FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Characterising Relativised Strong Equivalence with Projection for Non-Ground Logic Programs

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Software & Information Engineering

eingereicht von

Tobias Geibinger
Matrikelnummer 01427138

an der Fakultät für Informatik
der Technischen Universität Wien

Betreuung: a.o. Univ.-Prof. Dr. Hans Tompits

Wien, 6. September 2018

Tobias Geibinger	Hans Tompits

Erklärung zur Verfassung der Arbeit

Tobias Geibinger

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 6. September 2018

Tobias Geibinger

Acknowledgements

First, I would like to thank Hans Tompits for supervising this thesis. I would never have been able to finish this work without his advice and constant help. Furthermore, I would like to thank my girlfriend Marlies Grossberger for her support and advice. Finally, I am eternally grateful to my mother Hermine Geibinger and my late father Werner Geibinger, who have supported me all my life.

Abstract

This thesis deals with advanced notions of equivalence between logic programs under answer-set semantics. Particularly, we generalise the concepts of *relativised strong equivalence* and *relativised strong equivalence with projection* to the non-ground case, i.e., to programs with variables.

In recent work, a framework for expressing equivalences between answer-set programs was introduced and later generalised for non-ground programs. These works use said framework to characterise a fine-grained version of *strong equivalence*. For two programs to be (*ordinarly*) *equivalent*, they just have to have the same answer sets. In contrast, two programs are *strongly equivalent* if it is possible to add the same arbitray rules to both of them and their answer sets still match. In other words, if the (sub)programs P and Q are strongly equivalent, then P can be replaced with Q in any context program. The equivalence notion we are concerned with extends strong equivalence by enabling the restriction of the predicates of the context programs and the filtering of auxiliary predicates. Like already mentioned, this version of equivalence is called *relativised strong equivalence with projection* and was so far only studied for propositional (or ground) programs.

Aside from characterising relativised strong equivalence (with projection) for the non-ground case in the mentioned framework, we also provide model-theoretic characterisations for the non-ground setting. In particular, we introduce the generalization of a structure which captures relativised strong equivalence, namely the so-called *RSE-model*. Furthermore, we provide liftings of a *spoiler* and a *counterexample*, two structures used for checking relativised strong equivalence with projection between programs.

Contents

CHAPTER 1

Introduction

In contrast to Turing-complete programming languages, the question of program equivalence is not undecidable for some logic programming languages. We are interested in program equivalence, because it enables us to interchange equivalent program segments, similar to the *replacement theorem* in classical logic. In this thesis, we focus on the language of *disjunctive logic programs* (DLPs) under the *answer-set semantics*.

The core of *answer-set programming* (ASP) goes back to Gelfond and Lifschitz [1], and is syntactically very similar to other logic programming languages like *Prolog* or *Datalog*. In the last years, ASP has been subject to various extensions which aim at increasing expressiveness or tailor the language to specific problem domains. As already mentioned, in our work we focus on DLPs. Like ASP in general, DLPs are very useful in declarative problem solving. In ASP, the produced solutions are models ("answer sets" or "stable models") of the encoded problem and not proofs as in traditional logic-based formalisms. That and the high-level specification language offered by ASP can, for example, be leveraged to compute solutions for *NP-complete* search problems with very little time needed for problem encoding. Another import aspect in the case for ASP is the existence of sophisticated and efficent solvers, like `clasp` [2] or `DLV` [3]. For a thorough introduction to ASP, the reader is referred to the excellent primer by Eiter, Ianni, and Krennwallner [4].

Unlike for theories in classical logic, just having the same models (or in this case answer sets) is not enough to allow arbitrary replacements of programs in ASP. To illustrate, consider a program P and a subprogram $M \subseteq P$. What we want is to replace M with an "equivalent" program N such that P has the same answer sets as before, but like we already mentioned, it is not enough if N has the same answer sets as M. In order to solve this problem Lifschitz, Pearce, and Valverde [5] introduced the concept of *strong equivalence* which is very closely aligned to the replacement theorem. Our subprograms M and N would be considered strongly equivalent, if for any program R, the answer sets $M \cup R$ were the same as the answer sets of $N \cup R$. Lifschitz et al. only considered the propositional case, but the concept was later generalised to cover programs with variables

by Eiter, Fink, Tompits, and Woltran [6].
Based on the notion of strong quivalence, Woltran [7] introduced the more fine-grained concept of *relativised strong equivalence* for propositional DLPs. Basically, two programs M and N are strongly equivalent relative to an alphabet A if for any program R^A consisting only of predicates from A, the answer sets of $M \cup R^A$ match the answer sets of $M \cup R^A$. To put it in different terms: If the subprograms M and N are strongly equivalent relative to A and the surrounding program consists of predicates from A only, then they can be safely replaced.
In some cases, strong equivalence is too restrictive. For example, consider programs with auxiliary predicates or variables. Two programs might be in principle replaceable but their use of different auxiliary predicates keeps them from being strongly equivalent. To combat this, Eiter, Tompits, and Woltran [8] added *projection* to relativised strong equivalence. The projection filters out the auxiliary predicates in the comparison of the answer sets and thus restricts the equivalence to the important output predicates. Furthermore, they also introduced a *correspondence-checking framework* which can be used to characterise every notion of equivalence for propositional DLPs and was later lifted to the non-ground setting by Oetsch and Tompits [9].

The contribution of this work will be the generalisation of the mentioned relativised strong equivalence with projection to the non-ground setting. In practise most answer-set programs have variables, therefore the generalisation of the mentioned concepts is particularly interesting.

This thesis is structured the following way. The next chapter offers an introduction for the neccessary formal preliminaries of the work referenced above. In Chapter 3, we lift Woltran's [7] notion of relativised strong equivalence to the non-ground setting including their model based characterisations. Chapter 4 deals with generalising the correspondence-checking structures and concepts introduced by Eiter et al. [8]. Chapter 5 gives an overview about the related work. Finally, Chapter 6 offers a summary and an outlook to further work.

CHAPTER 2

Preliminaries

In this chapter, we introduce the neccessary preliminaries about logic programs under the *answer set semantics* and give a short introduction to program correspondence.

2.1 Logic Programs

Logic programs are defined over a vocabulary $\mathcal{V}$. A vocabulary $\mathcal{V}$ is a pair $(\mathbb{P}, \mathbb{D})$, where $\mathbb{P}$ is a set of *predicates* and $\mathbb{D}$ is a set of *constants* (also known as *domain* of $\mathcal{V}$). Each predicate in $\mathbb{P}$ has an *arity* $n \geq 0$. A logic program may also contain *variables*. Let $\mathcal{A}$ be the set of all variables contained in a program.
An *atom* is defined as $p(t_1, \ldots, t_n)$, where $p \in \mathbb{P}$ and $t_i \in \mathbb{D} \cup \mathcal{A}$, for $1 \leq i \leq n$. We call an atom *ground* if no variable occurs in it.
The set of all ground atoms of a vocabulary $\mathcal{V}$ is called the *Herbrand base* of $\mathcal{V}$, denoted by $HB_{\mathcal{V}}$. For a set of predicates $P \subseteq \mathbb{P}$ and a set of constants $C \subseteq \mathbb{D}$, the set of all ground atoms constructed by replacing the variables in P with constants of C is denoted by $HB_{P,C}$.

