same procedure is done the other direction, i.e., exchanging P and Q. Whenever k is fixed, this gives us a polynomial time algorithm. (More efficient algorithms may be given, but we do not focus on this here.) □

7 Language Variations

In this section, we briefly address how our results apply to variations of the language of logic programs. First, we consider modifications within the case of propositional programs, and then discuss the general DATALOG case.

7.1 Extensions in the Propositional Case

Adding Classical Negation. 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.

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 17 *Consider the extended logic programs* $P = \{a \vee b \leftarrow;\ \neg a \leftarrow a;\ \neg b \leftarrow b\}$ *and* $Q = \{a \leftarrow \mathit{not}\, b;\ b \leftarrow \mathit{not}\, a;\ \neg a \leftarrow a;\ \neg b \leftarrow b\}$. *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 [56, 37, 57] made no distinction between no answer set and inconsistent answer set, in [17] we adapted the definition of SE-models accordingly and got more general characterizations in terms of so-called SEE-models for extended programs. Many results easily carry over to the extended case: E.g., for positive programs, uniform and strong equivalence coincide also in this case and, 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^P_2-hard (resp. coNP-hard).

However, not all properties do carry over. As Example 17 reveals, in general a head-cycle free extended DLP P is no longer equivalent, and hence not uniformly equivalent, to its shift variant $P^{\leftarrow}$ (see [17] for a characterization of head-cycle and contradiction free programs for which this equivalence holds).

We expect a similar picture for relativized equivalences of extended logic programs but adapting corresponding proofs is still subject of future work.

Disallowing Constraints. Sometimes, it is desirable to consider constraints just as abbreviations, in order to have core programs which are definite, i.e., without constraints. The most direct approach is to replace each constraint $\leftarrow B$ by $w \leftarrow B, not\, w$; where w is a designated atom not occurring in the original program. Obviously, this does not influence ordinary equivalence tests, but for notions as uniform and strong equivalence some more care is required. Take the strongly equivalent programs $P = \{a \leftarrow not\, a\}$ and $Q = \{\leftarrow not\, a\}$. By above rewriting Q becomes $Q' = \{w \leftarrow not\, a, not\, w\}$. Then, $(\cdot, w) \notin SE(P)$ but $(\cdot, w) \in SE(Q')$. Hence, this rewriting is not sensitive under strong equivalence. However, if we disallow w to appear in possible extensions, i.e., employing $\equiv_s^A$ instead of $\equiv_s$ we can circumvent this problem. Simply take $A = U \setminus \{w\}$ where U is the universe of atoms. Observe that this employs bounded relativization, and in the light of Theorem 35 this workaround does not result in a more complex problem. For uniform equivalence the methodology can be applied in the same manner.

However, this approach requires (unstratified) negation. If we want to get rid off constraints for comparing positive programs, an alternative method is to use a designated (spoiled) answer set to indicate that the original program had no answer set. The idea is to replace each constraint $\leftarrow B$ by $w \leftarrow B$, where w is a designated atom as above; additionally we add the collection of rules $v \leftarrow w$ for each atom v of the universe to both programs (even if no constraint is present). This rewriting retains any equivalence notion, even if w is allowed to occur in the extensions.

The problem of comparing, say, a positive program P (with constraints) and a normal program Q is more subtle, if we require to replace the constraints in P by positive rules themselves. We leave this for further study, but refer to some results in [20], which suggest that these settings may not be solved in an easy manner. To wit, [20] reports that the complexity for some problems of the form "Given a program P from class C; does there exist a program Q from class C', such that $P \equiv_e Q$?" differs with respect to allowing constraints.

Using Nested Expressions. Programs with nested expressions [38] (also called nested logic programs) extend DLPs in such a way that arbitrarily nested formulas, formed from literals using negation as failure, conjunction, and disjunction, constitute the heads and bodies of rules. Our characterizations for uniform equivalence are well suited for this class as was shown in [49]. Since the proofs of our main results are generic in the use of reducts, we expect that all results (including relativized notions of equivalence) can be carried over to nested logic programs without any problems. Note however, that the concrete definitions for subclasses (positive, normal, etc.) have to be extended in the context of nested logic programs (see [42] for such an extension of head-cycle free programs). It remains for further work to apply our results to such classes.

7.2 DATALOG programs

The results in the previous sections on propositional logic programs provide an extensive basis for studying equivalences of DATALOG programs if, as usual, their semantics is given in terms of propositional programs. Basic notions and concepts for strong and uniform equivalence such as SE-models, UE-models, and the respective notions of consequence generalize naturally to this setting, using Herbrand interpretations over a relational alphabet and a set of constants in the usual way (see [13]). Furthermore, fundamental results can be lifted to DATALOG programs by reduction to the propositional case. In particular, the elementary characterizations $P \equiv_e Q$ iff $M_e(P) = M_e(Q)$ iff $P \models_e Q$ and $Q \models_e P$ carry over to the DATALOG setting for $e \in \{s, u\}$ and $M_s(\cdot) = SE(\cdot)$, respectively $M_u(\cdot) = UE(\cdot)$ (see also [13]). However, a detailed analysis of the DATALOG case including relativized notions of equivalence is subject of ongoing work.

