On Solution Correspondences in Answer-Set Programming*

Thomas Eiter, Hans Tompits, and Stefan Woltran
Institut für Informationssysteme, Technische Universität Wien,
Favoritenstraße 9–11, A-1040 Vienna, Austria
[eiter,tompits,stefan]@kr.tuwien.ac.at

Abstract

We introduce a general framework for specifying program correspondence under the answer-set semantics. The framework allows to define different kinds of equivalence notions, including previously defined notions like strong and uniform equivalence, in which programs are extended with rules from a given context, and correspondence is determined by means of a binary relation. In particular, refined equivalence notions based on projected answer sets can be defined within this framework, where not all parts of an answer set are of relevance. We study general characterizations of inclusion and equivalence problems, introducing novel semantical structures. Furthermore, we deal with the issue of determining counterexamples for a given correspondence problem, and we analyze the computational complexity of correspondence checking.

1 Introduction

With the availability of efficient implementations of the answer-set semantics [Gelfond and Lifschitz, 1991] for non-monotonic logic programs, *answer-set programming* (ASP) has been recognized as a fruitful paradigm for declarative problem solving. In this approach, a problem is encoded as a logic program such that its models, called *answer sets*, correspond to the solutions of the problem, which can be easily extracted from them. Due to the availability of default negation, ASP has become an important host for solving many AI problems, including planning, diagnosis, information integration, and inheritance reasoning (cf. Gelfond and Leone [2002] for an overview on ASP).

To support engineering of ASP solutions, an important issue is determining equivalence of different encodings, given by two programs. To this end, various notions of equivalence between programs under the answer-set semantics have been studied in the recent past, viz. *strong equivalence* [Lifschitz *et al.*, 2001], *uniform equivalence* [Eiter and Fink, 2003], and relativized notions thereof [Woltran, 2004], as well as *update equivalence* [Inoue and Sakama, 2004]. Informally, the former notions consider programs P and Q to be equivalent, if $P \cup R$ and $Q \cup R$ have always the same answer sets, where R is a set of rules from a particular collection of rules. Thus, if R is regarded as possible input, equivalent P and Q are guaranteed to compute always the same answer sets. Note that also open logic programs [Bonatti, 2001], in which part of the rules are to be added at runtime, fit within this scheme.

However, none of these works have considered the practicably important setting of *projected* answer sets in ASP. Here, not a whole answer set of a program P is of interest, but only its intersection on a subset $B \subseteq U$ of all letters; this includes, in particular, removal of auxiliary letters in computation. For a simple example, consider the programs

$$\begin{aligned}P = \{\ & sel(X) \leftarrow s(X), not\ out(X),\\ & out(X) \vee out(Y) \leftarrow s(X), s(Y), X \neq Y\ \} \text{ and}\\ Q = \{\ & sel(X) \leftarrow s(X), not\ skip(X),\\ & skip(X) \leftarrow sel(X), s(Y), X \neq Y\ \}.\end{aligned}$$

They should select, by means of sel, one element satisfying s in their answer sets. Here, an important issue is whether the programs are equivalent with respect to the "output" predicate sel, for all "inputs" s, where s may be defined by rules over predicates from a set A, say, or given by facts.

Another aspect is that, apart from equivalence, other relationships between the sets of answer sets of P and Q might be of interest. A natural example is *inclusion*, which means that each answer set of P is also an answer set of Q. Here Q can be viewed as an approximation of P, which is sound with respect to cautious reasoning from P.

Motivated by these observations, in this paper, we consider solution correspondences in ASP at a generic level. Our main contributions are briefly summarized as follows.

(1) We introduce a general framework for correspondences between the answer sets of programs P and Q, which are augmented by further rules from a *context* $\mathcal{C}$ of possible extensions, where the correspondence is determined by a binary relation ρ. Previous notions of equivalence and projected answer sets amount to particular instances of this framework.

(2) We provide characterizations of inclusion correspondence and equivalence between programs under projected answer sets, in terms of novel semantical structures, called *spoilers*, which refute this property, and *certificates*, which

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

capture the essence of equivalence, similar as SE-models and UE-models do for strong and uniform equivalence, respectively. Based on these characterizations, we present interesting correspondence results on varying projection sets in ASP.

(3) We show how spoilers can be used to construct *counterexamples* to an inclusion resp. equivalence correspondence, consisting of a suitable interpretation M and a rule set R such that M is an answer set of exactly one of $P \cup R$ and $Q \cup R$.

(4) Finally, we determine the computational complexity of correspondence checking for propositional disjunctive programs under projected answer sets. Our main result is that equivalence checking is Π_4^P-complete in general, and thus feasible in polynomial space as compared to a naive guess-and-check procedure which requires exponential space. Furthermore, we show that for restricted settings, the complexity gradually decreases from Π_4^P to coNP.

The results presented in this paper significantly advance the current state of equivalence testing in ASP towards highly relevant settings for practical applications, and provide novel insight into the structure of equivalent programs. Besides the papers quoted above, further related work mostly addresses semantic and complexity characterizations of equivalence [Lin, 2002; Turner, 2003], or describes implementation methods [Eiter *et al.*, 2004; Oikarinen and Janhunen, 2004]. The recent work by Pearce and Valverde [2004] addresses strong equivalence of programs over disjoint alphabets which are synonymous under structurally defined mappings.

The characterizations we present are non-trivial and, as shown by our complexity results, necessarily involved in the sense that no "simple" criteria exist. Together with counterexamples, they provide a basis for powerful program optimization and debugging tools for ASP, which are lacking to date.

