# Facts do not Cease to Exist Because They are Ignored: Relativised Uniform Equivalence with Answer-Set Projection[*]

Johannes Oetsch, Hans Tompits, and Stefan Woltran

Institut für Informationssysteme 184/3, Technische Universität Wien,
Favoritenstraße 9-11, A-1040 Vienna, Austria
{oetsch,tompits,stefan}@kr.tuwien.ac.at

**Abstract.** Recent research in answer-set programming (ASP) focuses on different notions of equivalence between programs which are relevant for program optimisation and modular programming. Prominent among these notions is *uniform equivalence*, which checks whether two programs have the same semantics when joined with an arbitrary set of facts. In this paper, we study a family of more fine-grained versions of uniform equivalence, where the alphabet of the added facts as well as the projection of answer sets is taken into account. The latter feature, in particular, allows the removal of auxiliary atoms in computation, which is important for practical programming aspects. We introduce novel semantic characterisations for the equivalence problems under consideration and analyse the computational complexity for checking these problems. We furthermore provide efficient reductions to quantified propositional logic, yielding a rapid-prototyping system for equivalence checking.

## 1 Introduction

An important issue in software development is to determine whether two encodings of a given problem are equivalent, i.e., whether they compute the same result on a given problem instance. Although the question is well known to be undecidable for Turing-complete programming languages, there are important KR programming languages where it is decidable. Our object of investigation is one such language, viz. the class of *disjunctive logic programs* (DLPs) *under the answer-set semantics* [1]. This formalism constitutes the arguably most important instance of the *answer-set programming* (ASP) *paradigm*, whose popularity rests not only on the availability of efficient solvers but also on the ease of modeling problems. The characteristic feature of ASP is that solutions to problems are given by the models (the "answer sets") of their encodings and not by proofs as in traditional logic-based formalisms.

Given the nonmonotonic nature of DLPs under the answer-set semantics, a standard equivalence notion in the sense that two programs are viewed as being equivalent if they have the same answer sets is too weak to yield a replacement property like in classical logic. That is to say, given a program $R$ along with some subprogram $P \subseteq R$, when replacing $P$ with an equivalent program $Q$ it is not guaranteed that $Q \cup (R \setminus P)$ is

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

equivalent to $R$. This led to the introduction of stricter notions of equivalence, viz. *strong equivalence* [2] and *uniform equivalence* [3].

While strong equivalence in effect amounts to a replacement property by definition, uniform equivalence checks whether two programs have the same answer sets for any arbitrary *input*, i.e., for any set of facts. In more formal terms, two programs $P, Q$ are strongly equivalent iff for any program $R$ (the "context program"), $P \cup R$ and $Q \cup R$ have the same answer sets, and $P$ and $Q$ are uniformly equivalent iff the former condition holds for any set $R$ of facts. While strong equivalence is relevant for program optimisation and modular programming in general [4–6], uniform equivalence is useful in the context of hierarchically structured program components, where lower-layered components provide input for higher-layered ones.[1] Strong and uniform equivalence are, however, too restrictive in the sense that standard programming techniques like the use of local (auxiliary) variables, which may occur in some subprograms but which are ignored in the final computation, are not taken into account. In other words, these notions do not admit the *projection* of answer sets to a set of designated output letters.

In previous work, Eiter, Tompits, and Woltran [8] introduced a general framework for defining parameterised notions of program correspondence, allowing both answer-set projection as well as the specification which kind of context class should be used for program comparison. This framework thus generalises not only strong and uniform equivalence but also *relativised* versions thereof [9] (where "relativised" means that the alphabet of the context class is an additional parameter). In their analysis, Eiter, Tompits, and Woltran [8] focused on correspondence problems for propositional DLPs effectively generalising strong equivalence—in other words, they considered correspondence problems amounting to *relativised strong equivalence with projection*. In this paper, we complement these investigations by considering correspondence problems amounting to *relativised uniform equivalence with projection*. More formally, in such correspondence problems, there are fixed alphabets $A, B$ (i.e., sets of atoms) and it is checked whether, for programs $P, Q$ and any set $R \subseteq A$ of facts, the answer sets of $P \cup R$ and $Q \cup R$ projected to $B$ coincide. (In a relativised strong equivalence problem with projection, $R$ would be a program over $A$.) Like Eiter, Tompits, and Woltran [8], we also consider *inclusion problems*, i.e., checking set inclusion of the projected answer sets rather than equality. In such a setting, $Q$ can be viewed as an approximation of $P$ which is sound with respect to cautious reasoning from $P$. Note that since relativised strong equivalence (resp., inclusion) with projection implies relativised uniform equivalence (resp., inclusion) with projection (with respect to the same alphabets) but not vice versa, characterisations of the former kinds of problems in general do not capture the latter kinds of problems and so new methods are needed. Developing such characterisations is actually one of the main goals of this paper.

Taking a database point of view, where programs are seen as queries over databases, we refer to the equivalence problems studied here as *propositional query equivalence problems* (PQEPs) and to the considered inclusion problems as *propositional query inclusion problems* (PQIPs).

The main contributions of our paper can be summarised as follows:

[1] It is worth noting that uniform equivalence was first studied in the context of datalog programs as a decidable approximation of datalog equivalence [7].

– We introduce semantic characterisations for PQEPs and PQIPs in terms of novel semantic structures associated with each program. We have that a PQEP holds iff the associated structures coincide, and a PQIP holds iff the structures meet set inclusion. Interestingly, our characterisation differs from the well-known characterisation of (relativised) uniform equivalence in terms of (relativised) UE-models [3, 9] in case the projection set is unrestricted. Thus, as a by-product, we obtain a new characterisation of these special forms of equivalence.
– We analyse the computational complexity of checking PQEPs and PQIPs. While checking the kinds of correspondence problems analysed by Eiter, Tompits, and Woltran [8] is $\Pi_4^P$-complete in general, checking PQEPs or PQIPs is only $\Pi_3^P$-complete. Checking relativised strong or uniform equivalence is $\Pi_2^P$-complete [9], thus projection adds a source of complexity, providing the polynomial hierarchy does not collapse. Hence, under this proviso, the famous quote "facts do not cease to exist because they are ignored" [10] is evidenced here.
– We provide efficient reductions of PQEPs and PQIPs into quantified propositional logic. Given the availability of off-the-shelf solvers for the latter language, we thus can employ these as back-end inference engines for checking PQEPs and PQIPs. In fact, we incorporated our translations into the system $\mathsf{cc}\top$ [11] which was developed as an implementation for checking the kinds of correspondence problems studied by Eiter, Tompits, and Woltran [8].