Definition 1 ([10]). *A* (disjunctive) rule r *has the following form:*

$$a_1 \vee \cdots \vee a_n \leftarrow b_1, \ldots, b_k, \mathit{not}\ b_{k+1}, \ldots, \mathit{not}\ b_m,$$

where $a_1, \ldots, a_n, b_1, \ldots, b_m$ *are atoms,* $n, m, k \geq 0$ *and* $n + m > 0$. *Furthermore, "not" denotes* default negation.

The *head* of r is the set $H(r) = \{a_1, \ldots, a_n\}$ and the *body* of r is denoted by $B(r) = \{b_1, \ldots, b_k, \mathit{not}\ b_{k+1}, \ldots, \mathit{not}\ b_m\}$. In addition we define $B^+(r) = \{b_1, \ldots, b_k\}$ and $B^-(r) = \{b_{k+1}, \ldots, b_m\}$.

There are different classifications of rules, as described by the following definition .

Definition 2. *A rule r is called*

(*1*) *a* fact, *if $m = 0$ and $n = 1$;*

(*2*) *a* constraint, *if $n = 0$;*

(*3*) normal, *if $n \leq 1$;*

(*4*) positive, *if $k = m$;*

(*5*) unary, *if $n = 1$ and $k = m \leq 1$;*

(*6*) Horn, *if $k = m$ and $n \leq 1$;*

(*7*) safe, *if each variable occuring in $H(r) \cup B^-(r)$ also occurs in $B^+(r)$; and*

(*8*) ground, *if all atoms in r are ground.*

A *program* is a set of rules. We call a program normal, positive, Horn, ground and/or safe if all of its rules are. From now own we assume that every program is safe.
The set of all constants appearing in a program P is called the *Herbrand universe* of P, symbolically HU_P. If no constant appears in P, then HU_P contains an arbitrary constant. Furthermore, the set of all predicates of P is denoted by $\mathcal{A}_P$. We define $HB_P := HB_{\mathcal{A}_P, HU_P}$ and $HB_{P,C} := HB_{\mathcal{A}_P, C}$.
Given a rule r and a set C of constants, we define $grd(r, C)$ as the set of all rules generated by replacing all variables of r with elements of C. For any program P, the *grounding of P with respect to C* is given by $grd(P, C) := \bigcup_{r \in P} grd(r, C)$. If P is a ground program, then $P = grd(P, C)$ for any C.

Example 1. *Consider the following program:*

$$P = \left\{ \begin{array}{l} accepted(x) \leftarrow applicant(x), not\ rejected(x), \\ rejected(x) \leftarrow applicant(x), not\ accepted(x), \\ applicant(jane) \leftarrow, \\ applicant(bob) \leftarrow \end{array} \right\}$$

The meaning of the program is simple. Every applicant is either accepted or rejected but never both nor neither.
As we mentioned above, the Herbrand universe of P is the set of all constants occuring in P, thus $HU_P = \{jane, bob\}$. Now, the Herbrand base of P can be obtained by enumerating all predicates of P and replacing each variable with each element in HU_P. This leads to the following Herbrand base

$$HB_P = \left\{ \begin{array}{l} applicant(jane), applicant(bob), accepted(jane), \\ accepted(bob), rejected(jane), rejected(bob) \end{array} \right\}.$$

Furthermore, the grounding of P over its Herbrand universe HU_P can be obtained in a similar fashion. We just have all variables of the ground P with elements of HU_P, thereby replacing all non-ground rules with possibly multiple new grounded rules.

$$grd(P, HU_P) = \left\{ \begin{array}{l} accepted(jane) \leftarrow applicant(jane), not\ rejected(jane), \\ accepted(bob) \leftarrow applicant(bob), not\ rejected(bob), \\ rejected(jane) \leftarrow applicant(jane), not\ accepted(jane), \\ rejected(bob) \leftarrow applicant(bob), not\ accepted(bob), \\ applicant(jane) \leftarrow, \\ applicant(bob) \leftarrow \end{array} \right\}.$$

A set of ground atoms is called an *interpretation*. Following the answer-set semantics for DLPs as given by Gelfond et al. [11], a ground rule r is *satisfied* by an interpretation I, denoted by $I \models r$, iff $H(r) \cap I \neq \emptyset$ whenever $B^+(r) \subseteq I$ and $B^-(r) \cap I = \emptyset$. For a ground program P, $I \models P$ iff each $r \in P$ is satisfied by I. The *Gelfond-Lifschitz reduct* [1] of a ground program P with respect to the interpretation I is given by

$$P^I := \{H(r) \leftarrow B^+(r) \ \mid \ r \in P,\ I \cap B^-(r) = \emptyset\}.$$

An interpretation I is an *answer set* of a non-ground program P iff I is a subset-minimal set satisfying $grd(P, HU_P)^I$. An alternate definition of answer sets is given the following way: An interpretation I is an answer set of a non-ground program P iff $I \models grd(P, HU_P)$ and $J \not\models grd(P, HU_P)^I$ for any $J \subset I$. We define $AS(P)$ as the set of all answer sets of P. For safe programs, it holds that if $I \in AS(P)$ then $I \subseteq HB_P$.

Example 2. *Recall our grounded program $grd(P, HU_P)$ from Example 1 and consider the interpretation*

$$I = \{applicant(jane), applicant(bob), accepted(jane), rejected(bob)\}.$$

The Gelfond-Lifschitz reduct of the program can easily be obtained by removing the two rules which contain "not accepted(jane)" and "not rejected(bob)", respectively:

$$grd(P, HU_P)^I = \left\{ \begin{array}{l} accepted(jane) \leftarrow applicant(jane), \\ rejected(bob) \leftarrow applicant(bob), \\ applicant(jane) \leftarrow, \\ applicant(bob) \leftarrow \end{array} \right\}$$

Now, since for each rule $r \in grd(P, HU_P)^I$, $B^+(r) \subseteq I$ implies $H(r) \cap I \neq \emptyset$ we obtain that I is a possible answer set of $grd(P, HU_P)$. Since there is no subset of I that satisfies all rules of $grd(P, HU_P)$, we gather that I is in fact an answer set and thus also an answer set of the non-ground P.

By $\mathcal{P}_\mathcal{V}^A$, we denote the set of all programs over $\mathcal{V}$ that contain only predicates of A. Similarily, by $\mathcal{F}_\mathcal{V}^A$ we denote the set of all sets of facts over $\mathcal{V}$ that only consist of predicates of A. If $A = \mathbb{P}$ we write $\mathcal{P}_\mathcal{V}$ and $\mathcal{F}_\mathcal{V}$.

2.2 Program Correspondence

Different notions of program equivalence exist for answer set programs. The most common ones are the following three.

Definition 3 ([12, 5]). *Logic Programs P and Q are*

(*i*) (ordinarily) equivalent *iff* $AS(P) = AS(Q)$;

(*ii*) unformly equivalent *iff* $AS(P \cup F) = AS(Q \cup F)$, *for any set of facts* F;

(*iii*) strongly equivalent *iff* $AS(P \cup R) = AS(Q \cup R)$, *for any program* R.

In this work, we are mainly interested in strong equivalence. Strong equivalence was introduced by Lifschitz, Pearce, and Valverde [5] and was characterised using Heyting's *logic of here-and-there*. In particular, two program where found to be strongly equivalent if and only if they had the same models in the logic of here-and-there.
Turner offered a different characterisation of strong equivalence [13]. He introduced the concept of an *SE-model*, which was then generalised for non-ground programs by Eiter et al. [6].