2 Preliminaries

We deal with propositional disjunctive logic programs, which are finite sets of rules of form

$$a_1 \vee \cdots \vee a_l \leftarrow a_{l+1}, \ldots, a_m, not\, a_{m+1}, \ldots, not\, a_n, \quad (1)$$

$n \geq m \geq l \geq 0$, where all a_i are propositional atoms and not denotes default negation; for $n = l = 1$, we usually identify the rule with the atom a_1. If all atoms occurring in a program P are from a given set A of atoms, we say that P is a program *over* A. The set of all programs over A is denoted by $\mathcal{P}_A$. We call a rule (resp., program) *normal* iff it contains no disjunction $\vee$. A program is *unary* if it contains only rules of the form $a \leftarrow b$ or $a \leftarrow$; $\mathcal{P}_A^{un}$ denotes the set of all unary programs over A. For a set of atoms $A = \{a_1, \ldots, a_m\}$, $not\, A$ denotes the set $\{not\, a_1, \ldots, not\, a_m\}$. Accordingly, rules of form (1) will also be written as $a_1 \vee \cdots \vee a_l \leftarrow B_1, not\, B_2$, where $B_1 = \{a_{l+1}, \ldots, a_m\}$ and $B_2 = \{a_{m+1}, \ldots, not\, a_n\}$.

Following Gelfond and Lifschitz [1991], an interpretation I, i.e., a set of atoms, is an *answer set* of a program P iff it is a minimal model of the *reduct* P^I, which results from P by (i) deleting all rules containing default negated atoms $not\, a$ such that $a \in I$ and (ii) deleting all default negated atoms in the remaining rules. The set of all answer sets of a program P is denoted by $\mathcal{AS}(P)$. The relation $I \models P$ between an interpretation I and a program P is defined as usual.

Under the answer-set semantics, two programs P and Q are regarded as (ordinarily) equivalent iff $\mathcal{AS}(P) = \mathcal{AS}(Q)$. The more restrictive forms of *strong equivalence* [Lifschitz *et al.*, 2001] and *uniform equivalence* [Eiter and Fink, 2003] have recently been generalized as follows [Woltran, 2004]: Let P, Q be programs over $\mathcal{U}$, and let $A \subseteq \mathcal{U}$. Then, P and Q are *strongly* (resp., *uniformly*) *equivalent relative to* A iff, for any $R \in \mathcal{P}_A$ (resp., $R \subseteq A$), $\mathcal{AS}(P \cup R) = \mathcal{AS}(Q \cup R)$.

If $A = \mathcal{U}$, strong (resp., uniform) equivalence relative to A reduces to strong (resp., uniform) equivalence simpliciter; if $A = \emptyset$, ordinary equivalence results in either case.

We use the following notation. For an interpretation I and a set $\mathcal{S}$ of (pairs of) interpretations, we write $\mathcal{S}|_I = \{Y \cap I \mid Y \in \mathcal{S}\}$ ($\mathcal{S}|_I = \{(X \cap I, Y \cap I) \mid (X, Y) \in \mathcal{S}\}$). If $\mathcal{S} = \{s\}$, we write $s|_I$ instead of $\mathcal{S}|_I$, if convenient.

For any $A \subseteq \mathcal{U}$ and any $Y \subseteq \mathcal{U}$, a pair (X, Y) of interpretations is an *A-SE-interpretation* (over $\mathcal{U}$) iff either $X = Y$ or $X \subset Y|_A$. (X, Y) is an *A-SE-model* of a program P iff

(i) $Y \models P$;

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

(iii) $X \subset Y$ implies the existence of an $X' \subseteq Y$ with $X'|_A = X$ such that $X' \models P^Y$ holds.

A pair (X, Y) is *total* iff $X = Y$, and *non-total* otherwise. The set of all A-SE-models of P is denoted by $SE^A(P)$.

For $A = \mathcal{U}$, A-SE-interpretations (resp., A-SE-models) are simply called *SE-interpretations* (resp., *SE-models*), coinciding with the notions defined by Turner [2003], and we write $SE(P)$ instead of $SE^{\mathcal{U}}(P)$.

Proposition 1 ([Woltran, 2004]) *Two programs P and Q are strongly equivalent relative to A iff $SE^A(P) = SE^A(Q)$.*

Example 1 *Consider the following two programs, P_1 and P_2, which we shall use as a running example:*

$$P_1 = P_0 \cup \{c \vee d \leftarrow a;\ c \vee d \leftarrow b\},$$
$$P_2 = P_0 \cup \{c \vee d \leftarrow a, b;\ d \leftarrow b, not\, c;\ c \leftarrow a, not\, d\},$$

for $P_0 = \{a \leftarrow c;\ b \leftarrow c;\ a \leftarrow d;\ b \leftarrow d;\ \leftarrow not\, c, not\, d\}$. They have the following SE-models:[1]

$$SE(P_1) = \{(\emptyset, abc), (\emptyset, abd), (\emptyset, abcd), (abcd, abcd), (abc, abcd), (abd, abcd), (abc, abc), (abd, abd)\},$$
$$SE(P_2) = SE(P_1) \cup \{(b, abc), (a, abd), (b, abcd), (a, abcd)\}.$$

Hence, P_1 and P_2 are not strongly equivalent. On the other hand, $\mathcal{AS}(P_1) = \mathcal{AS}(P_2) = \emptyset$, i.e., P_1 and P_2 are (ordinarily) equivalent. Moreover, P_1 and P_2 are strongly equivalent relative to A iff $A \cap \{a, b\} = \emptyset$. For $A = \{a, b\}$, we get

$$SE^A(P_1) = \{(\emptyset, abc), (\emptyset, abd), (abc, abc), (abd, abd)\},$$
$$SE^A(P_2) = SE^A(P_1) \cup \{(b, abc), (a, abd)\}.$$

3 General Framework

In order to deal with differing notions of program equivalence in a uniform manner, taking in particular the currently existing notions of equivalence, as presented above, as well as

[1] We write abc instead of $\{a, b, c\}$, a instead of $\{a\}$, etc.