## 2 Background

We are concerned with *propositional disjunctive logic programs* (DLPs) which are finite sets of rules of form

$$a_1 \vee \dots \vee a_l \leftarrow a_{l+1}, \dots, a_m, \mathrm{not}\ a_{m+1}, \dots, \mathrm{not}\ a_n, \tag{1}$$

where $n \geq m \geq l \geq 0$, all $a_i$ are propositional atoms from some fixed universe $\mathcal{U}$, and "not" denotes *default negation*. Rules of form $a \leftarrow$ are *facts* and are usually written without the symbol "$\leftarrow$". We denote by $At(P)$ the set of all atoms occurring in a program $P$, and say that a program is *over* $A$ if $At(P) \subseteq A$. We use $\mathcal{P}_A$ to refer to the set of all programs over $A$.

By an interpretation we understand a set of atoms. A rule $r$ of form (1) is *true* under an interpretation $I$, symbolically $I \models r$, iff $\{a_1, \dots, a_l\} \cap I \neq \emptyset$ whenever $\{a_{l+1}, \dots, a_m\} \subseteq I$ and $\{a_{m+1}, \dots, a_n\} \cap I = \emptyset$. If $I \models r$ holds, then $I$ is also said to be a *model* of $r$. As well, $I$ is a model of a program $P$, symbolically $I \models P$, iff $I \models r$, for all $r \in P$. Following Gelfond and Lifschitz [1], an interpretation $I$ is an *answer set* of a program $P$ iff it is a minimal model of the *reduct* $P^I$, resulting from $P$ by (i) deleting all rules containing a default negated atom $\mathrm{not}\ a$ such that $a \in I$, and (ii) deleting all default negated atoms in the remaining rules. The collection of all answer sets of a program $P$ is denoted by $\mathcal{AS}(P)$.

We use the following notation in the sequel: For an interpretation $I$ and a set $\mathcal{S}$ of interpretations, $\mathcal{S}|_I$ is defined as $\{Y \cap I \mid Y \in \mathcal{S}\}$. For a singleton set $\mathcal{S} = \{Y\}$, we also write $Y|_I$ instead of $\mathcal{S}|_I$. Furthermore, for sets $\mathcal{S}, \mathcal{S}'$ of interpretations, an interpretation $B$, and $\odot \in \{\subseteq, =\}$, we define $\mathcal{S} \odot_B \mathcal{S}'$ iff $\mathcal{S}|_B \odot \mathcal{S}'|_B$.

Following Eiter, Tompits, and Woltran [8], a *correspondence problem* (*over* $\mathcal{U}$) is a quadruple $\Pi = (P, Q, \mathcal{C}, \rho)$, where $P, Q \in \mathcal{P}_\mathcal{U}$ are programs over $\mathcal{U}$, $\mathcal{C} \subseteq \mathcal{P}_\mathcal{U}$ is a class of programs (the *context class* of $\Pi$), and $\rho \subseteq 2^{2^\mathcal{U}} \times 2^{2^\mathcal{U}}$ is a binary relation over sets of interpretations. $\Pi$ is said to *hold* iff, for each program $R \in \mathcal{C}$, $(AS(P \cup R), AS(Q \cup R)) \in \rho$. By instantiating $\mathcal{C}$ and $\rho$, different equivalence notions from the literature can be expressed. In particular, the following relations hold: $P$ and $Q$ are strongly equivalent [2] iff $(P, Q, \mathcal{P}_\mathcal{U}, =_\mathcal{U})$ holds; $P$ and $Q$ are uniformly equivalent iff $(P, Q, 2^\mathcal{U}, =_\mathcal{U})$ holds; $P$ and $Q$ are strongly equivalent relative to $A$ [9], for $A \subseteq \mathcal{U}$, iff $(P, Q, \mathcal{P}_A, =_\mathcal{U})$ holds; and $P$ and $Q$ are uniformly equivalent relative to $A$ [9], for $A \subseteq \mathcal{U}$, iff $(P, Q, 2^A, =_\mathcal{U})$ holds.

We also make use of *quantified propositional logic*, an extension of classical propositional logic in which formulas are permitted to contain quantifications over propositional variables. Similar to predicate logic, $\exists$ and $\forall$ are used as symbols for existential and universal quantification, respectively. It is customary to refer to formulas of quantified propositional logic as *quantified Boolean formulas* (QBFs).

For a QBF of form $\mathsf{Q}p\,\Psi$, where $\mathsf{Q} \in \{\exists, \forall\}$, we call $\Psi$ the *scope* of $\mathsf{Q}p$. An occurrence of an atom $p$ is *free* in a QBF $\Phi$ if it does not occur in the scope of a quantifier $\mathsf{Q}p$ in $\Phi$. Given a finite set $P$ of atoms, $\mathsf{Q}P\,\Psi$ stands for any QBF $\mathsf{Q}p_1\mathsf{Q}p_2\ldots\mathsf{Q}p_n\Psi$ such that $P = \{p_1, \ldots, p_n\}$. Finally, $\Phi[p/\phi]$ denotes the result of replacing each free occurrence of an atom $p$ in $\Phi$ by a formula $\phi$.

For an interpretation $I$ and a QBF $\Phi$, the relation $I \models \Phi$ is defined analogously as in classical propositional logic, with the additional conditions that $I \models \exists p\,\Psi$ iff $I \models \Psi[p/\top]$ or $I \models \Psi[p/\bot]$, and $I \models \forall p\,\Psi$ iff $I \models \Psi[p/\top]$ and $I \models \Psi[p/\bot]$, for $\Phi = \mathsf{Q}p\,\Psi$ with $\mathsf{Q} \in \{\exists, \forall\}$.