Definition 4 ([6]). *Let $\mathcal{V} = (\mathbb{P}, \mathbb{D})$ be a vocabulary, P a program over $\mathcal{V}$, $C \subseteq \mathbb{D}$, and let $X, Y \subseteq HB_{\mathbb{P},C}$ be sets of ground atoms. Then an* SE-interpretation *is a triple $(X, Y)_C$, such that $X \subseteq Y$. $(X, Y)_C$ is an* SE-model *of P iff $Y \models grd(P, C)$ and $X \models grd(P, C)^Y$. By $SE(P)$ we denote the set of all SE-models of P.*

For an interpretation Y we define $Y|_A := Y \cap A$ and for a triple $(I, J)_C$ we define $(I, J)_C|_A := (I|_A, J|_A)_C$, where A is a set. Furthermore, we define $S|_A := \{Y|_A \mid Y \in S\}$ where A again is a set and S is a set of interpretations or pairs of interpretations.
A triple $(I, J)_C$ is called *total* if $I = J$ and *non-total* otherwise. A set of pairs S is called non-total if for each total $(J, J)_C \in S$ there exists a non-total $(I, J)_C \in S$.
We use $Y \equiv^A_{\mathcal{V}} X$ as a shorthand for $Y|_{HB_{A,\mathbb{D}}} = X|_{HB_{A,\mathbb{D}}}$, where $\mathcal{V} = (\mathbb{P}, \mathbb{D})$ and $A \subseteq \mathbb{P}$. Furthermore, we use $Y \sqsubseteq^B_{\mathcal{V}} X$ as a shorthand for $Y|_{HB_{B,\mathbb{D}}} \subseteq X|_{HB_{B,\mathbb{D}}}$.

Following Turner's theorem for the propositional case, Eiter et al. showed that the following holds:

Theorem 1 ([6]). *Two (non-ground) logic programs P, Q are* strongly equivalent *iff $SE(P) = SE(Q)$.*

Example 3. *Recall the program P from Example 1 and consider the following program:*

$$Q = \left\{ \begin{array}{l} accepted(x) \vee rejected(x) \leftarrow applicant(x) \\ applicant(jane) \leftarrow, \\ applicant(bob) \leftarrow, \end{array} \right\}.$$

First, let us take a look at their answer sets.

$$AS(P) = AS(Q) = \left\{ \begin{array}{l} \{applicant(jane), applicant(bob), accepted(jane), accepted(bob)\}, \\ \{applicant(jane), applicant(bob), rejected(jane), rejected(bob)\}, \\ \{applicant(jane), applicant(bob), accepted(jane), rejected(bob)\}, \\ \{applicant(jane), applicant(bob), rejected(jane), accepted(bob)\} \end{array} \right\}.$$

From $AS(P) = AS(Q)$ it obviously follows that P and Q are ordinarily equivalent, but are they strongly equivalent as well? To figure that out, we compare their SE-models and use the following shorthands: $ap(\cdot)$ for $applicant(\cdot)$, $ac(\cdot)$ for $accepted(\cdot)$, $re(\cdot)$ for $rejected(\cdot)$, j for jane, and b for bob. Furthermore, we have the vocabulary $\mathcal{V} = (\mathbb{D}, \mathbb{P})$, where $\mathbb{D} = \{j, b\}$ and $\mathbb{P} = \{ap(\cdot), ac(\cdot), re(\cdot)\}$.

Then $SE(P)$ consists of elements $(X, Y)_C$ according to the following table:

X	Y	C
$\{ap(j), ap(b), ac(j), ac(b)\}$	$\{ap(j), ap(b), ac(j), ac(b)\}$	$\{j, b\}$
$\{ap(j), ap(b), re(j), re(b)\}$	$\{ap(j), ap(b), re(j), re(b)\}$	$\{j, b\}$
$\{ap(j), ap(b), ac(j), re(b)\}$	$\{ap(j), ap(b), ac(j), re(b)\}$	$\{j, b\}$
$\{ap(j), ap(b), re(j), ac(b)\}$	$\{ap(j), ap(b), re(j), ac(b)\}$	$\{j, b\}$
$\{ap(j), ap(b), ac(b)\}$ $\{ap(j), ap(b), ac(b), ac(j)\}$ $\{ap(j), ap(b), ac(b), re(j)\}$	$\{ap(j), ap(b), ac(j), ac(b), re(j)\}$	$\{j, b\}$
$\{ap(j), ap(b), ac(j)\}$ $\{ap(j), ap(b), ac(b), ac(j)\}$ $\{ap(j), ap(b), ac(b), re(b)\}$	$\{ap(j), ap(b), ac(j), ac(b), re(b)\}$	$\{j, b\}$
$\{ap(j), ap(b)\}$ $\{ap(j), ap(b), ac(j)\}$ $\{ap(j), ap(b), ac(j), ac(b)\}$ $\{ap(j), ap(b), ac(j), ac(b), re(j)\}$ $\{ap(j), ap(b), ac(j), ac(b), re(j), re(b)\}$	$\{ap(j), ap(b), ac(j), ac(b), re(j), re(b)\}$	$\{j, b\}$
$\{ap(j), ap(b), re(b)\}$ $\{ap(j), ap(b), re(b), re(j)\}$ $\{ap(j), ap(b), re(b), re(j), ac(j)\}$	$\{ap(j), ap(b), re(j), re(b), ac(j)\}$	$\{j, b\}$
$\{ap(j), ap(b), re(j)\}$ $\{ap(j), ap(b), re(j), re(b)\}$ $\{ap(j), ap(b), re(j), re(b), ac(b)\}$	$\{ap(j), ap(b), re(j), re(b), ac(b)\}$	$\{j, b\}$

Likewise $SE(Q)$ consists of elements $(X, Y)_C$ according to the following table:

X	Y	C
$\{ap(j), ap(b), ac(j), ac(b)\}$	$\{ap(j), ap(b), ac(j), ac(b)\}$	$\{j, b\}$
$\{ap(j), ap(b), re(j), re(b)\}$	$\{ap(j), ap(b), re(j), re(b)\}$	$\{j, b\}$
$\{ap(j), ap(b), ac(j), re(b)\}$	$\{ap(j), ap(b), ac(j), re(b)\}$	$\{j, b\}$
$\{ap(j), ap(b), re(j), ac(b)\}$	$\{ap(j), ap(b), re(j), ac(b)\}$	$\{j, b\}$
$\{ap(j), ap(b), ac(j), ac(b)\}$ $\{ap(j), ap(b), re(j), ac(b)\}$ $\{ap(j), ap(b), ac(j), ac(b), re(j)\}$	$\{ap(j), ap(b), ac(j), ac(b), re(j)\}$	$\{j, b\}$
$\{ap(j), ap(b), ac(j), ac(b)\}$ $\{ap(j), ap(b), re(b), ac(j)\}$ $\{ap(j), ap(b), ac(j), ac(b), re(b)\}$	$\{ap(j), ap(b), ac(j), ac(b), re(b)\}$	$\{j, b\}$
$\{ap(j), ap(b), ac(j), ac(b)\}$ $\{ap(j), ap(b), ac(j), re(b)\}$ $\{ap(j), ap(b), re(j), ac(b)\}$ $\{ap(j), ap(b), re(j), re(b)\}$ $\{ap(j), ap(b), ac(j), ac(b), re(j)\}$ $\{ap(j), ap(b), ac(j), ac(b), re(b)\}$ $\{ap(j), ap(b), ac(j), ac(b), ac(j)\}$ $\{ap(j), ap(b), ac(j), ac(b), ac(b)\}$ $\{ap(j), ap(b), ac(j), ac(b), re(j), re(b)\}$	$\{ap(j), ap(b), ac(j), ac(b), re(j), re(b)\}$	$\{j, b\}$
$\{ap(j), ap(b), re(j), re(b)\}$ $\{ap(j), ap(b), ac(j), re(b)\}$ $\{ap(j), ap(b), re(j), re(b), ac(j)\}$	$\{ap(j), ap(b), re(j), re(b), ac(j)\}$	$\{j, b\}$
$\{ap(j), ap(b), re(j), re(b)\}$ $\{ap(j), ap(b), re(j), ac(b)\}$ $\{ap(j), ap(b), re(j), re(b), ac(b)\}$	$\{ap(j), ap(b), re(j), re(b), ac(b)\}$	$\{j, b\}$