Nevertheless, let us 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, resp. model checking, increases by an exponential. Intuitively, this is explained by the exponential size of a Herbrand interpretation, i.e., the ground instance of a program over the universe. Note that [40] reported (without proof) that checking strong equivalence for programs in this setting is in coNP, and thus would have the same complexity as in the propositional case; however, for arbitrary programs, this is not correct. Unsurprisingly, over *infinite* domains, in the light of the results in [53, 27], decidability of equivalence and inference problems for DATALOG programs is no longer guaranteed. While strong equivalence and SE-inference remain decidable (more precisely complete for co-NEXPTIME), this is not the case for uniform equivalence (respectively inference) in general. For positive programs, however, the two notions coincide and are decidable (more precisely complete for co-NEXPTIME); see [13] for details. It remains as an issue for future work to explore the decidability versus undecidability frontier for classes of DATALOG programs, possibly under restrictions as in [27, 9].

8 Conclusion and Further Work

In this paper, we have extended the research about equivalence of nonmonotonic logic programs under answer set semantics, in order to simplify parts (or modules) of a program, without analyzing the entire program. Such local simplifications call for alternative notions of equivalence, since a simple comparison of the answer sets does not provide information whether a program part can be replaced by its simplification. To wit, by the non-monotonicity of the answer set semantics, two (ordinary) equivalent (parts of) programs may lead to different answer sets if they are used in the same global program R. Alternative notions of equivalence thus require that the answer sets of the two programs coincide under different R: strong equivalence [37], for instance, requires that the compared programs are equivalent under any extension R.

In this paper, we have considered further notions of equivalence, in which the actual form of R is syntactically constrained:

- Uniform equivalence of logic programs, which has been considered earlier for DATALOG and general Horn logic programs [52, 43]. Under answer set semantics uniform equivalence can be exploited for optimization of components in a logic program which is modularly structured.

- Relativized notions of both uniform and strong equivalence restrict the alphabet of the extensions. This allows to specify which atoms may occur in the extensions, and which do not. This notion of equivalence for answer set semantics was originally suggested by Lin in [40] but not further investigated. In practice, relativization is a natural concept, since it allows to specify internal atoms, which only occur in the compared program parts, but it is guaranteed that they do not occur anywhere else.

We have provided semantical characterizations of all these notions of equivalence by adopting the concept of SE-models [56] (equivalently, HT-models [37]), which capture the essence of a program with respect to strong equivalence. Furthermore, we have thoroughly analyzed the complexity of equivalence checking and related problems for the general case and several important fragments. This collection of results gives a valuable theoretical underpinning for advanced methods of program optimization and for enhanced ASP application development, as well as a potential basis for the development of ASP debugging tools.

Several issues remain for further work. One issue is a characterization of uniform equivalence in terms of "models" for arbitrary programs in the infinite case; as we have shown, no subset of SE-models serves this purpose. In particular, a notion of models which correspond to the UE-models in the case where the latter capture uniform equivalence would be interesting.

We focused here on the propositional case, to which general programs with variables reduce, and we just briefly mentioned a possible extension to a DATALOG setting [13]. Here, undecidability of uniform equivalence arises if negation may be present in programs. A thorough study of cases under which uniform equivalence and the other notions of equivalence are decidable is needed, along with complexity characterizations. Given that in addition to the syntactic conditions on propositional programs considered here, further ones involving predicates might be taken into account (cf. [9, 27]), quite a number of different cases remains to be analyzed.

Finally, an important issue is to explore the usage of uniform equivalence and relativized equivalence in program replacement and rewriting, and to develop optimization methods and tools for Answer Set Programming; a first step in this direction, picking up some of the results of this paper, has been made in [20]. However, much more remains to be done.

Acknowledgments.

The authors would like to thank David Pearce for interesting discussions and comments about this work and pointers to related literature, as well as Katsumi Inoue and Chiaki Sakama for their valuable comments on relativizing equivalence. We are also grateful to the anonymous reviewers of ICLP 2003 and JELIA 2004. Their comments on submissions preliminary to this article helped to improve this work, as well.

References

[1] J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. C. Przymusinski. Dynamic Updates of Non-Monotonic Knowledge Bases. *Journal of Logic Programming*, 45(1–3):43–70, 2000.

[2] C. Anger, K. Konczak, and T. Linke. `NoMoRe`: A System for Non-Monotonic Reasoning. In T. Eiter, W. Faber, and M. Truszczyński, editors, *Logic Programming and Nonmonotonic Reasoning — 6th International Conference, LPNMR'01, Vienna, Austria, September 2001, Proceedings*, number 2173 in Lecture Notes in AI (LNAI), pages 406–410. Springer Verlag, September 2001.

[3] D. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41:274–306, 1990.

[4] R. Ben-Eliyahu and R. Dechter. Propositional Semantics for Disjunctive Logic Programs. *Annals of Mathematics and Artificial Intelligence*, 12:53–87, 1994.

[5] W. Buntine. Generalised Subsumption and its Applications to Induction and Redundancy. *Artificial Intelligence*, 36(2):149–176, 1988.