## 3 Propositional Query Inclusion and Equivalence Problems

Eiter, Tompits, and Woltran [8] focus on two important instantiations of their framework, viz. on problems of form $(P, Q, \mathcal{P}_A, \subseteq_B)$ and $(P, Q, \mathcal{P}_A, =_B)$, where $A, B \subseteq \mathcal{U}$ are sets of atoms fixing the alphabet of the context class $\mathcal{P}_A$ and the alphabet relevant in comparing the answer sets, respectively. Our interest here are correspondence problems likewise parameterised by $A$ and $B$ as above, but where the context class is given by sets of facts from $A$ rather than $\mathcal{P}_A$.

**Definition 1.** *Let $\mathcal{U}$ be a set of atoms, $A, B \subseteq \mathcal{U}$, and $P, Q \in \mathcal{P}_\mathcal{U}$. Then, $(P, Q, 2^A, \subseteq_B)$ is called a* propositional query inclusion problem (over $\mathcal{U}$)*, or* PQIP *for short, and $(P, Q, 2^A, =_B)$ is called a* propositional query equivalence problem (over $\mathcal{U}$)*, or* PQEP.

*Example 1.* Consider $P = \{a \vee b \leftarrow; a \leftarrow c\}$ and $Q = \{a \leftarrow \mathrm{not}\, b; b \leftarrow \mathrm{not}\, a; c \leftarrow a\}$. Then, the answer sets of $P$ and $Q$ are given by $AS(P) = \{\{a\}, \{b\}\}$ and $AS(Q) = \{\{a, c\}, \{b\}\}$. Choosing $B = \{a, b\}$, we then have that $AS(P)|_B = AS(Q)|_B = \{\{a\}, \{b\}\}$. In fact, for $A = B = \{a, b\}$, the PQIP $(P, Q, 2^A, \subseteq_B)$ holds. $\Diamond$

Note that $(P, Q, \mathcal{P}_A, \subseteq_B)$ holds only if $(P, Q, 2^A, \subseteq_B)$ holds, but not vice versa. Indeed, $(P, Q, 2^A, \subseteq_B)$ from Example 1 holds but $(P, Q, \mathcal{P}_A, \subseteq_B)$ does not hold, as witnessed by the context program $\{a \leftarrow b; b \leftarrow a\} \in \mathcal{P}_A$.

It is convenient to assemble the objects witnessing the violation of a PQIP into a single concept. We introduce two versions of such a concept.

**Definition 2.** *Let $\Pi = (P, Q, 2^A, \subseteq_B)$ be a PQIP over $\mathcal{U}$.*

1. *A pair $(X, Y)$ with $X \subseteq A$ and $Y \subseteq \mathcal{U}$ is an* explicit counterexample (over $\mathcal{U}$) *for $\Pi$ iff $Y \in AS(P \cup X)$ and no $Y'$ with $Y'|_B = Y|_B$ is contained in $AS(Q \cup X)$.*
2. *A pair $(X, Y)$ with $X \subseteq A$ and $Y \subseteq B$ is a* projective counterexample *for $\Pi$ iff $Y \in AS(P \cup X)|_B$ and $Y \notin AS(Q \cup X)|_B$.*

**Theorem 1.** *Let $\Pi = (P, Q, 2^A, \subseteq_B)$ be a PQIP. Then, the following conditions are equivalent: (i) $\Pi$ does not hold; (ii) $\Pi$ has an explicit counterexample; and (iii) $\Pi$ has a projective counterexample.*

*For any explicit counterexample $(X, Y)$ for $\Pi$, $(X, Y|_B)$ is a projective counterexample for $\Pi$. Conversely, for any projective counterexample $(X, Y)$ for $\Pi$, there exists an explicit counterexample $(X, Y')$ with $Y'|_B = Y$.*

*Example 2.* Consider $P$ and $Q$ from Example 1. For $A = \{a, b, c\}$ and $B = \{a, b\}$, the PQIP $\Pi = (P, Q, 2^A, \subseteq_B)$ does not hold. This is witnessed by $(bc, abc)$[2] which is the unique explicit counterexample (over $\{a, b, c\}$) for $\Pi$. The corresponding projective counterexample for $\Pi$ is $(bc, ab)$. $\Diamond$

As far as PQEPs are concerned, we introduce the following notation:

**Definition 3.** *Let $\Pi = (P, Q, 2^A, =_B)$ be a PQEP. Then, $\Pi^{\rightarrow} = (P, Q, 2^A, \subseteq_B)$ and $\Pi^{\leftarrow} = (Q, P, 2^A, \subseteq_B)$ are the PQIPs* associated *with $\Pi$.*

Obviously, a PQEP $\Pi$ holds iff both $\Pi^{\rightarrow}$ and $\Pi^{\leftarrow}$ hold. We extend Definition 2 straightforwardly to PQEPs and call a pair $(X, Y)$ an explicit (resp., projective) counterexample for a PQEP $\Pi$ if $(X, Y)$ is an explicit (resp., projective) counterexample for $\Pi^{\rightarrow}$ or $\Pi^{\leftarrow}$.

Next, we introduce the novel concept of an *$A$-$B$-wedge* for programs over $\mathcal{U}$, where $A, B \subseteq \mathcal{U}$. $A$-$B$-wedges decide problems of form $(P, Q, 2^A, \odot_B)$, for $\odot \in \{\subseteq, =\}$, in such a way that they can be computed *separately* for $P$ and $Q$. In particular, an $A$-$B$-wedge for a program $P$ collects the projected answers sets for all possible extensions of $P$.