It can easily be seen that $SE(P) \neq SE(Q)$ and thus, according to Theorem 1, P and Q are not strongly equivalent. This fact can also be visualised with the help of the program

$$R = \left\{ \begin{array}{l} accepted(x) \leftarrow rejected(x), \\ rejected(x) \leftarrow accepted(x) \end{array} \right\}.$$

Appending R to our programs we get the following answer sets:

$$AS(P \cup R) = \emptyset$$

but

$$AS(Q \cup R) = \{\{ap(j), ap(b), ac(j), re(j), ac(b), re(b)\}\}.$$

Therefore, R is also a witness that P and Q are not strongly equivalent.

Another important definition in the context of program equivalence is the notion of a *correspondence frame*. It was introduced by Eiter et al. [8] and lifted to the non-ground setting by Oetsch et al. [9].

Definition 5 ([9]). *By a* correspondence frame*, or simply a* frame*, F, we understand a triple $(\mathcal{V}, \mathcal{C}, \rho)$, where $\mathcal{V}$ is a vocabulary, $\mathcal{C} \subseteq \mathcal{P}_V$, called the* context class *of F, and $\rho \subseteq 2^{2^{HB_\mathcal{V}}} \times 2^{2^{HB_\mathcal{V}}}$.*
*For every program P, Q over $\mathcal{V}$, we say that P and Q are F-*corresponding*, symbolically $P \simeq_F Q$, iff, for all $R \in \mathcal{C}$, $(AS(P \cup R), AS(Q \cup R)) \in \rho$.*

Following Eiter et al. [8], a *correspondence problem* Π over a vocabulary $\mathcal{V}$ is a tuple $(P, Q, \mathcal{C}, \rho)$, where P and Q are programs over $\mathcal{V}$ and $(\mathcal{V}, \mathcal{C}, \rho)$ is a frame. We say that $(P, Q, \mathcal{C}, \rho)$ holds iff $P \simeq_{(\mathcal{V},\mathcal{C},\rho)} Q$. Furthermore we call a correspondence problem of the form $(P, Q, \mathcal{P}^A_\mathcal{V}, \sqsubseteq^B_\mathcal{V})$ an *inclusion problem* and one of the form $(P, Q, \mathcal{P}^A_\mathcal{V}, \equiv^B_\mathcal{V})$ an *equivalence problem*.
Using correspondence problems we can characterise every equivalence notion mentioned above.

Theorem 2 ([8, 9]). *Let P and Q be programs over $\mathcal{V}$, then they are*

(*i*) *(ordinarily) equivalent iff $(P, Q, \{\emptyset\}, =)$ holds;*

(*ii*) *uniformly equivalent iff $(P, Q, \mathcal{F}_\mathcal{V}, =)$ holds;*

(*iii*) *strongly equivalent iff $(P, Q, \mathcal{P}_\mathcal{V}, =)$ holds.*

The more fine-grained notions of *relativised strong equivalence* and *relativised strong equivalence with projection* were also characterised as correspondence problems in the propositional case by Eiter et al.[8]. Later on, we will see that the generalised versions of those notions can also be expressed as correspondence problems.

CHAPTER 3

Generalised Relativised Strong Equivalence

When Woltran [7] introduced the concept of relativised strong equivalence, he dealt with propositional DLPs only. In this chapter, we will (similarily to what Eiter et al. [6] did with strong equivalence) generalise the notion to the non-ground setting. This is important because variables play an essential part in practise.

3.1 Basic Definitions and Properties

Following Woltran [7], we introduce the concept of relativised strong equivalence for non-ground logic programs.

Definition 6. *Let P and Q be logic programs over $\mathcal{V} = (\mathbb{P}, \mathbb{D})$ and let $A \subseteq \mathbb{P}$ be a set of predicates. Then, P and Q are strongly equivalent relative to A iff $AS(P \cup R) = AS(Q \cup R)$, for any program $R \in \mathcal{P}^A_{\mathcal{V}}$.*

Relativised strong equivalence as defined in Definition 6 can also be expressed as a correspondence problem.

Theorem 3. *Two programs P and Q over $\mathcal{V} = (\mathbb{P}, \mathbb{D})$ are strongly equivalent relative to $A \subseteq \mathbb{P}$ iff the equivalence problem $(P, Q, \mathcal{P}^A_{\mathcal{V}}, =)$ holds.*

Obviously if $A = \mathbb{P}$, then we obtain strong equivalence.

Before we can go to the model-theoretic characterisations, we have to establish some neccessary auxiliary definitions.
The following lemma is a slight adaption of the previous work of Eiter et al. [6].

Lemma 1. *Let P be a program over $\mathcal{V} = (\mathbb{P}, \mathbb{D})$, $C, C' \subseteq \mathbb{D}$ sets of constants such that $C \subseteq C'$, and $Y \subseteq HB_{P,C}$. Then, $Y \models grd(P, C)$ iff $Y \models grd(P, C')$.*

Proof. We start by showing that $Y \models grd(P, C)$ implies $Y \models grd(P, C')$.
Towards a contradiction, assume $Y \models grd(P, C)$ but $Y \not\models grd(P, C')$. $Y \not\models grd(P, C')$ implies that there is at least one rule $r' \in grd(P, C')$ with $Y \not\models r'$.
If $r' \in grd(P, C)$, then we obviously have a contradiction and are done.

So, consider $r' \notin grd(P, C)$. By the answer-set semantics, $Y \not\models r'$ can only hold if $B^+(r') \subseteq Y$ holds as well. Since $r' \notin grd(P, C)$ holds and $r' \in grd(P, C')$ holds by hypothesis, we get that $B^+(r')$ contains at least one constant $c \in (C' \setminus C)$. Therefore, there exists a predicate containing c with $p(c) \in B^+(r')$. Obviously $c \notin C$ holds and since $Y \subseteq HB_{P,C}$ holds as well, $p(c) \notin Y$ has to hold. Thus $B^+(r') \not\subseteq Y$ holds and $Y \not\models r'$ cannot hold.

The other direction $Y \models grd(P, C')$ implies $Y \models grd(P, C)$ follows trivially from $grd(P, C) \subseteq grd(P, C')$. □

The next lemma follows directly from the previous and will be used quite often in the rest of this work.