equivalence notions based on the projection of answer sets into account, we introduce a general framework for expressing solution correspondences between logic programs. In this framework, we parameterize, on the one hand, the set R of rules to be added to the programs P and Q, and, on the other hand, the relation that has to hold between the collection of answer sets of $P \cup R$ and $Q \cup R$. Concerning the latter parameter, besides equality, other comparison relations like set-inclusion may be used. This leads to the following notion:

Definition 1 *By a* correspondence frame*, or simply* frame*, $\mathcal{F}$, we understand a triple $(\mathcal{U}, \mathcal{C}, \rho)$, where (i) $\mathcal{U}$ is a set of atoms, called the* universe of *$\mathcal{F}$, (ii) $\mathcal{C} \subseteq \mathcal{P}_{\mathcal{U}}$, called the* context programs of *$\mathcal{F}$, or simply the* context*, and (iii) $\rho \subseteq 2^{2^{\mathcal{U}}} \times 2^{2^{\mathcal{U}}}$.*

For all programs $P, Q \in \mathcal{P}_{\mathcal{U}}$, we say that P and Q are $\mathcal{F}$-corresponding*, symbolically $P \simeq_{\mathcal{F}} Q$, iff, for all $R \in \mathcal{C}$, $(\mathcal{AS}(P \cup R), \mathcal{AS}(Q \cup R)) \in \rho$.*

Intuitively, in a correspondence frame $\mathcal{F} = (\mathcal{U}, \mathcal{C}, \rho)$, $\mathcal{U}$ determines the general alphabet under consideration, $\mathcal{C}$ determines the kind of rules used for comparison, and ρ is the specific operation used for checking the correspondence of two programs.

It is quite obvious that the equivalence notions discussed above are special cases of $\mathcal{F}$-correspondence. Indeed, for any universe $\mathcal{U}$ and any $A \subseteq \mathcal{U}$, we have that strong equivalence relative to A coincides with $(\mathcal{U}, \mathcal{P}_A, =)$-correspondence, and uniform equivalence relative to A coincides with $(\mathcal{U}, 2^A, =)$-correspondence. Consequently, it holds that (i) strong equivalence coincides with $(\mathcal{U}, \mathcal{P}_{\mathcal{U}}, =)$-correspondence, (ii) uniform equivalence coincides with $(\mathcal{U}, 2^{\mathcal{U}}, =)$-correspondence, and (iii) ordinary equivalence coincides with $(\mathcal{U}, \{\emptyset\}, =)$-correspondence.

In this paper, we are mainly concerned with correspondence frames of form $(\mathcal{U}, \mathcal{P}_A, \subseteq_B)$ and $(\mathcal{U}, \mathcal{P}_A, =_B)$, where $A \subseteq \mathcal{U}$ is some set of atoms, and $\subseteq_B$ and $=_B$ are projections of the standard subset and set-equality relation, respectively, to a set $B \subseteq \mathcal{U}$, defined as follows: for sets $\mathcal{S}, \mathcal{S}'$ of interpretations,

- $\mathcal{S} \subseteq_B \mathcal{S}'$ iff $\mathcal{S}|_B \subseteq \mathcal{S}'|_B$, and
- $\mathcal{S} =_B \mathcal{S}'$ iff $\mathcal{S}|_B = \mathcal{S}'|_B$.

In particular, $=_B$ amounts to answer-set existence if $B = \emptyset$, and to correspondence between answer sets if $B = \mathcal{U}$.

In what follows, if $\mathcal{F}$ is of the form $(\mathcal{U}, \mathcal{C}, =_B)$, we refer to $\mathcal{F}$ also as an *equivalence frame*, and, accordingly, to $\mathcal{F}$-correspondence also as *$\mathcal{F}$-equivalence*. As well, a frame of the form $(\mathcal{U}, \mathcal{C}, \subseteq_B)$ is also referred to as an *inclusion frame*.

For later purposes, we also introduce the following notion: A *correspondence problem, Π, over $\mathcal{U}$*, is a quadruple $(P, Q, \mathcal{C}, \rho)$, where $P, Q \in \mathcal{P}_{\mathcal{U}}$ and $(\mathcal{U}, \mathcal{C}, \rho)$ is a frame. We say that Π *holds* iff $P \simeq_{(\mathcal{U},\mathcal{C},\rho)} Q$ holds. In accord with the above designations, we call Π an *equivalence problem* if $(\mathcal{U}, \mathcal{C}, \rho)$ is an equivalence frame and an *inclusion problem* if $(\mathcal{U}, \mathcal{C}, \rho)$ is an inclusion frame.

For a correspondence problem $\Pi = (P, Q, \mathcal{C}, \rho)$ over $\mathcal{U}$, we usually leave $\mathcal{U}$ implicit, assuming that it consists of all atoms occurring in P, Q, and $\mathcal{C}$.

We next list some basic properties of $\mathcal{F}$-equivalence.

Proposition 2 *Let $(\mathcal{U}, \mathcal{C}, =_B)$ be an equivalence frame and $P, Q \in \mathcal{P}_{\mathcal{U}}$. Then, the following conditions hold:*

1. *If $P \simeq_{(\mathcal{U},\mathcal{C},=_B)} Q$, then $P \simeq_{(\mathcal{U},\mathcal{C}',=_{B'})} Q$, for all $\mathcal{C}' \subseteq \mathcal{C}$ and all $B' \subseteq B$.*
2. *$P \simeq_{(\mathcal{U},\mathcal{C},=_B)} Q$ iff $P \simeq_{(\mathcal{U},\mathcal{C},\subseteq_B)} Q$ and $Q \simeq_{(\mathcal{U},\mathcal{C},\subseteq_B)} P$.*

We recall an important result due to Woltran [2004], extending an analogous result by Lifschitz *et al.* [2001].