[6] S. Buss. The boolean formula value problem is in alogtime. In *19th Annual ACM Symposium on Theory of Computing*, pages 123–131, 1987.

[7] P. Cabalar. A Three-Valued Characterization for Strong Equivalence of Logic Programs. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI 2002)*, pages 106–111. AAAI Press/MIT Press, 2002.

[8] M. Cadoli and M. Lenzerini. The Complexity of Propositional Closed World Reasoning and Circumscription. *Journal of Computer and System Sciences*, 48(2):255–310, Apr. 1994.

[9] S. Chaudhuri and M. Y. Vardi. On the Equivalence of Recursive and Nonrecursive Datalog Programs. In *Proceedings of the 11th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems (PODS 1992)*, pages 55–66. ACM, 1992.

[10] S. Cosmadakis and P. Kanellakis. Parallel Evaluation of Recursive Rule Queries. In *Proceedings of the 5th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems (PODS 1986)*, pages 280–293. ACM, 1986.

[11] D. 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.

[12] Y. Dimopoulos, B. Nebel, and J. Koehler. Encoding Planning Problems in Nonmonotonic Logic Programs. In S. Steel and R. Alami, editors, *Proceedings of the European Conference on Planning 1997 (ECP'97)*, volume 1348 of *LNCS*, pages 169–181. Springer Verlag, 1997.

[13] T. Eiter, W. Faber, M. Fink, G. Greco, D. Lembo, and H. Tompits. Methods and Techniques for Query Optimization. Technical Report IST 2001-33570 (INFOMIX), D5-3., European Commission, June 2004.

[14] T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. A Logic Programming Approach to Knowledge-State Planning, II: the DLV$^{\mathcal{K}}$ System. *Artificial Intelligence*, 144(1-2):157–211, 2003.

[15] T. Eiter, W. Faber, N. Leone, G. Pfeifer, and A. Polleres. A Logic Programming Approach to Knowledge-State Planning: Semantics and Complexity. *ACM Transactions on Computational Logic*, 5(2):206–263, 2004.

[16] T. Eiter and M. Fink. Uniform Equivalence of Logic Programs under the Stable Model Semantics. In C. Palamidessi, editor, *Proceedings 19th International Conference on Logic Programming (ICLP 2003)*, number 2916 in LNCS, pages 224–238. Springer Verlag, 2003.

[17] T. Eiter and M. Fink. Uniform Equivalence of Logic Programs under the Stable Model Semantics. Technical Report INFSYS RR-1843-03-08, Institut für Informationssysteme, Technische Universität Wien, Austria, May/September 2003.

[18] T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. A Framework for Declarative Update Specifications in Logic Programs. In B. Nebel, editor, *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 649–654. Morgan Kaufmann, 2001.

[19] T. Eiter, M. Fink, H. Tompits, and S. Woltran. On Eliminating Disjunctions in Stable Logic Programming. In D. Dubois, C. A. Welty, and M.-A. Williams, editors, *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 447–458. AAAI Press, 2004.

[20] T. Eiter, M. Fink, H. Tompits, and S. Woltran. Simplifying Logic Programs Under Uniform and Strong Equivalence. In V. Lifschitz and I. Niemelä, editors, *Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2004)*, volume 2923 of *LNCS*, pages 87–99. Springer Verlag, 2004.

[21] T. Eiter and G. Gottlob. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. *Annals of Mathematics and Artificial Intelligence*, 15(3/4):289–323, 1995.

[22] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive Datalog. *ACM Transactions on Database Systems*, 22(3):364–418, September 1997.

[23] E. Erdem, V. Lifschitz, L. Nakhleh, and D. Ringe. Reconstructing the Evolutionary History of Indo-European Languages Using Answer Set Programming. In V. Dahl and P. Wadler, editors, *Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages (PADL 2003), New Orleans, LA, USA, January 13-14, 2003*, volume 2562 of *Lecture Notes in Computer Science*, pages 160–176. Springer, 2003.

[24] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In R. Kowalski and K. Bowen, editors, *Proceedings of the Fifth International Conference on Logic Programming (ICLP 1988)*, pages 1070–1080. The MIT Press, 1988.

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

[26] P. Giorgini, F. Massacci, J. Mylopoulos, and N. Zannone. Requirements Engineering meets Trust Management: Model, Methodology, and Reasoning. In *Proceedings of the Second International Conference on Trust Management (iTrust 2004)*, volume 2995 of *LNCS*, pages 176–190. Springer Verlag, 2004.

[27] A. Y. Halevy, I. S. Mumick, Y. Sagiv, and O. Shmueli. Static Analysis in Datalog Extensions. *Journal of the ACM*, 48(5):971–1012, 2001.

[28] K. Heljanko and I. Niemelä. Bounded LTL Model Checking with Stable Models. In T. Eiter, W. Faber, and M. Truszczynski, editors, *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'01)*, volume 2173 of *LNCS*, pages 200–212. Springer Verlag, 2001.

[29] N. Immerman. *Descriptive Complexity*. Springer, 1999.

[30] K. Inoue and C. Sakama. Updating Extended Logic Programs through Abduction. In M. Gelfond, N. Leone, and G. Pfeifer, editors, *Proceedings of the 5th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'99)*, volume 1730 of *LNCS*, pages 147–161. Springer Verlag, 1999.