**Definition 4.** *Let $\mathcal{U}$ be a set of atoms, $A, B \subseteq \mathcal{U}$, and $P \in \mathcal{P}_{\mathcal{U}}$. A pair $(X, Y)$ of interpretations $X, Y \subseteq \mathcal{U}$ is an $A$-$B$-*wedge *(over $\mathcal{U}$) of $P$ iff $X \subseteq A$ and $Y \in AS(P \cup X)|_B$. The set of all $A$-$B$-wedges of $P$ is denoted by $\omega_{A,B}(P)$.*

Clearly, $(X, Y)$ is a projective counterexample for $\Pi = (P, Q, 2^A, \subseteq_B)$ iff $(X, Y) \in \omega_{A,B}(P) \setminus \omega_{A,B}(Q)$. From this, together with Theorem 1, the following central property is easily shown:

**Theorem 2.** *For any $\Pi$ of form $(P, Q,\ 2^A, \odot_B)$, where $\odot \in \{\subseteq, =\}$, $\Pi$ holds iff $\omega_{A,B}(P) \odot \omega_{A,B}(Q)$.*

[2] Whenever convenient, we use strings like $abc$ as a shorthand for $\{a, b, c\}$.

*Example 3.* Reconsider the programs $P$ and $Q$ from Example 1 and take $\mathcal{U} = \{a, b, c\}$. First, consider $A = B = \{a, b\}$. One can verify that $\omega_{A,B}(P) = \omega_{A,B}(Q) = \mathcal{S}$, where $\mathcal{S} = \{(\emptyset, a), (\emptyset, b), (a, a), (b, b), (ab, ab)\}$. Hence, the PQEP $(P, Q, 2^A, =_B)$ holds.

Second, consider the PQEP $(P, Q, 2^{A'}, =_B)$ with $A' = A \cup \{c\}$. We now obtain

$$\begin{aligned}\omega_{A',B}(P) &= \mathcal{S} \cup \{(c, a), (ac, a), (bc, ab), (abc, ab)\},\\ \omega_{A',B}(Q) &= \mathcal{S} \cup \{(c, a), (c, b), (ac, a), (bc, b), (abc, ab)\}.\end{aligned}$$

By Theorem 2, $(P, Q, 2^{A'}, =_B)$ does not hold. All projective counterexamples are given by the symmetric difference $\omega_{A',B}(P) \triangle \omega_{A',B}(Q) = \{(c, b), (bc, b), (bc, ab)\}$, and the corresponding explicit counterexamples are $(c, bc)$, $(bc, bc)$, and $(bc, abc)$. ♢

*Model-Theoretic Characterisations.* We now introduce semantic characterisations for explicit counterexamples and $A$-$B$-wedges in the style of UE-models [3] and $A$-UE-models [9]. Recall that UE- and $A$-UE-models have been introduced to capture uniform equivalence and uniform equivalence relative to $A$, respectively. More specifically, two programs are uniformly equivalent iff their UE-models coincide, and they are uniformly equivalent relative to $A$ iff their $A$-UE-models coincide. Let us note that UE-models can be characterised thus: a pair $(X, Y)$ is a UE-model of a program $P$ iff $X \subseteq Y$, $Y \models P$, $X \models P^Y$, and, for each $X'$ with $X \subset X' \subset Y$, $X' \not\models P^Y$.

We first deal with explicit counterexamples.

**Theorem 3.** *Let $\Pi = (P, Q, 2^A, \subseteq_B)$ be a PQIP over $\mathcal{U}$ and consider $X, Y \subseteq \mathcal{U}$. Then, $(X, Y)$ is an explicit counterexample over $\mathcal{U}$ for $\Pi$ iff*

1. *$Y \models P$ and $X \subseteq Y|_A$,*
2. *for each $Y'$ with $X \subseteq Y' \subset Y$, $Y' \not\models P^Y$, and*
3. *for each $Z$ with $X \subseteq Z$, $Z|_B = Y|_B$, and $Z \models Q$, there is some $Z'$ with $X \subseteq Z' \subset Z$ such that $Z' \models Q^Z$.*

*Proof.* We first show that $Y \in AS(P \cup X)$ and $X \subseteq A$ jointly hold iff the first two items of the theorem hold. We only show the only-if direction; the if-direction is by essentially the same arguments. So, assume that $Y \in AS(P \cup X)$ and $X \subseteq A$. Since, $Y \in AS(P \cup X)$, we have $Y \models (P \cup X)^Y = (P^Y \cup X)$. Hence, $Y \models P^Y$ and thus $Y \models P$. Moreover, $X \subseteq Y$ has to hold. Since $X \subseteq A$ by hypothesis, we get $X \subseteq Y|_A$. Furthermore, $Y \in AS(P \cup X)$ implies that there exists no $Y'$ with $Y' \subset Y$ such that $Y' \models (P \cup X)^Y = (P^Y \cup X)$. In particular, this yields that for each such $Y'$ with $X \subseteq Y'$, $Y' \not\models P^Y$ has to hold.

Finally, it can be shown that there exist no $Z$ with $Z|_B = Y|_B$ such that $Z \in AS(Q \cup X)$ iff the third item of the theorem holds. □

Next, we characterise $A$-$B$-wedges.

**Theorem 4.** *A pair $(X, Y)$ is an $A$-$B$-wedge of $P$ iff (i) $X \subseteq A$ and (ii) there is a $Y'$ with $X \subseteq Y'$ and $Y = Y'|_B$ such that $Y' \models P$ and, for each $X'$ with $X \subseteq X' \subset Y'$, $X' \not\models P^{Y'}$.*

*Proof.* According to Definition 4, $(X, Y)$ is an $A$-$B$-wedge of $P$ iff $X \subseteq A$ and $Y \in AS(P \cup X)|_B$. It thus remains to show that the latter condition is equivalent to (ii). Now, $Y \in AS(P \cup X)|_B$ iff there is some $Y'$ with $Y = Y'|_B$ and $Y' \in AS(P \cup X)$. By the definition of an answer set, the latter is equivalent to

$(*)$ $Y'$ is a minimal model of $(P \cup X)^{Y'}$.

Since $(P \cup X)^{Y'} = (P^{Y'} \cup X)$ and $Y' \models P^{Y'}$ iff $Y' \models P$, $(*)$ is in turn equivalent to $Y' \models P$, $X \subseteq Y'$, and for each $X'$ with $X \subseteq X' \subset Y'$, $X' \not\models P^{Y'}$. □

Since uniform equivalence between programs over $\mathcal{U}$ is captured by PQEPs over $\mathcal{U}$ of form $(P, Q, 2^{\mathcal{U}}, =_{\mathcal{U}})$, let us now describe the relation between UE-models and $A$-$B$-wedges with $A = B = \mathcal{U}$.