Lemma 2. *Let P and Q be two programs. Then, $Y \models grd(P \cup Q, HU_{P\cup Q})$ holds iff $Y \models grd(P, HU_P)$ and $Y \models grd(Q, HU_Q)$ both hold.*

Proof. We start by showing that $Y \models grd(P \cup Q, HU_{P\cup Q})$ implies $Y \models grd(P, HU_P)$ and $Y \models grd(Q, HU_Q)$.
Towards a contradiction, assume $Y \models grd(P \cup Q, HU_{P\cup Q})$, and either

(i) $Y \not\models grd(P, HU_P)$ or

(ii) $Y \not\models grd(Q, HU_Q)$ holds.

If (i) holds, then there is a rule $r \in grd(P, HU_P)$ with $Y \not\models r$. By the definition of grounding, $grd(P, HU_P) \subseteq grd(P \cup Q, HU_{P\cup Q})$ and thus $r \in grd(P \cup Q, HU_{P\cup Q})$. Since $Y \not\models r$, $Y \not\models grd(P \cup Q, HU_{P\cup Q})$ holds. This is obviously a contradiction.

The proof for (ii) works the same way.

Now we show the other direction.
Towards a contradiction assume $Y \not\models grd(P \cup Q, HU_{P\cup Q})$, $Y \models grd(P, HU_P)$ and $Y \models grd(Q, HU_Q)$. $Y \not\models grd(P \cup Q, HU_{P\cup Q})$ implies a rule $r \in grd(P \cup Q, HU_{P\cup Q})$ with $Y \not\models r$. Since $grd(P \cup Q, HU_{P\cup Q}) = (grd(P, HU_{P\cup Q}) \cup grd(Q, HU_{P\cup Q}))$ holds, we have either

(i) $r \in grd(P, HU_{P\cup Q})$; or

(ii) $r \in grd(Q, HU_{P\cup Q})$.

If (i), then $Y \not\models r$ implies $Y \not\models grd(P, HU_{P\cup Q})$ and we have a contradiction.

Similarily if (ii), then $Y \not\models r$ implies $Y \not\models grd(Q, HU_{P\cup Q})$ and we again have a contradiction. □

The main lemma of this section is lifted from the propositional case [7] and lays the groundwork for the following model-based characterisations of relativised strong equivalence.

Lemma 3. *For programs P and Q over a vocabulary $\mathcal{V}$ with predicates $\mathbb{P}$ and domain $\mathbb{D}$, and a set of predicates $A \subseteq \mathbb{P}$, the following propositions are equivalent:*

(1) $AS(P \cup R) \not\subseteq AS(Q \cup R)$ *for some program* $R \in \mathcal{P}^A_\mathcal{V}$*;*

(2) there exists a unary program $U \in \mathcal{P}^A_\mathcal{V}$ *such that*

$$AS(P \cup U) \not\subseteq AS(Q \cup U);$$

(3) there exists an interpretation $Y \subseteq (HB_{P,C_P} \cap HB_{Q,C_Q})$*, and two sets of constants* $HU_P \subseteq C_P \subseteq \mathbb{D}$ *and* $HU_Q \subseteq C_Q \subseteq \mathbb{D}$ *such that:*

- *(a)* $Y \models grd(P, C_P)$*,*
- *(b) for each* $Y' \subset Y$ *with* $Y' \equiv^A_\mathcal{V} Y$*,* $Y' \not\models grd(P, C_P)^Y$ *holds, and*
- *(c)* $Y \models grd(Q, C_Q)$ *implies the existence of an* $X \subset Y$ *such that* $X \models grd(Q, C_Q)^Y$ *and for each* $X' \subset Y$ *with* $X' \equiv^A_\mathcal{V} X$*,* $X' \not\models grd(P, C_P)^Y$ *holds.*

Proof. We first show that (1) implies (3).
Let $Y \subseteq (HB_{P,C_P} \cap HB_{Q,C_Q})$ be an interpretation and R a program over $(A, \mathbb{D})$ such that $Y \in AS(P \cup R)$ and $Y \notin AS(Q \cup R)$ hold.
By definition of answer sets, $Y \in AS(P \cup R)$ implies

(i) $Y \models grd(P \cup R, HU_{P \cup R})$, and

(ii) for each $Z \subset Y$, $Z \not\models grd(P \cup R, HU_{P \cup R})^Y$.

Condition (i) implies $Y \models grd(P, HU_{P \cup R})$ and, according to Lemma 2, it follows that $Y \models grd(P, HU_P)$ holds. $C_P \supseteq HU_P$ holds by definition and thus according to Lemma 1, (a) holds.
Condition (i) and Lemma 2 also imply $Y \models grd(R, HU_R)$ and therefore (by definition of a reduct) $Y \models grd(grd(R, HU_R)^Y$ holds. Since R contains predicates from A only, we get that $Y' \models grd(R, HU_R)$ holds for every Y' with $Y' \equiv^A_\mathcal{V} Y$. Combing this last observation with (ii), we obtain that for each $Y' \subset Y$ with $Y' \equiv^A_\mathcal{V} Y$, $Y' \not\models grd(P, HU_P)^Y$ holds. Hence, by Lemma 1, (b) is satisfied.
By definition, $Y \notin AS(Q \cup R)$ implies either

(iii) $Y \not\models grd(Q \cup R, HU_{Q \cup R})$; or

(iv) there exists an interpretation $X \subset Y$, such that $X \models grd(Q \cup R, HU_{Q \cup R})^Y$.

If (iii) holds, we know that $Y \not\models grd(Q, HU_Q)$ holds as well, since we already established $Y \models grd(R, HU_R)$ above and by Lemma 2, $Y \not\models grd(Q \cup R, HU_{Q\cup R})$ implies $Y \not\models grd(Q, HU_Q)$ or $Y \not\models grd(R, HU_R)$. So if (iii) holds, so does (c).

If (iv) holds, it is implied by Lemma 2 that $X \models grd(R, HU_R)^Y$ and $X \models grd(Q, HU_Q)^Y$ hold for each $X \subset Y$. Since $X \models grd(Q, HU_Q)$ implies $X \models grd(Q, C_Q)$ by Lemma 1, the first condition of the consequens in (c) is satisfied.
$X \models grd(R, HU_R)^Y$ implies $X' \models grd(R, HU_R)^Y$ for each $X' \equiv^A_\mathcal{V} X$, since R only contains predicates from A. Combining this observation with (ii) and Lemma 2, we get that $X' \not\models grd(P, HU_P)^Y$ holds for each $X' \subset Y$ with $X' \equiv^A_\mathcal{V} X$. By Lemma 1, $X' \not\models grd(P, HU_P)^Y$ implies $X' \not\models grd(P, C_P)^Y$ and thus the second condition of the consequens is satisfied. Hence (c) holds.

We now show that (3) implies (2).
Let $Y \subseteq (HB_{P,C_P} \cap HB_{Q,C_Q})$ be an interpretation for which conditions (a)-(c) hold, and let $C_P \subseteq HU_P$ and $C_Q \subseteq HU_Q$ be sets of constants. We have two cases: Either $Y \not\models grd(Q, HU_Q)$ or $Y \models grd(Q, HU_Q)$.

First, suppose $Y \not\models grd(Q, HU_Q)$ and let U be a unary program over A with $U = (Y \cap HB_{A,\mathbb{D}})$. We need to show that $Y \in AS(P \cup U)$ and $Y \notin AS(Q \cup U)$ hold.
$Y \in AS(P \cup U)$ holds iff:

(i) $Y \models grd(P \cup U, HU_{P\cup U})$; and

(ii) $Y' \not\models grd(P \cup U, HU_{P\cup U})^Y$, for every $Y' \subset Y$.