Proposition 3 *Consider a frame $(\mathcal{U}, \mathcal{P}_A, =)$, for $A \subseteq \mathcal{U}$, and let $P, Q \in \mathcal{P}_{\mathcal{U}}$. Then, $P \simeq_{(\mathcal{U},\mathcal{P}_A,=)} Q$ iff $P \simeq_{(\mathcal{U},\mathcal{P}^{un}_A,=)} Q$.*

Example 2 *We have already seen that for P_1, P_2 from Example 1, $P_1 \not\simeq_{(\mathcal{U},\mathcal{P}_A,=)} P_2$ holds, for $A = \{a, b\}$ and $\mathcal{U} = \{a, b, c, d\}$. Hence, by Proposition 3, $P_1 \not\simeq_{(\mathcal{U},\mathcal{P}^{un}_A,=)} P_2$. This is witnessed by the fact that, e.g., $\mathcal{AS}(P_1 \cup \{a\}) = \{abc, abd\}$ while $\mathcal{AS}(P_2 \cup \{a\}) = \{abc\}$.*

The relevance of Proposition 3 is that it allows to drastically reduce the number of required rules for equivalence checking. However, the proposition does not generalize to projections of answer sets.

Theorem 1 *For any equivalence frame $(\mathcal{U}, \mathcal{P}_A, =_B)$ with $A, B \subseteq \mathcal{U}$, and any $P, Q \in \mathcal{P}_{\mathcal{U}}$, if $P \simeq_{(\mathcal{U},\mathcal{P}_A,=_B)} Q$, then $P \simeq_{(\mathcal{U},\mathcal{P}^{un}_A,=_B)} Q$, but the converse does not hold in general.*

Proof. That $P \simeq_{(\mathcal{U},\mathcal{P}_A,=_B)} Q$ implies $P \simeq_{(\mathcal{U},\mathcal{P}^{un}_A,=_B)} Q$ is immediate from Part 1 of Proposition 2. Our running example shows the failure of the converse: $P_1 \simeq_{(\mathcal{U},\mathcal{P}^{un}_A,=_B)} P_2$ holds, where $\mathcal{U} = \{a, b, c, d\}$ and $A = B = \{a, b\}$, but $P_1 \not\simeq_{(\mathcal{U},\mathcal{P}_A,=_B)} P_2$. The latter holds in view of $\mathcal{AS}(P_1 \cup \{a \vee b \leftarrow\}) = \{abc, abd\}$ but $\mathcal{AS}(P_2 \cup \{a \vee b \leftarrow\}) = \emptyset$, as easily shown. $\Box$

The above theorem also holds if the set $\mathcal{P}^{un}_A$ is substituted by the class of all normal programs over A. Indeed, programs P_1 and P_2 of our running example are corresponding with respect to a frame using normal programs over $A = \{a, b\}$ as context, because in each stable model of $P_1 \cup R$ resp. $P_2 \cup R$, both a and b must be true. Thus, rules in R $(\subseteq \mathcal{P}_A)$ with negative literals in the body are immaterial.

Theorem 1 indicates that equivalence for projected answer sets is more involved. The same holds for inclusion, since as an easy corollary to Theorem 1, $P \simeq_{(\mathcal{U},\mathcal{P}_A,\subseteq_B)} Q$ implies $P \simeq_{(\mathcal{U},\mathcal{P}^{un}_A,\subseteq_B)} Q$, but not vice versa. In fact, the next result shows that, in general, a smallest extension R violating inclusion has exponential size.

Theorem 2 *There exists a family of problems $\Pi = (P, Q, \mathcal{P}_A, \subseteq_A)$ such that each $R \subseteq \mathcal{P}_A$ witnessing $\mathcal{AS}(P \cup R)|_A \not\subseteq \mathcal{AS}(Q \cup R)|_A$ is exponential in the size of P and Q.*

Proof (Sketch). The idea is to encode a propositional CNF $\phi = \bigwedge_{i=1}^{n} C_i$ over atoms V, for $C_i = c_{i,1} \vee \cdots \vee c_{i,k_i}$, into $\Pi = (P, Q, \mathcal{P}_A, \subseteq_A)$ such that, for any $R \in \mathcal{P}_A$, $\mathcal{AS}(P \cup R)|_A \not\subseteq \mathcal{AS}(Q \cup R)|_A$ and $\neg\tilde{R}$ must include a DNF for ϕ, where $\tilde{R}$ is the result of interpreting R as a classical formula.

Let $\bar{V} = \{\bar{v} \mid v \in V\}$, $V' = \{v' \mid v \in V\}$, $\bar{V}' = \{\bar{v}' \mid v \in V\}$, and $G = \{g_1, \ldots, g_n\}$ be sets of new atoms. We define

$$
\begin{aligned}
P \;=\; & \{v \vee \bar{v} \leftarrow;\; \leftarrow \mathit{not}\, v;\; \leftarrow \mathit{not}\, \bar{v} \mid v \in V\} \cup \\
& \{v \leftarrow u, \bar{u};\; \bar{v} \leftarrow u, \bar{u} \mid v, u \in V\} \cup \\
& \{v \leftarrow C_i^*;\; \bar{v} \leftarrow C_i^* \mid v \in V;\; 1 \leq i \leq n\},
\end{aligned}
$$

where $C_i^* = c_{i,1}^*, \ldots, c_{i,k_i}^*$, $v^* = \bar{v}$, and $(\neg v)^* = v$, and