[31] K. Inoue and C. Sakama. Equivalence of Logic Programs under Updates. In J. J. Alferes and J. A. Leite, editors, *Logics in Artificial Intelligence, 9th European Conference, JELIA 2004, Proceedings*, volume 3229 of *LNCS*, pages 174–186. Springer Verlag, 2004.

[32] T. Janhunen and E. Oikarinen. LPEQ and DLPEQ - Translators for Automated Equivalence Testing of Logic Programs. In V. Lifschitz and I. Niemelä, editors, *Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-04)*, volume 2923 of *LNCS*, pages 336–340. Springer Verlag, 2004.

[33] H. Kautz and B. Selman. Planning as Satisfiability. In *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI '92)*, pages 359–363, 1992.

[34] V. Kowalski. The Calculus of the Weak "Law of Excluded Middle". *Mathematics of the USSR*, 8:648–658, 1968.

[35] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV System for Knowledge Representation and Reasoning. Technical Report cs.AI/0211004, arXiv.org, Nov. 2002. To appear in ACM TOCL.

[36] V. Lifschitz. Action Languages, Answer Sets and Planning. In K. Apt, V. W. Marek, M. Truszczyński, and D. S. Warren, editors, *The Logic Programming Paradigm – A 25-Year Perspective*, pages 357–373. Springer Verlag, 1999.

[37] V. Lifschitz, D. Pearce, and A. Valverde. Strongly Equivalent Logic Programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.

[38] V. Lifschitz, L. Tang, and H. Turner. Nested Expressions in Logic Programs. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):369–389, 1999.

[39] V. Lifschitz and H. Turner. Splitting a Logic Program. In P. Van Hentenryck, editor, *Proceedings of the Eleventh International Conference on Logic Programming (ICLP 1994)*, pages 23–38, Santa Margherita Ligure, Italy, June 1994. MIT-Press.

[40] F. Lin. Reducing Strong Equivalence of Logic Programs to Entailment in Classical Propositional Logic. In D. Fensel, F. Giunchiglia, D. McGuinness, and M.-A. Williams, editors, *Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning (KR 2002)*, pages 170–176. Morgan Kaufmann, 2002.

[41] 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.

[42] T. Linke, H. Tompits, and S. Woltran. On Acyclic and Head-Cycle Free Nested Logic Programs. In B. Demoen and V. Lifschitz, editors, *Proceedings of the 20th International Conference on Logic Programming (ICLP-04)*, volume 3132 of *LNCS*, pages 225–239. Springer-Verlag, 2004.

[43] M. J. Maher. Equivalences of Logic Programs. In Minker [44], pages 627–658.

[44] J. Minker, editor. *Foundations of Deductive Databases and Logic Programming*. Morgan Kaufmann Publishers, Inc., Washington DC, 1988.

[45] 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 Verlag, 2001.

[46] D. Pearce. Simplifying Logic Programs under Answer Set Semantics. In B. Demoen and V. Lifschitz, editors, *Proceedings of the 20th International Conference on Logic Programming (ICLP 2004)*, volume 3132 of *LNCS*. Springer Verlag, 2004.

[47] D. Pearce, H. Tompits, and S. Woltran. Encodings for Equilibrium Logic and Logic Programs with Nested Expressions. In P. Brazdil and A. Jorge, editors, *Progress in Artificial Intelligence, Knowledge Extraction, Multi-agent Systems, Logic Programming and Constraint Solving, 10th Portuguese Conference on Artificial Intelligence (EPIA 2001)*, volume 2258 of *LNCS*, pages 306–320. Springer Verlag, 2001.

[48] D. Pearce and A. Valverde. Synonymous Theories in Answer Set Programming and Equilibrium Logic. In R. L. de Mántaras and L. Saitta, editors, *Proceedings of the 16th Eureopean Conference on Artificial Intelligence (ECAI 2004)*, pages 388–392. IOS Press, 2004.

[49] D. Pearce and A. Valverde. Uniform Equivalence for Equilibrium Logic and Logic Programs. In V. Lifschitz and I. Niemelä, editors, *Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2004)*, volume 2923 of *LNCS*, pages 194–206. Springer Verlag, 2004.

[50] A. Provetti and T. C. Son, editors. *Proceedings AAAI 2001 Spring Symposium on Answer Set Programming: Towards Efficient and Scalable Knowledge Representation and Reasoning*, Stanford, CA, March 2001. AAAI Press.

[51] T. Przymusinski. Stable Semantics for Disjunctive Programs. *New Generation Computing Journal*, 9:401–424, 1991.

[52] Y. Sagiv. Optimizing Datalog Programs. In Minker [44], pages 659–698.

[53] O. Shmueli. Equivalence of Datalog Queries is Undecidable. *Journal of Logic Programming*, 15(3):231–242, Feb. 1993.

[54] P. Simons, I. Niemelä, and T. Soininen. Extending and Implementing the Stable Model Semantics. *Artificial Intelligence*, 138:181–234, June 2002.