First of all, a pair $(X, Y)$ is a $\mathcal{U}$-$\mathcal{U}$-wedge of some program only if $X \subseteq Y|_{\mathcal{U}}$, i.e., only if $X \subseteq Y$. Now, for a program $P$, $(Y, Y)$ is a $\mathcal{U}$-$\mathcal{U}$-wedge of $P$ iff $Y \models P$. Furthermore, for $X \subset Y$, $(X, Y)$ is a $\mathcal{U}$-$\mathcal{U}$-wedge of $P$ iff $Y \models P$ and for all $X'$ with $X \subseteq X' \subset Y$, $X' \not\models P^Y$ holds. So, there is only a subtle difference between $\mathcal{U}$-$\mathcal{U}$-wedges and UE-models, laid down in detail by the next result.

**Theorem 5.** *Let $X \subseteq Y \subseteq \mathcal{U}$ and $P \in \mathcal{P}_{\mathcal{U}}$. Then:*

1. *$(Y, Y)$ is a UE-model of $P$ iff $(Y, Y)$ is a $\mathcal{U}$-$\mathcal{U}$-wedge of $P$. Moreover, if $(Y, Y)$ is a UE-model of $P$ but no $(X, Y)$ with $X \subset Y$ is a UE-model of $P$ (i.e., $Y$ is an answer set of $P$), then, for all $X \subseteq Y$, $(X, Y)$ is a $\mathcal{U}$-$\mathcal{U}$-wedge of $P$.*
2. *If $(X, Y)$ is a UE-model of $P$ and $X \subset Y$, then $(X', Y)$ is a $\mathcal{U}$-$\mathcal{U}$-wedge for any $X \subset X' \subseteq Y$.*
3. *If $(X, Y)$ is a $\mathcal{U}$-$\mathcal{U}$-wedge of $P$ and $(\emptyset, Y)$ is not a $\mathcal{U}$-$\mathcal{U}$-wedge of $P$, then there exists an UE-model $(X', Y)$ of $P$ with $X' \subset X$.*

*Example 4.* Consider the programs $P = \{a \vee b\}$ and $Q = \{a \leftarrow \text{not } b;\ b \leftarrow \text{not } a\}$, which are uniformly equivalent. The UE-models of $P$ and $Q$ are $(a, a)$, $(b, b)$, $(a, ab)$, $(b, ab)$, and $(ab, ab)$, but the $\mathcal{U}$-$\mathcal{U}$-wedges of the two programs are $(\emptyset, a)$, $(a, a)$, $(\emptyset, b)$, $(b, b)$, and $(ab, ab)$. ♢

While UE-models were defined with the aim to select a subset of SE-models [12] (which characterise strong equivalence), wedges are not designed in this respect. Rather, they have a much closer relation to projective counterexamples. Furthermore, a relation between $A$-UE-models [9] and $A$-$\mathcal{U}$-wedges can be established similar to Theorem 5 in the context of relativised uniform equivalence.

## 4 Computational Complexity

We now analyse the complexity of deciding PQIPs and PQEPs. Let us first summarise some results from Eiter, Tompits, and Woltran [8].

**Proposition 1.** *Given programs $P, Q \in \mathcal{P}_{\mathcal{U}}$, sets $A, B \subseteq \mathcal{U}$, and $\odot \in \{\subseteq, =\}$, deciding whether $(P, Q, \mathcal{P}_A, \odot_B)$ holds is $\Pi_4^P$-complete. Moreover, the problem is* coNP-*complete if $A = \mathcal{U}$.*

**Proposition 2.** *Given programs $P, Q \in \mathcal{P}_{\mathcal{U}}$, $B \subseteq \mathcal{U}$, $\odot \in \{\subseteq, =\}$, and $\mathcal{C} \subseteq \mathcal{P}_{\mathcal{U}}$, where each $R \in \mathcal{C}$ is polynomial in the size of $P \cup Q$, deciding whether $(P, Q, \mathcal{C}, \odot_B)$ holds is $\Pi_3^P$-complete.*

Another relevant previous result concerns the complexity of checking relativised uniform equivalence [9, 13], which thus provides us complexity bounds for PQIPs and PQEPs without projection.

**Proposition 3.** *Given programs $P, Q \in \mathcal{P}_{\mathcal{U}}$, $A \subseteq \mathcal{U}$, and $\odot \in \{\subseteq, =\}$, deciding whether $(P, Q, 2^A, \odot_{\mathcal{U}})$ holds is $\Pi_2^P$-complete. Moreover, hardness holds even for arbitrary but fixed $A$.*

For general PQIPs and PQEPs, we expect an increase in complexity but, in view of Proposition 2, it cannot be beyond $\Pi_3^P$. However, Proposition 2 does not provide details about the hardness of such problems. In fact, Eiter, Tompits, and Woltran [8] report $\Pi_3^P$-hardness for ordinary equivalence with projection, i.e., PQEPs of the form $(P, Q, 2^A, =_B)$ with $A = \emptyset$. Our main result below shows that nearly all parameterisations for PQIPs and PQEPs result in a matching lower bound. In particular, we show that $\Pi_3^P$-hardness holds even if the context alphabet $A$ is fixed arbitrarily. Thus, also uniform equivalence without restriction of the context class is hard for $\Pi_3^P$ as long as $B \subset \mathcal{U}$, where $B$ is the projection set. This is in stark contrast to Proposition 1, which shows that considering programs over $A$ (instead of *sets of facts* over $A$) remains in coNP for arbitrary $B$, providing $A = \mathcal{U}$.