Since U is already ground and positive, $grd(P \cup U, HU_{P\cup U}) = (grd(P, HU_P) \cup U)$ and $grd(P \cup U, HU_{P\cup U})^Y = (grd(P, HU_P)^Y \cup U)$.
From (a) we know that $Y \models grd(P, C_P)$ holds and thus, by Lemma 1, we obtain $Y \models grd(P, HU_P)$. $Y \models U$ is implied by $Y \supseteq U$. Hence, by Lemma 2, (i) holds.
From (b) and Lemma 2 we get $Y' \not\models grd(P, C_P)^Y$ for each $Y' \subset Y$ with $Y' \equiv^A_\mathcal{V} Y$. So, in the case of $Y' \equiv^A_\mathcal{V} Y$, (ii) holds. If $Y' \not\equiv^A_\mathcal{V} Y$, then there is an $a \in HB_{A,\mathbb{D}}$ which is in Y but not in Y'. From $a \in Y$ and $U = (Y \cap HB_{A,\mathbb{D}})$ we get $a \in U$ and since U is unary, $a \in U^Y$ holds. Therefore, $Y' \not\models U^Y$ holds and hence (ii) holds.
$Y \not\models grd(Q, HU_Q)$ implies $Y \not\models grd(Q, HU_Q) \cup U$ and thus $Y \notin AS(Q \cup U)$ holds.

Now, suppose $Y \models grd(Q, HU_Q)$. By Lemma 1 this implies $Y \models grd(Q, C_Q)$ and thus according to (c), there exists an $X \subset Y$ for which $X \models grd(Q, C_Q)^Y$ holds. Furthermore, let U' be a unary program with $U' = (X \cap HB_{A,\mathbb{D}}) \cup \{p \leftarrow q \mid p, q \in (Y \setminus X) \cap HB_{A,\mathbb{D}}\}$.
We again need to show that $Y \notin AS(Q \cup U')$ and $Y \in AS(P \cup U')$ both hold.
$Y \notin AS(Q \cup U)'$ holds by definition iff:

(iii) $Y \models grd(Q \cup U', HU_{Q\cup U'})$; and

(iv) $X \models grd(Q \cup U', HU_{Q\cup U'})^Y$ hold.

Again, since U' is already grounded and positive, $grd(Q \cup U', HU_{Q\cup U'}) = (grd(Q, HU_Q) \cup U')$ and $grd(Q \cup U', HU_{Q\cup U'})^Y = (grd(Q, HU_Q)^Y \cup U')$.
By Lemma 2, (iii) holds iff $Y \models grd(Q, HU_Q)$ and $Y \models U'$ hold. From the above we

already know that $Y \models grd(Q, HU_Q)$ holds.
$Y \models U'$ holds iff:

(v) $Y \models X \cap HB_{A,\mathbb{D}}$; and

(vi) $Y \models \{p \leftarrow q \mid p, q \in (Y \setminus X) \cap HB_{A,\mathbb{D}}\}$ hold.

$X \subset Y$ implies $Y \supseteq (X \cap HB_{A,\mathbb{D}})$ and thus (v) holds. Since $((Y \setminus X) \cap HB_{A,\mathbb{D}}) \subseteq Y$, $p, q \in ((Y \setminus X) \cap HB_{A,\mathbb{D}})$ implies $p, q \in Y$. So for each rule $p \leftarrow q$ in U', both p and q are in Y. Hence (vi) holds and therefore $Y \models U'$ holds as well as (iii).
Now, (iv) holds iff $X \models grd(Q, HU_Q)^Y$ and $X \models U'$ hold. We know from above that $X \models grd(Q, C_Q)^Y$ holds, which according to Lemma 1 implies $X \models grd(Q, HU_Q)^Y$.
$X \models U'$ holds iff:

(vii) $X \models X \cap HB_{A,\mathbb{D}}$; and

(viii) $X \models \{p \leftarrow q \mid p, q \in (Y \setminus X) \cap HB_{A,\mathbb{D}}\}$ hold.

Obviously (vii) holds, since $X \supseteq (X \cap HB_{A,\mathbb{D}})$.
We know that $((Y \setminus X) \cap HB_{A,\mathbb{D}}) \cap X = \emptyset$, therefore, for each $p, q \in ((Y \setminus X) \cap HB_{A,\mathbb{D}})$, $p, q \notin X$ holds. Thus, for every rule $p \leftarrow q$, neither p nor q are in X, which means X is a model for every rule. Hence, (viii) holds and therefore $X \models U'$ holds as well as (iv).
Since (iii) and (iv) both hold, $Y \notin AS(Q \cup U')$ has to hold.

It remains to show that $Y \in AS(P \cup U')$ holds. Towards a contradiction, let us assume this is not the case. From the above, we already know that $Y \models grd(P, HU_P)$ as well as $Y \models U'$ holds. By Lemma 2, $Y \models grd(P, HU_P)$ and $Y \models U'$ imply $Y \models grd(P \cup U', HU_{P \cup U'})$. In order for $Y \notin AS((P \cup U')$ to hold, the following has to be true: There exists a $Z \subset Y$ such that $Z \models grd(P \cup U', HU_{P \cup U'})^Y$ holds.
$Z \models grd(P \cup U', HU_{P \cup U'})^Y$ is (similar as above) equivalent to $Z \models grd(P, HU_P)^Y \cup U'$.
$Z \models grd(P, HU_P)^Y \cup U'$ implies $Z \models U'$, which in turn implies $Z \supseteq (X \cap HB_{A,\mathbb{D}})$ by definition of U'. $(X \cap HB_{A,\mathbb{D}}) \subseteq Z$ implies $(X \cap HB_{A,\mathbb{D}}) \subseteq (Z \cap HB_{A,\mathbb{D}})$ and $Z \subset Y$ implies $(Z \cap HB_{A,\mathbb{D}}) \subseteq (Y \cap HB_{A,\mathbb{D}})$.
If $(Z \cap HB_{A,\mathbb{D}}) = (Y \cap HB_{A,\mathbb{D}})$, condition (b) is violated since $Z \subset Y$ and $Z \models grd(P, C_P)^Y$ are implied by $Z \models grd(P \cup U', HU_{P \cup U'})^Y$ and Lemma 1.
If $(X \cap HB_{A,\mathbb{D}}) = (Y \cap HB_{A,\mathbb{D}})$, condition (c) is violated since $Y \models grd(Q, C_Q)$, $X \subset Y$, $X \models grd(Q, C_Q)^Y$, $Z \subset Y$ and $Z \models grd(P, C_P)^Y$ all hold. This leaves the case with $(X \cap HB_{A,\mathbb{D}}) \subset (Z \cap HB_{A,\mathbb{D}}) \subset (Y \cap HB_{A,\mathbb{D}})$.
$(X \cap HB_{A,\mathbb{D}}) \subset (Z \cap HB_{A,\mathbb{D}})$ implies a q with $q \in (Z \cap HB_{A,\mathbb{D}})$ and $q \notin (X \cap HB_{A,\mathbb{D}})$, and since $(Z \cap HB_{A,\mathbb{D}}) \subset (Y \cap HB_{A,\mathbb{D}})$, $q \in (Y \cap HB_{A,\mathbb{D}})$ holds.
$q \in (Z \cap HB_{A,\mathbb{D}})$ implies $q \in Z$, $q \in (Y \cap HB_{A,\mathbb{D}})$ implies $q \in Y$ and $q \notin (X \cap HB_{A,\mathbb{D}})$ in addition with $q \in (Y \cap HB_{A,\mathbb{D}})$ implies $q \in ((Y \cap HB_{A,\mathbb{D}}) \setminus (Y \cap HB_{A,\mathbb{D}}))$.
$(Z \cap HB_{A,\mathbb{D}}) \subset (Y \cap HB_{A,\mathbb{D}})$ implies a p with $p \in (Y \cap HB_{A,\mathbb{D}})$ and $p \notin (Z \cap HB_{A,\mathbb{D}})$, and since $(X \cap HB_{A,\mathbb{D}}) \subset (Y \cap HB_{A,\mathbb{D}})$, $p \notin (X \cap HB_{A,\mathbb{D}})$. Also $p \notin (Z \cap HB_{A,\mathbb{D}})$ implies $p \notin Z$, $p \in (Y \cap HB_{A,\mathbb{D}})$ implies $p \in Y$ and $p \notin (X \cap HB_{A,\mathbb{D}})$ in addition with $p \in (Y \cap HB_{A,\mathbb{D}})$ implies $p \in ((Y \cap HB_{A,\mathbb{D}}) \setminus (Y \cap HB_{A,\mathbb{D}}))$.
To summarise, we have $p \notin Z$, $q \in Z$, and since $((Y \cap HB_{A,\mathbb{D}}) \setminus (Y \cap HB_{A,\mathbb{D}})) = ((Y \setminus X) \cap$