[55] V. S. Subrahmanian and C. Zaniolo. Relating Stable Models and AI Planning Domains. In L. Sterling, editor, *Proceedings of the 12th International Conference on Logic Programming (ICLP'95)*, pages 233–247. MIT Press, 1995.

[56] H. Turner. Strong Equivalence for Logic Programs and Default Theories (Made Easy). In T. Eiter, W. Faber, and M. Truszczynski, editors, *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2001)*, volume 2173 of *LNCS*, pages 81–92. Springer Verlag, 2001.

[57] H. Turner. Strong Equivalence Made Easy: Nested Expressions and Weight Constraints. *Theory and Practice of Logic Programming*, 3(4-5):602–622, 2003.

[58] S. Woltran. Characterizations for Relativized Notions of Equivalence in Answer Set Programming. In J. J. Alferes and J. A. Leite, editors, *Logics in Artificial Intelligence, 9th European Conference, JELIA 2004, Proceedings*, volume 3229 of *LNCS*, pages 161–173. Springer Verlag, 2004.

[59] Y. Zhang and N. Y. Foo. Updating Logic Programs. In H. Prade, editor, *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98)*, pages 403–407. Wiley, 1998.

A Proofs

A.1 Proof of Lemma 5

For the only-if direction, suppose $P \equiv_u^A Q$. If (Y,Y) is neither A-SE-model of P, nor of Q, then (X,Y) is not an A-SE-model of any of the programs P and Q. Without loss of generality, assume $(Y,Y) \in SE^A(P)$ and $(Y,Y) \notin SE^A(Q)$. Let $F = (Y \cap A)$. We have the following situation by definition of A-SE-models. First, from $(Y,Y) \in SE^A(P)$, we get $Y \models P$. Hence, $Y \models P \cup F$. Second, $(Y,Y) \in SE^A(P)$ implies that for each $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$, $Y' \not\models P^Y$. Hence, for each such Y', $Y' \not\models (P \cup F)^Y$. Finally, for each $X \subset Y$ with $(X \cap A) \subset (Y \cap A)$, $X \not\models F$ and thus $X \not\models P^Y \cup F$. To summarize, we arrive at $Y \in \mathcal{AS}(P \cup F)$. On the other hand, $Y \notin \mathcal{AS}(Q \cup F)$. This can be seen as follows. By $(Y,Y) \notin SE^A(Q)$, either $Y \not\models Q$ or there exists an $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$, such that $Y' \models Q^Y$. But then, $Y' \models (Q \cup F)^Y$. Hence, this contradicts our assumption $P \equiv_u^A Q$, since F is a set of facts over A. Item (i) must hold.

To show (ii), assume first that (X,Y) is an A-SE-model of P but not of Q. In view of (i), it is clear that $X \subset Y$ must hold. Moreover, $X \subseteq A$. Suppose now that for every set X', $X \subset X' \subset Y$, it holds that (X',Y) is not an A-SE-model of Q. Then, since no subset of X models $Q^Y \cup X$, (Y,Y) is the only A-SE-model of $Q \cup X$ of form $(\cdot, Y)$. Thus, $Y \in \mathcal{AS}(Q \cup X)$ in this case, while $Y \notin \mathcal{AS}(P \cup X)$. This is seen as follows: Since $(X,Y) \in SE^A(P)$, there exists an $X' \subseteq Y$ with $(X' \cap A) = (X \cap A)$, such that $X' \models P^Y$. Moreover, $X' \models (P \cup X)^Y$. Thus, $Y \notin \mathcal{AS}(P \cup X)$. This contradicts $P \equiv_u^A Q$, since $X \subseteq A$. Thus, it follows that for some M such that $X \subset M \subset Y$, (M,Y) is an A-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 A-SE-interpretation (X,Y) which is an A-SE-model of exactly one of P and Q. Suppose that there exist sets of atoms $F \subseteq A$ and Z, such that w.l.o.g., $Z \in \mathcal{AS}(P \cup F)$, but $Z \notin \mathcal{AS}(Q \cup F)$. Since $Z \in \mathcal{AS}(P \cup F)$, we have that $F \subseteq Z$, $Z \models P$, and, for each $Z' \subset Z$ with $(Z' \cap A) = (Z \cap A)$, $Z' \not\models P^Z$. Consequently, (Z,Z) is an A-SE-model of P. Since $Z \notin \mathcal{AS}(Q \cup F)$, either $Z \not\models (Q \cup F)$, or there exists a $Z' \subset Z$ such that $Z' \models (Q \cup F)^Z$.