$$\begin{aligned} Q \;=\; & \{v \vee \bar{v} \leftarrow;\; v' \leftarrow \bar{v}, not\, \bar{v}';\; \bar{v}' \leftarrow v, not\, v' \mid v \in V\} \cup \\ & \{\leftarrow v', \bar{v}';\; \leftarrow not\, v', not\, \bar{v}' \mid v \in V\} \cup \\ & \{v \leftarrow u';\; \bar{v} \leftarrow u';\; v \leftarrow \bar{u}';\; \bar{v} \leftarrow \bar{u}' \mid v, u \in V\} \cup \\ & \{g_i \leftarrow \widehat{c}_{i,j} \mid i = 1..n, j = 1..k_i\} \cup \{\leftarrow g_1, \ldots, g_n\}, \end{aligned}$$

with $\widehat{v} = v'$ and $\widehat{\neg v} = \bar{v}'$. Let $A = V \cup \bar{V}$. Informally, $P \cup R$ only admits answer sets containing A, and indeed it holds that $(A, A) \in SE(P \cup R)|_A \setminus SE(Q \cup R)|_A$, for some R. For any such R, the SE-models (X, A) with $X \subset A$ must be precisely those where X is a countermodel of ϕ. This means that $\tilde{R}^A$ is a CNF for $\neg\phi$, and thus $\neg\tilde{R}^A$ amounts to a DNF for ϕ. This proves the claim. Now, as well-known, the smallest DNF for a CNF ϕ can be exponential in ϕ, which proves the result. □

We can similarly construct a family of problems showing that a smallest extension R violating equivalence has exponential size. Hence, a naive guess and check algorithm to disprove inclusion or equivalence needs exponential space. For a better algorithm, we have to develop suitable semantical characterizations, which must take disjunctive extensions into account.

4 Characterizations

In this section, we first present some characteristic structures associated with inclusion and equivalence problems under projections of answer sets, termed *spoilers* and *certificates*. Based on them, we then discuss some interesting invariance results.

We start with some general properties.

Definition 2 *A set $\mathcal{S}$ of SE-interpretations is* complete *iff, for each $(X, Y) \in \mathcal{S}$, also $(Y, Y) \in \mathcal{S}$ as well as $(X, Z) \in \mathcal{S}$, for any $Y \subseteq Z$ with $(Z, Z) \in \mathcal{S}$.*

It can be shown that the set $SE(P)$ of all SE-models of a program P is always complete.

The following guarantees that a complete set $\mathcal{S}$ of SE-interpretations can be represented by some program P.

Proposition 4 *Let $\mathcal{S}$ be a complete set of SE-interpretations, and let A be a set of atoms. Then, there exists a program $P_{\mathcal{S},A} \in \mathcal{P}_A$ such that $SE(P_{\mathcal{S},A})|_A = \mathcal{S}|_A$.*

One possibility to obtain $P_{\mathcal{S},A}$ from $\mathcal{S}$ is as follows: take rules $\leftarrow Y, not\,(A \setminus Y)$, for each $Y \subseteq A$ such that $(Y, Y) \notin \mathcal{S}|_A$, and rules $\bigvee_{p \in (Y \setminus X)} p \leftarrow X, not\,(A \setminus Y)$, for each $X \subset Y$ such that $(X, Y) \notin \mathcal{S}|_A$ and $(Y, Y) \in \mathcal{S}|_A$.

4.1 Spoilers

The first class of characteristic structures associated with program correspondence we are dealing with are of such a nature that their existence prevents the equivalence of programs under projected answer sets.

We need the following auxiliary notation: Let $\mathcal{S}$ be a set of SE-interpretations and Y, C sets of atoms. Then, $\sigma_Y^C(\mathcal{S}) = \{(X, Z) \in \mathcal{S} \mid Z|_C = Y|_C\}$.

Definition 3 *Let $\Pi = (P, Q, \mathcal{P}_A, \subseteq_B)$ be an inclusion problem over $\mathcal{U}$, let $Y \subseteq \mathcal{U}$ be an interpretation, and consider $\mathcal{S} \subseteq \sigma_Y^{A \cup B}(SE^A(Q))$. The pair $(Y, \mathcal{S})$ is a* spoiler for Π *iff*

(i) $(Y, Y) \in SE^A(P)$,

(ii) for each $(Z, Z) \in \mathcal{S}$, some non-total $(X, Z) \in S$ exists,

(iii) $(Z, Z) \in \mathcal{S}$ iff $(Z, Z) \in \sigma_Y^{A \cup B}(SE^A(Q))$, and

(iv) $(X, Z) \in \mathcal{S}$ implies $(X, Y) \notin SE^A(P)$.

Intuitively, in a spoiler $(Y, \mathcal{S})$, the interpretation Y is an answer set of $P \cup R$ but not of $Q \cup R$, for some R, which is semantically given by $\mathcal{S}$.

Example 3 *For P_1 and P_2 from our running example and $A = \{a, b\}$, $(Y_1, \mathcal{S})$ and $(Y_2, \mathcal{S})$ are the only spoilers for $(P_1, P_2, \mathcal{P}_A, \subseteq_A)$, where $Y_1 = \{abc\}$, $Y_2 = \{abd\}$, and $\mathcal{S} = \{(a, abd), (b, abc), (abc, abc), (abd, abd)\}$, with the latter being a subset of $\sigma_{Y_1}^A(SE^A(P_2)) = \sigma_{Y_2}^A(SE^A(P_2)) = S \cup \{(\emptyset, abc), (\emptyset, abd)\}$, as required in Definition 3.*

The central property of spoilers is as follows:

Theorem 3 *Let $\mathcal{F} = (\mathcal{U}, \mathcal{P}_A, \subseteq_B)$ be a frame. Then, for any $P, Q \in \mathcal{P}_\mathcal{U}$, $P \simeq_\mathcal{F} Q$ iff there is no spoiler for $(P, Q, \mathcal{P}_A, \subseteq_B)$.*

An immediate consequence of this theorem, together with Part 2 of Proposition 2, is the following result:

Corollary 1 *Let $\mathcal{F} = (\mathcal{U}, \mathcal{P}_A, =_B)$ be an equivalence frame and $P, Q \in \mathcal{P}_\mathcal{U}$. Then, $P \simeq_\mathcal{F} Q$ iff neither $(P, Q, \mathcal{P}_A, \subseteq_B)$ nor $(Q, P, \mathcal{P}_A, \subseteq_B)$ has a spoiler.*

As discussed later on, spoilers provide a semantical basis for counterexample generation.