$HB_{A,\mathbb{D}}$), $p, q \in ((Y \setminus X) \cap HB_{A,\mathbb{D}})$. So there has to be a rule $p \leftarrow q$ in U and since $p \notin Z$ and $q \in Z$, Z is not a model of that rule and therefore $Z \not\models U'$ has to hold by definition. This is obviously a contradiction with the above and therefore $Y \in AS(P \cup U')$ holds.

(2) implies (1) holds trivially. □

3.2 Model-Theoretic Characterisations

Following Woltran's [7] model-theoretic approach for relativised strong equivalence of propositional DLPs, we now introduce a similar characterisation in the non-ground setting.

Definition 7. *Let $\mathcal{V}$ be a vocabulary with predicates $\mathbb{P}$ and domain $\mathbb{D}$, $C \subseteq \mathbb{D}$ a set of constants , P a logic program over $\mathcal{V}$, and $X, Y \subseteq HB_{P,C}$ interpretations. Furthermore, let $A \subseteq \mathbb{P}$ be a set of predicates.*
Then:

(1) $(X,Y)_C$ *is an* RSE-interpretation[1] *of P relative to A if either $X = Y$ or $X \subset Y|_{HB_{A,C}}$.*

(2) An RSE-interpretation $(X,Y)_C$ of P relative to A is an RSE-model *of P relative to A if*

- *(i)* $Y \models grd(P,C)$,
- *(ii) for all* $Y' \subset Y$ *with* $Y' \equiv^A_{\mathcal{V}} Y$, $Y' \not\models grd(P,C)^Y$, *and*
- *(iii)* $X \subset Y$ *implies the existence of an* $X' \subseteq Y$ *with* $X' \equiv^A_{\mathcal{V}} X$ *such that* $X' \models grd(P,C)^Y$.

The set of all RSE-Models of P relative to A is denoted by $RSE^A(P)$ or $SE^A(P)$.

From here on we might drop the explicit mentioning of the set A an RSE-model is relative to, if it is clear from context.

The next Lemma is an adaption of the one given by Eiter et al. [6] and follows directly from Lemma 1.

Lemma 4. *Let P be a program, $C, C' \subseteq \mathbb{D}$ sets of constants such that $C \subseteq C'$, and $X \subseteq Y \subseteq HB_{P,C}$. Then, $(X,Y)_C \in SE^A(P)$ iff $(X,Y)_{C'} \in SE^A(P)$.*

Proof. Assume $(X,Y)_C \in SE^A(P)$. From Definition 7 we know,

1. $Y \models grd(P,C)$,
2. for all $Y' \subset Y$ with $Y' \equiv^A_{\mathcal{V}} Y$, $Y' \not\models grd(P,C)^Y$, and

[1] Woltran called his structures *A-SE-interpretations* and *A-SE-models*, respectively.

3. $X \subset Y$ implies the existence of an $X' \subseteq Y$ with $X' \equiv_{\mathcal{V}}^{A} X$ such that $X' \models grd(P,C)^Y$.

By Lemma 1, $Y \models grd(P,C)$ iff $Y \models grd(P,C')$ since $C \subseteq C'$. Also, by Lemma 1 and $C \subseteq C'$, $Y' \not\models grd(P,C)^Y$ holds iff $Y' \not\models grd(P,C')^Y$ holds for any $Y' \subset Y$ with $Y' \equiv_{\mathcal{V}}^{A} Y$. Now, we have two cases, either $X = Y$ or $X \subset Y$.

If $X = Y$, then we are done since (i), (ii), and (iii) hold for $(X,Y)_{C'}$ and P.

If $X \subset Y$, then there exists an $X' \subseteq Y$ with $X' \equiv_{\mathcal{V}}^{A} X$ such that $X' \models grd(P,C)^Y$. By $C \subseteq C'$ and Lemma 1, we obtain $X' \models grd(P,C)^Y$ iff $X' \models grd(P,C')^Y$. Thus, (iii) of Definition 7 holds for $(X,Y)_{C'}$ and P. Hence, (i), (ii), and (iii) holds in this case as well and thus $(X,Y)'_C \in SE^A(P)$ holds.

The proof of the other direction is similar. □

Now that we have laid the neccessary groundwork, we can introduce the main theorem of this chapter.

Theorem 4. *Two logic programs P and Q are strongly equivalent relative to A iff their RSE-models relative to A are the same.*

Proof. Suppose P and Q are not strongly equivalent relative to A. Without loss of generality, according to Lemma 3, there has to be an $Y \subseteq (HB_{P,C_P} \cap HB_{Q,C_Q})$ and sets of constants $C_P \supseteq HU_P$ and $C_Q \supseteq HU_Q$ such that

(a) $Y \models grd(P,C_P)$,

(b) for each $Y' \subset Y$ with $Y' \equiv_{\mathcal{V}}^{A} Y$, $Y' \not\models grd(P,C_P)^Y$ holds, and

(c) $Y \models grd(Q,C_Q)$ implies the existence of an $X \subset Y$ such that $X \models grd(Q,C_Q)^Y$ and for each $X' \subset Y$ with $X' \equiv_{\mathcal{V}}^{A} X$, $X' \not\models grd(P,C_P)^Y$ holds.

What we now want to show is that there exists at least one RSE-model which is in $SE^A(P)$ but not in $SE^A(Q)$ or vice versa.
Set $C_{P\cup Q} = C_P \cup C_Q$. By Definition 7, $(Y,Y)_{C_{P\cup Q}}$ is an RSE-model of P relative to A iff:

(i) $Y \models grd(P,C_{P\cup Q})$; and

(ii) $Y' \not\models grd(P,C_{P\cup Q})^Y$ for each $Y' \subset Y$ with $Y' \equiv_{\mathcal{V}}^{A} Y$.