Let us first assume $Z \not\models (Q \cup F)$. However, since $F \subseteq Z$, we get $Z \not\models Q$. We immediately get $(Z,Z) \notin SE^A(Q)$, i.e., (Z,Z) violates (i). It follows that $Z \models (Q \cup F)$ must hold, and that there must exist a $Z' \subset Z$ such that $Z' \models (Q \cup F)^Z = Q^Z \cup F$. We have two cases: If $(Z' \cap A) = (Z \cap A)$, then, by definition of A-SE-models, $(Z,Z) \notin SE^A(Q)$, as well. Hence, the following relations hold $Z \models Q$; for each Z' with $(Z' \cap A) = (Z \cap A)$, $Z' \not\models Q^Z$, and there exists an Z'' with $(Z'' \cap A) \subset (Z \cap A)$, such that $Z'' \models Q^Z$. We immediately get that $((Z'' \cap A), Z) \in SE^A(Q)$. But $(Z'', Z) \notin SE^A(P)$. To see the latter, note that $F \subseteq Z$ must hold. So, if $((Z'' \cap A), Z)$ were an A-SE-model of P, then it would also be an A-SE-model of $P \cup F$, contradicting the assumption that $Z \in \mathcal{AS}(P \cup F)$. Again we get an A-SE-model, $((Z'' \cap A), Z)$, of exactly one of the programs, Q in this case. Hence, according to (ii), there exists an A-SE-model (M,Z) of P, $Z'' \subset M \subset Z$. However, because of $F \subseteq Z$, it follows that (M,Z) is also an A-SE-model of $P \cup F$, contradicting our assumption that $Z \in \mathcal{AS}(P \cup F)$.

This proves that, given (i) and (ii) for every A-SE-model (X, Y) such that (X, Y) is an A-SE-model of exactly one of P and Q, no sets of atoms $F \subseteq A$ and Z exists such that Z is an answer set of exactly one of $P \cup F$ and $Q \cup F$. That is, $P \equiv_u^A Q$ holds. □

A.2 Proof of Theorem 11

For (a), by Theorem 10, $P \equiv_u^A Q$ implies $UE^A(P) = UE^A(Q)$. Each A-UE-model of a program is, by definition, an A-SE-model of that program. We immediately get $UE^A(P) = UE^A(Q) \subseteq SE^A(Q)$ and $UE^A(Q) = UE^A(P) \subseteq SE^A(P)$.

For (b), suppose $P \not\equiv_u^A Q$, and either P, Q, or A is finite. By Theorem 10 we have $UE^A(P) \neq UE(Q)$. Wlog, assume interpretations X, Y, such that $(X, Y) \in UE^A(P)$ and $(X, Y) \notin UE^A(Q)$. We have two cases: If $(X, Y) \notin SE^A(Q)$, we are done, since then $UE^A(P) \subseteq SE^A(Q)$ cannot hold. If $(X, Y) \in SE^A(Q)$, this implies existence of an X' with $X \subset X' \subset Y$, such that $(X', Y) \in UE^A(Q)$. However, since $(X, Y) \in UE^A(P)$, for each such X', $(X', Y) \notin SE^A(P)$. Hence, $UE^A(Q) \subseteq SE^A(P)$ cannot hold. □

A.3 Proof of Theorem 23

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 18 be verified with a call to a NP-oracle resp. in polynomial time. In case of a positive P, by Theorem 6, $P \models_u r$ 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 20. For the program Q constructed there, it holds $Q \models_u a \leftarrow$ if and only if none of the SE-models $\{(\sigma_X(J), \sigma_X(J) \cup \mathcal{A})\}$ of Q is an UE-model of Q as well, i.e., 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 21: the positive program P constructed there satisfies, by Theorem 21, $P \models_u a \leftarrow$ if and only if $P \equiv_u Q$ holds, which is equivalent to unsatisfiability of the CNF F there. Since P is HCF we can, as in the proof of Theorem 22, again use $P^{\rightarrow}$ and Theorem 15 in order to show coNP-hardness for head-cycle free (non-positive) programs. □

A.4 Proof of Theorem 24

First consider P is HCF. Then, coNP-membership of $P \models_u Q$ is an immediate consequence of the result in Theorem 23 by testing $P \models_u r$, for each $r \in Q$. Since the class coNP is closed under conjunction, coNP-membership for $P \models_u Q$ follows.

Next, suppose Q is HCF. We first show the claim for normal Q, using the complementary problem $P \not\models_u Q$. By inspecting the characterizations of uniform equivalence, $P \not\models_u Q$ iff (i) $P \not\models Q$, or (ii) there exists an SE-model (X, Y) of P, such that no (X', Y) with $X \subseteq X' \subset Y$ is SE-model of Q. Test (i) is obviously in NP. For containment in NP of Test (ii), we argue as follows: We guess a pair (X, Y) and check in polynomial time whether it is SE-model of P. In order to check that no (X', Y) with $X \subseteq X' \subset Y$ is SE-model of Q we test unsatisfiability of the program $Q^Y \cup X \cup Y_{\subset}$, which is Horn, whenever Q is normal. Therefore, this test is is feasible in polynomial time. Hence, $P \models_u Q$ is in coNP for normal Q. Recall that for a HCF program Q, we have $Q \equiv_u Q^{\rightarrow}$. This implies that $P \not\models_u Q^{\rightarrow}$ iff $P \not\models_u Q$. Therefore, the claim holds for HCF programs as well.

We proceed with the matching lower bound. Let P and Q as in the proof of Theorem 21, then $P^{\rightarrow}$ is normal, Q is Horn, and $P \models_u Q$ iff $P^{\rightarrow} \models_u Q$ iff $P^{\rightarrow} \models_u a \leftarrow$, which is coNP-hard. □

A.5 Proof of Theorem 30

Membership is due to Theorem 28.