**Theorem 6.** *Given programs $P, Q \in \mathcal{P}_{\mathcal{U}}$ and sets $A, B \subseteq \mathcal{U}$ of atoms, deciding whether $(P, Q, 2^A, \subseteq_B)$ holds is $\Pi_3^P$-complete. Hardness holds even for arbitrary but fixed $A$.*

*Proof* (*Sketch*). Membership in $\Pi_3^P$ follows from Proposition 2. We show $\Pi_3^P$-hardness by reducing the $\Pi_3^P$-hard problem of checking validity of a QBF of form $\forall Z \exists X \forall Y \phi$, where $\phi$ is a propositional formula in disjunctive normal form (DNF) and $Z \cup X \cup Y$ are the variables occurring in $\phi$, into PQIPs.

The reduction is as follows: Let $\Phi = \forall Z \exists X \forall Y \phi$ be a QBF of the described form, with $\phi = \bigvee_{i=1}^{n} C_i$ being a formula in DNF. Define $\Pi_\Phi = (P_\Phi, Q_\Phi, 2^A, \subseteq_Z)$, where $A$ is an arbitrary set of atoms and $P_\Phi, Q_\Phi$ are given as follows:

$$\begin{aligned}
P_\Phi &= \{z \vee \bar{z} \leftarrow;\ \leftarrow z, \bar{z} \mid z \in Z\} \cup \{\leftarrow v;\ \leftarrow \bar{v} \mid v \in X \cup Y\}; \\
Q_\Phi &= \{v \vee \bar{v} \leftarrow;\ \leftarrow v, \bar{v} \mid v \in Z \cup X\} \cup \\
&\quad\ \{y \vee \bar{y} \leftarrow;\ y \leftarrow a;\ \bar{y} \leftarrow a;\ a \leftarrow y, \bar{y} \mid y \in Y\} \cup \\
&\quad\ \{a \leftarrow C_i^* \mid 1 \leq i \leq n\} \cup \{a \leftarrow \mathrm{not}\ a\}.
\end{aligned}$$

Here, $C^*$ is a sequence of atoms containing each atom $w$ occurring positively in $C$, and $\bar{w}$ for each $w$ occurring negatively in $C$. Moreover, $a$ and all $\bar{v}$'s are new distinct atoms. It can be shown that $\Phi$ is valid iff $\Pi_\Phi$ holds. □

Since a PQEP $\Pi$ holds iff its associated PQIPs $\Pi^{\rightarrow}$ and $\Pi^{\leftarrow}$ both hold, it follows that the complexity of checking PQEPs is in $\Pi_3^P$ as well. The matching lower bounds for PQEPs $(P, Q, 2^A, =_B)$ for arbitrary $A$ follow in view of the following lemma, which slightly generalises a result by Eiter, Tompits, and Woltran [8].

**Lemma 1.** *The PQIP* $(P, Q, 2^A, \subseteq_B)$ *holds iff the PQEP* $(L_{P,Q} \cup \{g_Q \leftarrow \text{not}\, g_P\}$, $L_{P,Q} \cup \{g_Q \vee g_P \leftarrow\}, 2^C, =_B)$ *holds, where* $L_{P,Q} = \{\leftarrow g_P, g_Q\} \cup \{H \leftarrow g_R, B \mid R \in \{P, Q\}, H \leftarrow B \in R\}$, $A \subseteq C \subseteq A \cup \{g_P, g_Q\}$, *and* $g_P, g_Q$ *are new atoms.*

Roughly speaking, the above lemma yields two properties: First, it maps PQIPs into PQEPs via two new atoms. Second, it shows that these new atoms can be arbitrarily fixed in the context of the resulting PQEP. Thus, hardness carries over also for arbitrary but fixed alphabets and we have the following result:

**Theorem 7.** *Given programs* $P, Q \in \mathcal{P}_{\mathcal{U}}$ *and sets* $A, B \subseteq \mathcal{U}$ *of atoms, deciding whether* $(P, Q, 2^A, =_B)$ *holds is* $\Pi_3^P$*-complete. Hardness holds even for arbitrary but fixed* $A$*.*

We observe that thus also the special case when $A = B$, which amounts to notions similar to modular equivalence [15], and database-like settings, where $A$ is a subset of the common EDBs and $B$ is a subset of common IDBs, remain hard for $\Pi_3^P$.

## 5 Translating Query Problems

In this section, we discuss issues for computing PQIPs and PQEPs. We adopt a reduction approach here, translating the problems under consideration into problems for which solvers already exist. Naturally, the translations we seek should be constructible in polynomial time.

First of all, we remark that since checking PQIPs and PQEPs is $\Pi_3^P$-complete, these tasks cannot be efficiently reduced to DLPs under the answer-set semantics, unless the polynomial hierarchy collapses. Hence, a more expressive language is required. This leads us to quantified propositional logic as a suitable target language, as any decision problem in PSPACE can be efficiently reduced to QBFs. Moreover, there are several practicably efficient solvers for QBFs available, which can be used as back-end inference engines for solving the encoded problems.

In fact, such a reduction approach to QBFs was already adopted for realising the system $\mathrm{cc}\top$ [11], which allows to verify the kinds of correspondence problems studied by Eiter, Tompits, and Woltran [8]. In principle, we can use $\mathrm{cc}\top$ as such to verify PQIPs and PQEPs, because the latter problems can be reduced to the former, in view of the following observation:

**Theorem 8.** *Given* $\Pi = (P, Q, 2^A, \odot_B)$, *for* $\odot \in \{\subseteq, =\}$, *we have that* $\Pi$ *holds iff* $(P \cup G_A, Q \cup G_A, \{\emptyset\}, \odot_B)$ *holds, where* $G_A = \{a' \vee a'' \leftarrow; a \leftarrow a' \mid a \in A\}$ *and all* $a', a''$ *are new, mutually disjoint atoms.*

However, verifying PQIPs and PQEPs that way would involve two reduction steps, as $\mathrm{cc}\top$ relies itself on a reduction to QBFs. The direct encodings described next avoid this.