4.2 Certificates

After having introduced structures which *disprove* program correspondence, we now discuss structures which *prove* program correspondence. Roughly speaking, the structures introduced below express the essence of a program P, with respect to program equivalence, in terms of a semantic condition on P alone.

Definition 4 *Let C be a set of atoms and $\mathcal{S}$ a set of SE-interpretations. A pair $(\mathcal{X}, Y)$, where $\mathcal{X}$ is a set of interpretations and $Y \subseteq C$, is a C*-projection of *$\mathcal{S}$ iff there exists some set Z such that (i) $(Z, Z) \in \mathcal{S}$, (ii) $Z|_C = Y|_C$, and (iii) $\mathcal{X} = \{X \mid (X, Z) \in \mathcal{S}, X \subset Z\}$.*

For a program P, we call an $(A \cup B)$-projection of $SE^A(P)$ an (A, B)-certificate of *P.*

The following lemma can be shown by means of spoilers, and expresses that programs are corresponding with respect to inclusion frames iff their certificates satisfy a certain containment relation.

Lemma 1 *Correspondence of $(P, Q, \mathcal{P}_A, \subseteq_B)$ holds, iff for each (A, B)-certificate $(\mathcal{X}, Y)$ of P, an (A, B)-certificate $(\mathcal{X}', Y)$ of Q exists with $\mathcal{X}' \subseteq \mathcal{X}$.*

The next result expresses the central property of certificates. Towards its formulation, we require a further concept: An (A,B)-certificate $(\mathcal{X},Y)$ of a program P is *minimal* iff, for any (A,B)-certificate $(\mathcal{Z},Y)$ of P, $\mathcal{Z} \subseteq \mathcal{X}$ implies $\mathcal{Z} = \mathcal{X}$.

Theorem 4 *Let $\mathcal{F} = (\mathcal{U}, \mathcal{P}_A, =_B)$ be an equivalence frame. Then, for any $P, Q \in \mathcal{P}_\mathcal{U}$, $P \simeq_\mathcal{F} Q$ iff the minimal (A, B)-certificates of P and Q coincide.*

Note that this result is the pendant to Proposition 1, which deals with a model-theoretic characterization of relativized strong equivalence. Indeed, two programs $P, Q \in \mathcal{P}_\mathcal{U}$ are strongly equivalent relative to A iff their minimal $(A,\mathcal{U})$-certificates coincide. Some further relations between program correspondence and relativized and non-relativized strong equivalence, respectively, are given in the next subsection.

Example 4 *In our running example with $A = B = \{a, b\}$, we get that P_1 has a single (A, A)-certificate, $(\{\emptyset\}, \{ab\})$, while P_2 has two (A, A)-certificates, $(\{\emptyset, \{a\}\}, \{ab\})$ and $(\{\emptyset, \{b\}\}, \{ab\})$, all of them minimal. Since they do not coincide, we obtain that P_1 and P_2 are not $(\mathcal{U}, \mathcal{P}_A, =_A)$-equivalent, as expected.*

4.3 Invariance Results

Theorem 4 allows us to derive some interesting invariance results with respect to varying projection sets B.

Theorem 5 *Let $\mathcal{U}$ be a set of atoms and $A, B \subseteq \mathcal{U}$. Then, for any $P, Q \in \mathcal{P}_\mathcal{U}$, $P \simeq_{(\mathcal{U},\mathcal{P}_A,=_B)} Q$ iff $P \simeq_{(\mathcal{U},\mathcal{P}_A,=_{A\cup B})} Q$.*

This result follows immediately from Theorem 4 and Definition 4, by observing that (A, B)-certificates of a program P and $(A, A\cup B)$-certificates of P are actually identical objects, since $(A \cup B)$-projections of $SE^A(P)$ trivially coincide with $(A \cup (A \cup B))$-projections of $SE^A(P)$.

Theorem 5 has several interesting consequences.

Corollary 2 *For programs P, Q and any set A of atoms, $(P, Q, \mathcal{P}_A, =_\emptyset)$ holds iff $(P, Q, \mathcal{P}_A, =_A)$ holds.*

That is, answer-set existence (which is relevant regarding Boolean properties) relative to additions from $\mathcal{P}_A$ is tantamount to relativized strong equivalence under projection to A.

Corollary 3 *Let $\mathcal{U}$ be a set of atoms. Then, for any programs $P, Q \in \mathcal{P}_\mathcal{U}$ and any set $B \subseteq \mathcal{U}$ of atoms, $P \simeq_{(\mathcal{U},\mathcal{P}_\mathcal{U},=_B)} Q$ iff P and Q are strongly equivalent.*

This result is quite striking as it shows that strong equivalence corresponds to $(\mathcal{U}, \mathcal{P}_\mathcal{U}, =_B)$-equivalence, for *any* projection set B. It is derived from the fact that, for any B, the $(\mathcal{U}, B)$-certificates of a program P are in a one-to-one correspondence to the SE-models of P as follows: $(\{X_1, \ldots, X_m\}, Y)$ is a $(\mathcal{U}, B)$-certificate of P iff $(X_1, Y), \ldots, (X_m, Y), (Y, Y)$ are all the SE-models of P with fixed second component Y.