The hardness part is by a similar construction as above, i.e., consider a QBF of the form $\exists X \forall Y \phi$ with $\phi = \bigvee_{i=1}^{n} D_i$ a DNF. We take here the following programs, viz.

$$
\begin{aligned}
P \quad = \quad & \{x \vee \bar{x} \leftarrow;\ \leftarrow x, \bar{x} \mid x \in X\} \cup \\
& \{y \vee \bar{y} \leftarrow;\ y \leftarrow a;\ \bar{y} \leftarrow a;\ a \leftarrow y, \bar{y} \mid y \in Y\} \cup \\
& \{a \leftarrow D_i^* \mid 1 \leq i \leq n\}
\end{aligned}
$$

which is the same program as above, but without $\leftarrow \mathit{not}\, a$, and thus positive. For the second program take

$$
\begin{aligned}
Q \quad = \quad & \{x \vee \bar{x} \leftarrow;\ \leftarrow x, \bar{x} \mid x \in X\} \cup \\
& \{y \vee \bar{y} \leftarrow;\ \leftarrow y, \bar{y} \mid y \in Y\} \cup \\
& \{\leftarrow D_i^* \mid 1 \leq i \leq n\} \cup \\
& \{\leftarrow a\}.
\end{aligned}
$$

We start computing the SE-models of the two programs. Let, for any $J \subseteq X$,

$$
M[J] \quad = \quad \sigma_X(J) \cup Y \cup \bar{Y} \cup \{a\},
$$

and suppose $A \subseteq X \cup \bar{X}$. The set of classical models of P is given by $\{M[J] \mid J \subseteq X\}$ and $\sigma(J \cup I)$, for each $I \subseteq Y$, such that ϕ is false under $J \cup I$. Thus, we get:

$$
\begin{aligned}
SE(P) \quad = \quad & \{(\sigma(J \cup I), \sigma(J \cup I)),\ \ (\sigma(J \cup I), M[J]) \mid J \subseteq X, I \subseteq Y : J \cup I \not\models \phi\} \cup \\
& \{(M[J], M[J]) \mid J \subseteq X\}; \\
SE(Q) \quad = \quad & \{(\sigma(J \cup I), \sigma(J \cup I)) \mid J \subseteq X, I \subseteq Y : J \cup I \not\models \phi\}.
\end{aligned}
$$

First, each pair $(\sigma(J \cup I), \sigma(J \cup I)) \in SE(P)$ is A-SE-model of both, P and Q. Second, P possesses additional A-SE-models, if there exists at least one $J \subseteq X$ with $(M[J], M[J]) \in SE^A(P)$. This is the case, if no $I \subseteq Y$ makes ϕ false under $J \cup I$, i.e., if the QBF $\exists X \forall Y \phi$ is true. This shows Σ_2^P-hardness of deciding $P \not\equiv_s^A Q$ with P and Q positive. Consequently, $P \equiv_s^A Q$ under this setting is Π_2^P-hard. Note that since the argumentation holds also for $card(A) < 2$, we captured both $\equiv_s^A$ and $\equiv_u^A$.

It remains to show Π_2^P-hardness for $P \equiv_e^A Q$, for the case where P is positive and Q is normal, $e \in \{s, u\}$. As a consequence of Corollary 6 (see also Example 16), for a disjunctive rule $r = v \vee w \leftarrow$, $Q \equiv_s^A Q_r^{\rightarrow}$ holds for any A ,whenever $\leftarrow v, w \in Q$. Hence, we can shift each disjunctive rule in Q and get $Q \equiv_s^A Q^{\rightarrow}$. This shows Π_2^P-hardness for $P \equiv_s^A Q$, for the case where P is positive and Q is normal. Again, we immediately get the respective result for $P \equiv_u^A Q$, since the argumentation holds also for $card(A) < 2$. □

A.6 Proof of Theorem 32

It remains to show coNP-membership for two cases, viz. (i) $P \equiv_e^A Q$ with P positive and Q Horn; and $P \equiv_s^A Q$ with P HCF and Q Horn. Therefore, we first show the following additional result:

Lemma 13 *For positive programs P, Q, and a set of atoms A, $P \equiv_e^A Q$ holds iff (i) each A-minimal model of P is a classical model of Q; and (ii) for each interpretation Y, $Y \models Q$ implies existence of a $Y' \subseteq Y$ with $(Y' \cap A) = (Y \cap A)$, such that $Y' \models P$.*

Proof. For the only-if direction, first suppose (i) does not hold. It is easily seen, that then the A-minimal models cannot coincide, and thus $P \not\equiv_e^A Q$. So suppose (ii) does not hold; i.e., there exists an interpretation Y, such that $Y \models Q$ but no $Y' \subseteq Y$ with $(Y' \cap A) = (Y \cap A)$ is a model of P. Again, the A-minimal models of P and Q cannot coincide.