In what follows, we make use of sets of globally new atoms in order to refer to different assignments of the same atoms within a single formula. More formally, given a set $V$ of atoms, we assume (pairwise) disjoint copies $V^i = \{v^i \mid v \in V\}$, for every $i \geq 1$. Furthermore, we introduce the following abbreviations:

1. $(V^i \leq V^j) = \bigwedge_{v \in V}(v^i \rightarrow v^j)$;
2. $(V^i < V^j) = (V^i \leq V^j) \wedge \neg(V^j \leq V^i)$; and
3. $(V^i = V^j) = (V^i \leq V^j) \wedge (V^j \leq V^i)$.

Observe that the latter is equivalent to $\bigwedge_{v \in V}(v^i \leftrightarrow v^j)$.

These operators allow to compare different subsets of atoms from a common set $V$ under subset inclusion, proper-subset inclusion, and equality, respectively, in the following way: Given $X, Y \subseteq V$, an interpretation $I$ with $I|_{V^i} = X^i$ and $I|_{V^j} = Y^j$ is (i) a model of $V^i \leq V^j$ iff $X \subseteq Y$, (ii) a model of $V^i < V^j$ iff $X \subset Y$, and (iii) a model of $V^i = V^j$ iff $X = Y$.

We use superscripts as a general renaming schema for formulas and rules. That is, for each $i \geq 1$, $\alpha^i$ expresses the result of replacing each occurrence of an atom $v$ in $\alpha$ by $v^i$, where $\alpha$ is any formula or rule. For a rule $r$ of form (1), we define $H(r) = a_1 \vee \cdots \vee a_l$, $B^+(r) = a_{l+1} \wedge \cdots \wedge a_m$, and $B^-(r) = \neg a_{m+1} \wedge \cdots \wedge \neg a_n$. We identify empty disjunctions with $\bot$ and empty conjunctions with $\top$.

The following abbreviation is central: For any program $P$,

$$P^{\langle i,j \rangle} = \bigwedge_{r \in P} \big((B^+(r^i) \wedge B^-(r^j)) \rightarrow H(r^i)\big).$$

Then, the following relation holds:

**Proposition 4 ([16]).** *Let $P$ be a program with $At(P) = V$, $I$ an interpretation, and $X, Y \subseteq V$ such that, for some $i$, $j$, $I|_{V^i} = X^i$ and $I|_{V^j} = Y^j$. Then, $X \models P^Y$ iff $I \models P^{\langle i,j \rangle}$.*

*Example 5.* Consider the program $Q = \{a \leftarrow \text{not } b;\ b \leftarrow \text{not } a\}$. Then, for instance, $Q^{\langle 1,2 \rangle}$ is given by $(\neg b^2 \rightarrow a^1) \wedge (\neg a^2 \rightarrow b^1)$, and we have that the interpretation $\{a^2, b^2\} \cup X^1$ is a model of $Q^{\langle 1,2 \rangle}$, for each $X^1 \subseteq \{a^1, b^1\}$, reflecting the fact that any interpretation $X$ is a model of the reduct $Q^{\{a,b\}}$. $\Diamond$

With these building blocks at hand, we proceed with our central encoding.

**Definition 5.** *Let $\Pi = (P, Q, 2^A, \subseteq_B)$ be a PQIP, $At(P \cup Q) = V$, and $A, B \subseteq V$. Then,*

$$\mathcal{T}[\Pi] = \Phi_\Pi \wedge \forall V^4\big((B^4 = B^1) \rightarrow \Psi_\Pi\big), \text{ where}$$
$$\Phi_\Pi = P^{\langle 1,1 \rangle} \wedge (A^2 \leq A^1) \wedge \forall V^3\Big(\big((A^2 \leq A^3) \wedge (V^3 < V^1)\big) \rightarrow \neg P^{\langle 3,1 \rangle}\Big) \text{ and}$$
$$\Psi_\Pi = \Big(\big(Q^{\langle 4,4 \rangle} \wedge (A^2 \leq A^4)\big) \rightarrow \exists V^5\big(\big((A^2 \leq A^5) \wedge (V^5 < V^4)\big) \wedge Q^{\langle 5,4 \rangle}\big)\Big).$$

Observe that the free variables of $\mathcal{T}[\Pi]$ are given by $V^1 \cup A^2$. Assignments to $V^1 \cup A^2$ yield the explicit counterexamples for $\Pi$, in case $\mathcal{T}[\Pi]$ is satisfied by those assignments. More specifically, $\mathcal{T}[\Pi]$ expresses the conditions of Theorem 3, where assignments for $V^1$, $A^2$, $V^3$, $V^4$, and $V^5$ correspond to $Y$, $X$, $Y'$, $Z$, and $Z'$, respectively. Taking the semantics of the introduced building blocks into account, $Y^1 \cup X^2 \models \Phi_\Pi$ iff $X$ and $Y$ satisfy the first and the second item of Theorem 3, and $Y^1 \cup X^2 \models \forall V^4((B^4 = B^1) \rightarrow \Psi_\Pi)$ iff $X$ and $Y$ satisfy the third item of Theorem 3. Formally, we have the following key property:

**Lemma 2.** *Let $\Pi = (P, Q, 2^A, \subseteq_B)$ be a PQIP, $At(P \cup Q) = V$, $A, B \subseteq V$, $X \subseteq A$, and $Y \subseteq V$. Then, $(X, Y)$ is an explicit counterexample for $\Pi$ iff $Y^1 \cup X^2 \models \mathcal{T}[\Pi]$.*

Expressing the task whether a PQIP holds is now a simple matter to realise:

**Theorem 9.** *For any PQIP $\Pi = (P, Q, 2^A, \subseteq_B)$, $\Pi$ holds iff $\neg\exists V^1 \exists A^2 \mathcal{T}[\Pi]$ is valid.*

The extension of the encodings to PQEPs is done by means of the associated PQIPs.