In particular, consistency under answer-set semantics (i.e., if $B = \emptyset$) coincides with strong equivalence. More generally:

Corollary 4 *Let $\mathcal{U}$ be a set of atoms. Then, for all programs $P, Q \in \mathcal{P}_\mathcal{U}$ and all sets A and B of atoms such that $A \cup B = \mathcal{U}$, $P \simeq_{(\mathcal{U},\mathcal{P}_A,=_B)} Q$ iff P and Q are strongly equivalent relative to A.*

In fact, in this setting, a correspondence between A-SE-models of a program and its (A, B)-certificates is analogously established as above.

5 Counterexamples

Given that a correspondence problem $\Pi = (P, Q, \mathcal{P}_A, \subseteq_B)$ does not hold, it is interesting to know why this is the case. We define a *counterexample for* Π as a pair (R, M), where

- $R \in \mathcal{P}_A$ such that $\mathcal{AS}(P \cup R) \not\subseteq_B \mathcal{AS}(Q \cup R)$, and
- $M \in \mathcal{AS}(P \cup R)$ and $M|_B \notin \mathcal{AS}(Q \cup R)|_B$.

Furthermore, a counterexample for an equivalence problem $(P, Q, \mathcal{P}_A, =_B)$ is any counterexample for either $(P, Q, \mathcal{P}_A, \subseteq_B)$ or $(Q, P, \mathcal{P}_A, \subseteq_B)$.

Our notion of a spoiler from Definition 3 provides a basis for such counterexamples.

Theorem 6 *Suppose $(Y, \mathcal{S})$ is a spoiler for a correspondence problem $\Pi = (P, Q, \mathcal{P}_A, \subseteq_B)$. Then, $(P_{\mathcal{S},A}, Y)$ is a counterexample for Π, where $P_{\mathcal{S},A}$ is as in Proposition 4.*

This result follows from Theorem 3 and Proposition 4 by the fact that $\mathcal{S}|_A$ is complete for any spoiler $(Y, \mathcal{S})$.

Example 5 *For our example and the sketched construction after Proposition 4, we derive counterexamples $(R, \{abc\})$ and $(R, \{abd\})$, where $R = \{a \vee b \leftarrow;\ \leftarrow not\, a; \leftarrow not\, b; \leftarrow not\, a, not\, b\}$; obviously, the last rule is redundant.*

Note that if an inclusion problem $(P, Q, \mathcal{P}_A, \subseteq_B)$ fails, some counterexample as in Theorem 6 does exist.

We observe that the programs in the counterexamples of Theorem 6 may contain redundant clauses, as succinctness is not a concern of spoilers. For instance, in our example, $R' = \{a \vee b \leftarrow\}$ would yield a simpler counterexample. In fact, spoilers are not geared towards providing minimal counterexamples with respect to particular syntactic subclasses of contexts.

Towards facilitating special counterexamples, we may extend the notion of a spoiler to pairs $(Y, \mathcal{S})$, where $\mathcal{S} \not\subseteq \sigma_Y^{A\cup B}(SE^A(Q))$ is admitted for complete $\mathcal{S}$, by replacing in Definition 3 the set $\mathcal{S}$ in (ii) with $\mathcal{S} \cap \sigma_Y^B(SE^A(Q))$ and in (iii) and (iv) with $\mathcal{S} \cap \sigma_Y^{A\cup B}(SE^A(Q))$, calling the result an *extended spoiler*.

Example 6 *In our running example, $\Pi = (P_1, P_2, \mathcal{P}_A, \subseteq_A)$ has spoilers $(\{abc\}, \mathcal{S})$ and $(\{abd\}, \mathcal{S})$, with $\mathcal{S} = \{(a, abd), (b, abc), (abc, abc), (abd, abd)\}$. Since both $(ab, ab) \notin SE^A(P_2)$ and $(abcd, abcd) \notin SE^A(P_2)$, one can verify that any complete superset $\mathcal{S}'$ of $\mathcal{S}$ not containing any $(\emptyset, Z)$, with $Z|_A = \{ab\}$, yields extended spoilers $(\{abc\}, \mathcal{S}')$ and $(\{abd\}, \mathcal{S}')$. In particular, we may set $\mathcal{S}' = \mathcal{S} \cup \{(a, a), (b, b)\}$. Note that $\mathcal{S}'|_A = \{(a, a), (a, ab), (b, b), (b, ab), (ab, ab)\}$. Now, the simpler program $R' = \{a \vee b \leftarrow\}$ fulfills $SE(R')|_A = \mathcal{S}'|_A$; thus, $(R', \{abc\})$ and $(R', \{abd\})$ are counterexamples for Π.*

Theorems 3 and 6 generalize to extended spoilers. More counterexamples can be constructed using such spoilers in general, which may include a counterexample of particular form. In our example, adding $\{(a, a), (b, b)\}$ to an ordinary spoiler allowed us to give a counterexample program which is positive. On the other hand, we can show that no counterexample program which is normal exists (as no $(\emptyset, Z)$ with $Z|_A = \{ab\}$ can be added). An elaboration of this issue remains for further work.

6 Computational Complexity

Our first result is concerned with recognizing whether an interpretation is a discriminating answer set for the extended programs $P \cup R$ and $Q \cup R$, i.e., a "partial" spoiler.

Lemma 2 *Given programs P and Q, sets A and B of atoms, and an interpretation Y, deciding whether $(P, Q, \mathcal{P}_A, \subseteq_B)$ has some spoiler whose first component is Y is Π_3^P-complete.*

Proof (*Sketch*). We show Σ_3^P-membership of deciding that no spoiler of form $(Y, \mathcal{S})$ exists. By Definition 3, it suffices to check whether (a) $(Y, Y) \notin SE^A(P)$ or (b) whether there exists a $(Z, Z) \in \sigma_Y^{A \cup B}(SE^A(Q))$ such that each non-total $(X, Z) \in SE^A(Q)$ implies $(X, Y) \in SE^A(P)$. Part (a) is feasible in polynomial time with an NP oracle. For Part (b), note that given Y and Z, checking whether, for each non-total $(X, Z) \in SE^A(Q)$ also $(X, Y) \in SE^A(P)$, is in Π_2^P. Therefore, Part (b) is in Σ_3^P, and the entire test is in Σ_3^P.