For the if direction, suppose $P \not\equiv_e^A Q$. First let Y be A-minimal for P but not for Q. If $Y \not\models Q$ we are done, since (i) is violated. Otherwise, there exists a $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$ which is a model of Q but not a classical model of P; (ii) is violated. Second, suppose there exists an Y which is A-minimal for Q but not for P. If, each $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$ does not model P, (ii) is violated. Otherwise, if there exists a $Y' \subset Y$ with $(Y' \cap A) = (Y \cap A)$ and $Y' \models P$, then there exists a Y'' with $(Y'' \cap A) = (Y \cap A)$, which is A-minimal for P but not a classical model of Q; whence (i) is violated. □

We proceed by proving (i) and (ii).

(i): Since both P and Q are positive, $e = s$ and $e = u$ are the same concepts. coNP-membership is obtained by applying Theorem 13. In fact, this result suggests the following algorithm:

1. Check whether each A-minimal model of Q is model of P;
2. Check whether, for each model Y of P, there exists a $Y' \subseteq Y$ with $(Y' \cap A) = Y$ being model of Q.

We show that, for both steps, the complementary problem is in NP. For Step 1, we guess a Y and check whether it is A-minimal for Q but not a classical model of P. The latter test is feasible in polynomial time. The former reduces to test unsatisfiability of the Horn theory $Q \cup (Y \cap A) \cup Y_{\subset}$. For the second step the argumentation is similar. Again, we guess an interpretation Y, check whether it is a model of P, and additionally, whether all $Y' \subseteq Y$ with $Y' \cap A = Y$ are not model of Q. The latter reduces to test unsatisfiability of the Horn program $Q \cup (Y \cap A) \cup Y_{\subseteq}$.

(ii) In this setting, coNP-membership is obtained by the following algorithm:

1. Check whether the total A-SE-models of P and Q coincide;
2. Check whether, for each $X \subset Y$, $(X, Y) \in SE^A(P)$ implies $(X, Y) \in SE^A(Q)$.

The correctness of this procedure is a consequence of Proposition 10, i.e., that $SE^A(Q) \subseteq SE^A(P)$ holds for positive Q, whenever the total A-SE-models of P and Q coincide. Since Q is Horn and thus positive, it is sufficient to check $SE^A(P) \subseteq SE^A(Q)$ which is accomplished by Step 2, indeed. The first step is clearly in coNP, since for total A-SE-interpretations, A-SE-model checking and A-UE-model checking is the same task. By Theorem 25, A-UE-model checking is polynomial for HCF programs. For the second step, we show NP-membership for the complementary task. We guess some X' and Y, and test whether $(Y, Y) \in SE^A(P)$, $X' \models P^Y$, and $((X' \cap A), Y) \notin SE^A(Q)$. All tests are feasible in polynomial time and imply that $(X, Y) \in SE^A(P)$ but $(X, Y) \notin SE^A(Q)$, with $X = (X' \cap A)$. □

A.7 Proof of Theorem 33

Membership has already been obtained in Theorem 32. For the hardness-part we reduce UNSAT to $P \equiv_e^A Q$, where P and Q are Horn. The case of definite programs is discussed below.[3] Hence, let $F = \bigwedge_{i=1}^{n} c_{i,1} \vee \cdots \vee c_{i,n_i}$ be given over atoms V and consider $G = \{g_1, \ldots, g_n\}$ as new atoms. Define

$$P = \{\leftarrow v, \bar{v} \mid v \in V\} \cup \{g_i \leftarrow c^*_{i,j} \mid 1 \leq i \leq n; 1 \leq j \leq n_i\};$$

and let $A = V \cup \bar{V}$. Then, F is unsatisfiable iff

$$P \quad \equiv_e^A \quad (P \cup \{\leftarrow g_1, \ldots, g_n\})$$

with $e \in \{s, u\}$. We show the claim for $e = s$. Recall that RSE and RUE are the same for Horn programs. For the only-if direction suppose F is unsatisfiable. Then, there does not exist an interpretation $I \subseteq V$, such that $\sigma(I) \cup G$ is A-minimal for P. To wit, there exists at least a $G' \subseteq G$ such that $\sigma(I) \cup G'$ is model of P as well. It is easily verified that under these conditions, $P \equiv_e^A P \cup \{\leftarrow g_1, \ldots, g_n\}$ holds. On the other hand, if F is satisfiable, there exists an interpretation $I \subseteq V$ such that $\sigma(I) \cup G$ is an A-minimal model of P. However, $\sigma(I) \cup G$ is not a model of $P \cup \{\leftarrow g_1, \ldots, g_n\}$. This proves the claim.

We show that coNP-hardness holds also for definite programs. Therefore, we introduce further atoms a, b and change P to

$$P = \{a \leftarrow v, \bar{v} \mid v \in V\} \cup \{g_i \leftarrow c^*_{i,j} \mid 1 \leq i \leq n; 1 \leq j \leq n_i\} \cup \{u \leftarrow a \mid u \in \mathcal{A}\}$$

where $\mathcal{A} = \{b\} \cup V \cup \bar{V} \cup G$. Then, F is unsatisfiable iff

$$P \equiv_e^A P \cup \{b \leftarrow g_1, \ldots, g_n\}$$

with $A = \{a, b\} \cup V \cup \bar{V}$. The correctness of the claim is by analogous arguments as above. □

[3] Our proof closely follows concepts used in [8] to establish coNP-hardness results for closed world reasoning over Horn theories.