**Lemma 3.** *Let $\Pi = (P, Q, 2^A, =_B)$ be a PQEP, $At(P \cup Q) = V$, $A, B \subseteq V$, $X \subseteq A$, and $Y \subseteq V$. Then, $(X, Y)$ is an explicit counterexample for $\Pi$ iff $Y^1 \cup X^2 \models \mathcal{T}[\Pi^{\rightarrow}] \vee \mathcal{T}[\Pi^{\leftarrow}]$.*

**Theorem 10.** *For any PQEP $\Pi = (P, Q, 2^A, =_B)$, $\Pi$ holds iff $\neg\exists V^1 \exists A^2 (\mathcal{T}[\Pi^{\rightarrow}] \vee \mathcal{T}[\Pi^{\leftarrow}])$ is valid.*

It is easily observed that our encodings in Theorems 9 and 10 are (i) always polynomial in the size of $P$, $Q$, $A$, and $B$, and (ii) possess at most two quantifier alternations in any branch of the formula tree. Thus, the complexity of evaluating these QBFs is not harder than the complexity of the encoded decision problems, which shows that our encodings are *adequate* in the sense of Besnard *et al.* [17].

The reductions described above have been incorporated into the system cc$\top$, which is available on the Web at

http://www.kr.tuwien.ac.at/research/ccT.

In fact, the implemented translations include optimised versions such that adequacy is retained also for (relativised) uniform equivalence. Details about these optimisations are omitted for space reasons.

## 6 Discussion

In this paper, we studied refined versions of uniform equivalence for disjunctive logic programs under the answer-set semantics, where the alphabet of the context class as well as removal of auxiliary atoms is taken into account. We also considered correspondence problems in which projective set inclusion is taken as basic comparison relation instead of projective set equality. We furthermore provided a novel model-theoretic characterisation in terms of wedges which at the same time yields new characterisations for (relativised) uniform equivalence (vis-a-vis UE-models), and analysed the computational complexity of correspondence checking. Finally, we described efficient reductions of PQIPs and PQEPs to QBFs, yielding an implementation of these problems by means of off-the-shelf QBF solvers.

Other refined equivalence notions in the context of answer-set programming are, e.g., *visible equivalence* [18], a form of ordinary equivalence with projection, and *update equivalence* [19]. We also mention the system SELP [20] for checking strong equivalence, which is based on a reduction to classical logic, very much in the spirit of our implementation approach.

An open topic for future work is the extension of our results to more general classes of programs like, e.g., nested logic programs. A further interesting issue concerns the case of nonground programs—however, thereby we have to face undecidability which holds already for uniform equivalence [21] between nonground programs.

## References

1. Gelfond, M., Lifschitz, V.: Classical Negation in Logic Programs and Disjunctive Databases. New Generation Computing **9** (1991) 365–385
2. Lifschitz, V., Pearce, D., Valverde, A.: Strongly Equivalent Logic Programs. ACM TOCL **2**(4) (2001) 526–541
3. Eiter, T., Fink, M.: Uniform Equivalence of Logic Programs under the Stable Model Semantics. In Proc. ICLP 2003. Volume 2916 of LNCS, Springer (2003) 224–238
4. Eiter, T., Fink, M., Tompits, H., Woltran, S.: Simplifying Logic Programs Under Uniform and Strong Equivalence. In Proc. LPNMR-7. Volume 2923 of LNCS, Springer (2004) 87–99
5. Pearce, D.: Simplifying Logic Programs under Answer Set Semantics. In Proc. ICLP 2004. Volume 3132 of LNCS, Springer (2004) 210–224
6. Lin, F., Chen, Y.: Discovering Classes of Strongly Equivalent Logic Programs. In Proc. IJCAI 2005. (2005) 516–521
7. Sagiv, Y.: Optimizing Datalog Programs. In Foundations of Deductive Databases and Logic Programming. Morgan Kaufmann (1988) 659–698
8. Eiter, T., Tompits, H., Woltran, S.: On Solution Correspondences in Answer Set Programming. In Proc. IJCAI 2005. (2005) 97–102
9. Woltran, S.: Characterizations for Relativized Notions of Equivalence in Answer Set Programming. In Proc. JELIA 2004. Volume 3229 of LNCS, Springer (2004) 161–173
10. Huxley, A.: Proper Studies. Doubleday Doran (1928)
11. Oetsch, J., Seidl, M., Tompits, H., Woltran, S.: ccT: A Tool for Checking Advanced Correspondence Problems in Answer-Set Programming. In Proc. CIC 2006, IEEE Computer Society Press (2006) 3–10
12. Turner, H.: Strong Equivalence Made Easy: Nested Expressions and Weight Constraints. TPLP **3**(4-5) (2003) 602–622
13. Eiter, T., Fink, M., Woltran, S.: Semantical Characterizations and Complexity of Equivalences in Stable Logic Programming. ACM TOCL (2007). To appear
14. Eiter, T., Gottlob, G.: On the Computational Cost of Disjunctive Logic Programming: Propositional Case. Annals of Mathematics and Artificial Intelligence **15**(3/4) (1995) 289–323
15. Oikarinen, E., Janhunen, T.: Modular Equivalence for Normal Logic Programs. In Proc. ECAI 2006, IOS Press (2006) 412–416
16. Tompits, H., Woltran, S.: Towards Implementations for Advanced Equivalence Checking in Answer-Set Programming. In Proc. ICLP 2005. Volume 3668 of LNCS, Springer (2005) 189–203
17. Besnard, P., Schaub, T., Tompits, H., Woltran, S.: Representing Paraconsistent Reasoning via Quantified Propositional Logic. In Inconsistency Tolerance. Volume 3300 of LNCS, Springer (2005) 84–118
18. Janhunen, T., Oikarinen, E.: Automated Verification of Weak Equivalence within the SMODELS System. TPLP **7**(4) (2007) 1–48
19. Inoue, K., Sakama, C.: Equivalence of Logic Programs Under Updates. In Proc. JELIA 2004. Volume 3229 of LNCS, Springer (2004) 174–186
20. Chen, Y., Lin, F., Li, L.: SELP - A System for Studying Strong Equivalence Between Logic Programs. In Proc. LPNMR 2005. Volume 3552 of LNAI, Springer (2005) 442–446
21. Eiter, T., Fink, M., Tompits, H., Woltran, S.: Strong and Uniform Equivalence in Answer-Set Programming: Characterizations and Complexity Results for the Non-Ground Case. In Proc. AAAI 2005, AAAI Press (2005) 695–700