The Π_3^P-hardness is shown by a sophisticated reduction from suitable quantified Boolean formulas (QBFs), using machinery from the exponential counterexample construction which was used for showing Theorem 2. □

From this result, we can easily derive the membership part of our main complexity result given below; its hardness part is again shown by an encoding of QBFs.

Theorem 7 *Given programs P and Q and sets A and B of atoms, deciding whether $(P, Q, \mathcal{P}_A, \subseteq_B)$ holds is Π_4^P-complete. Moreover, Π_4^P-hardness holds even for $B = \emptyset$, i.e., for answer-set existence.*

For the particular case where $B = A$, which constitutes the setting where auxiliary letters are used in logic programs, we obtain in combination with Theorem 5 the same complexity.

Notice that the "partial" spoiler of Lemma 2 avoids a naive guess of a (possibly exponentially large) full spoiler $(Y, \mathcal{S})$ which proves the failure of $(P, Q, \mathcal{P}_A, \subseteq_B)$, at the expense of checking involved conditions on $SE^A(P)$ and $SE^A(Q)$.

The above results for inclusion problems carry over to equivalence problems, since they are polynomially intertranslatable, as seen by Part 2 of Proposition 2 and the following fact:

Proposition 5 $P \simeq_{(\mathcal{U}, \mathcal{P}_A, \subseteq_B)} Q$ *iff* $Q \simeq_{(\mathcal{U}, \mathcal{P}_A, =_B)} L_{P,Q}$, *where* $L_{P,Q} = \{g_P \vee g_Q \leftarrow; \ \leftarrow g_P, g_Q\} \cup \{H \leftarrow g_R, B \mid R \in \{P, Q\}, H \leftarrow B \in R\}$ *and* g_P, g_Q *are new atoms.*

This holds by virtue of Lemma 1 and Theorem 4, and the fact that the (A, B)-certificates of $L_{P,Q}$ are those of P and Q.

If the size of each program in the context $\mathcal{C}$ is polynomially bound by the size of the compared programs P and Q (as is the case for ordinary and uniform equivalence), the complexity is lower. Let us call such problems $(P, Q, \mathcal{C}, \rho)$ *bound*.

Theorem 8 *Given programs P and Q, a context $\mathcal{C}$, a set B of atoms, such that $\Pi = (P, Q, \mathcal{C}, \subseteq_B)$ is bound, deciding whether Π holds is Π_3^P-complete. Π_3^P-hardness holds even for $\mathcal{C} = \{\emptyset\}$, i.e., for ordinary equivalence with projection.*

For other instances of the framework, the complexity is even lower.

Theorem 9 *Given programs P and Q over $\mathcal{U}$ and sets A and B of atoms, deciding whether $(P, Q, \mathcal{P}_A, =_B)$ holds is* (*i*) coNP*-complete if $A = \mathcal{U}$, and* (*ii*) Π_2^P*-complete if $(A \cup B) = \mathcal{U}$.*

This result follows from the invariance results in Section 4.2 and complexity results due to Woltran [2004].

7 Conclusion and Further Work

We have presented a general framework for expressing solution correspondences between nonmonotonic logic programs, and have then developed semantic characterizations of inclusion and equivalence problems under projected answer sets within a context of possible changes. As we have shown, they match the intrinsic complexity of the problem.

Our results provide a semantical basis for developing optimization and debugging techniques, which are lacking at present but vital for further enhancements of ASP as a programming paradigm.

Several issues remain for future work. One is to extend our study to different classes of contexts and compared programs, and to provide suitable semantical and complexity characterizations. Another issue concerns the construction of "good" counterexamples, according to their possible form. Finally, exploring other notions of correspondences than $=_B$ and $\subseteq_B$ in the general framework is an intriguing issue.

References

[Bonatti, 2001] P.A. Bonatti. Reasoning with open logic programs. In *Proc. LPNMR 2001*, pp. 147–159.

[Eiter and Fink, 2003] T. Eiter and M. Fink. Uniform equivalence of logic programs under the stable model semantics. In *Proc. ICLP 2003*, pp. 224–238.

[Eiter *et al.*, 2004] T. Eiter, M. Fink, H. Tompits, and S. Woltran. Simplifying logic programs under uniform and strong equivalence. In *Proc. LPNMR-7*, pp. 87–99.

[Gelfond and Leone, 2002] M. Gelfond and N. Leone. Logic Programming and Knowledge Representation - The A-Prolog Perspective. *AIJ*, 138(1-2):3–38, 2002.

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

[Inoue and Sakama, 2004] K. Inoue and C. Sakama. Equivalence of logic programs under updates. In *Proc. JELIA 2004*, pp. 174–186.

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

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

[Oikarinen and Janhunen, 2004] E. Oikarinen and T. Janhunen. Verifying the equivalence of logic programs in the disjunctive case. In *Proc. LPNMR-7*, pp. 180–193.

[Pearce and Valverde, 2004] D. Pearce and A. Valverde. Synonymous theories in answer set programming and equilibrium logic. In *Proc. ECAI 2004*, pp. 388–392.

[Turner, 2003] H. Turner. Strong equivalence made easy: Nested expressions and weight constraints. *Theory & Practice of Logic Programming*, 3(4-5):602–622, 2003.

[Woltran, 2004] S. Woltran. Characterizations for relativized notions of equivalence in answer set programming. In *Proc. JELIA 2004*, pp. 161–173.