We know from (a) that $Y \models grd(P,C_P)$ holds and since $C_P \subseteq C_{P\cup Q}$, (i) holds by Lemma 1. From (b) we know that $Y' \not\models grd(P,C_P)^Y$ holds for each $Y' \subset Y$ with $Y' \equiv_{\mathcal{V}}^{A} Y$. Again by Lemma 1, we obtain $Y' \not\models grd(P,C_{P\cup Q})^Y$ and therefore (ii) holds. Hence $(Y,Y)_{C_{P\cup Q}}$ is an RSE-model of P.
Furthermore, condition (c) gives us three cases we need to explore. Either

(1) $Y \not\models grd(Q,C_Q)$;

(2) $Y \models grd(Q, C_Q)$ and $X \equiv_{\mathcal{V}}^{A} Y$; or

(3) $Y \models grd(Q, C_Q)$ and $(X \cap HB_{A,\mathbb{D}}) \subset (Y \cap HB_{A,\mathbb{D}})$.

If (1), then $(Y, Y)_{C_{P\cup Q}}$ cannot be an RSE-model of Q, since by Lemma 1 and $C_Q \subseteq C_{P\cup Q}$, $Y \not\models grd(Q, C_Q)$ implies $Y \not\models grd(Q, C_{P\cup Q})$.

If (2), then according to (c) $X \subset Y$ and $X \models grd(Q, C_Q)^Y$, which by Lemma 1 implies $X \models grd(Q, C_{P\cup Q})^Y$. Thus, condition (ii) of Definition 7 is violated (just set $X = Y'$). Hence,$(Y, Y)_{P\cup Q}$ is not an RSE-model of Q.

If (3), then $(X \cap HB_{A,\mathbb{D}}, Y)_{C_{P\cup Q}}$ satisfies conditions (i), (ii), and (iii) of Definition 7 and is therefore an RSE-Model of Q. But by condition (c) for every Z with $(Z \cap HB_{A,\mathbb{D}}) = (X \cap HB_{A,\mathbb{D}})$, $Z \not\models grd(P, C_P)^Y$ holds and by Lemma 1, $Z \not\models grd(P, C_{P\cup Q})^Y$ holds. This means that condition (iii) of Definition 7 cannot be fulfilled and thus $((X \cap HB_{A,\mathbb{D}}), Y)_{C_{P\cup Q}}$ is not an RSE-Model of P.

Now we proceed with the other direction of the theorem.
Suppose $(Z, Y)_C$ is an A-SE-Model of P but not of Q. By Definition 7, $C \subseteq \mathbb{D}$ and thus according to Lemma 4, $(Z, Y)_{\mathbb{D}} \in SE^A(P)$ and $(Z, Y)_{\mathbb{D}} \notin SE^A(Q)$. Since $HU_{P\cup Q} \subseteq \mathbb{D}$ obviously holds, we obtain $(Z, Y)_{HU_{P\cup Q}} \in SE^A(P)$ and $(Z, Y)_{HU_{P\cup Q}} \notin SE^A(Q)$ by applying Lemma 4. Now consider $C_{P\cup Q} \supseteq HU_{P\cup Q}$. Again by Lemma 4, $(Z, Y)_{C_{P\cup Q}} \in SE^A(P)$ and $(Z, Y)_{C_{P\cup Q}} \notin SE^A(Q)$ hold.

If $Z = Y$, then from Definition 7, it follows that $Y \models grd(P, C_{P\cup Q})$ and for each $Y' \subset Y$ with $Y' \equiv_{\mathcal{V}}^{A} Y$, $Y' \not\models grd(P, C_{P\cup Q})$. Thus, (a) and (b) from Lemma 3 hold. Since (Y, Y) is not an RSE-Model of Q we get by Definition 7 that either $Y \not\models grd(Q, C_{P\cup Q})$ or there is an $Y' \subset Y$ with $Y' \equiv_{\mathcal{V}}^{A} Y$ such that $Y' \models grd(Q, C_{P\cup Q})$. In the case of $Y \not\models grd(Q, C_{P\cup Q})$, condition (c) of Lemma 3 is obviously satisfied. Otherwise, if $Y \models grd(Q, C_{P\cup Q})$, condition (c) is satisfied by setting $X = Y'$. Hence, P and Q are not strongly equivalent relative to A.

If $Z \neq Y$, then whenever $(Z, Y)_{C_{P\cup Q}}$ is an RSE-model of P, $(Y, Y)_{C_{P\cup Q}}$ is an RSE-model as well. The case with $(Y, Y)_{C_{P\cup Q}}$ not being an RSE-model of Q was shown above, so we only have to prove the case with $(Y, Y)_{C_{P\cup Q}}$ being an RSE-model of Q. Supposing this, we get that $Y \models grd(Q, C_{P\cup Q})$ and for each $Y' \subset Y$ with $Y' \equiv_{\mathcal{V}}^{A} Y$, $Y' \not\models grd(Q, C_{P\cup Q})^Y$. Obviously, condition (a) and (b) are satisfied for Y and Q. Since $(Z, Y)_{C_{P\cup Q}}$ is not an RSE-model of Q we get the following from (ii) of Definition 7: for each $(X' \cap HB_{A,\mathbb{D}}) = Z$, $X' \not\models grd(Q, C_{P\cup Q})^Y$. Also, because $(Z, Y)_{C_{P\cup Q}}$ is an RSE-model of P, there is an $X'' \subset Y$ with $(X'' \cap HB_{A,\mathbb{D}}) = Z$ such that $X'' \models grd(Q, C_{P\cup Q})^Y$ (condition (iii) of Definition 7). Those two observations imply that conditions (c) of Lemma 3 is satisfied for Y and Q and therefore P and Q are not strongly equivalent relative to A. □

Example 4. *Let us consider a modified version of the program given in Example 1:*

$$M = \left\{ \begin{array}{l} accepted(x) \leftarrow applicant(x), not\ rejected(x), \\ rejected(x) \leftarrow applicant(x), not\ accepted(x), \\ applicant(x) \leftarrow person(x), not\ hired(x), \\ person(jane) \leftarrow, \\ person(bob) \leftarrow \end{array} \right\}.$$

Furthermore, consider the following subprograms of M:

$$P = \left\{ \begin{array}{l} accepted(x) \leftarrow applicant(x), not\ rejected(x), \\ rejected(x) \leftarrow applicant(x), not\ accepted(x), \end{array} \right\}$$

and

$$R = \left\{ \begin{array}{l} applicant(x) \leftarrow person(x), not\ hired(x), \\ person(jane) \leftarrow, \\ person(bob) \leftarrow \end{array} \right\}.$$

Obviously, $M = P \cup R$.
Now, assume we want to replace the subprogram P with

$$Q = \{accepted(x) \vee rejected(x) \leftarrow applicant(x)\}.$$

Let us take a look at their RSE-models relative to $A = \{applicant(\cdot), person(\cdot), hired(\cdot)\}$.

We again use the following shorthands: $ap(\cdot)$ for $applicant(\cdot)$, $ac(\cdot)$ for $accepted(\cdot)$, $re(\cdot)$ for $rejected(\cdot)$, j for jane, and b for bob. Furthermore, we have the vocabulary $\mathcal{V} = (\mathbb{D}, \mathbb{P})$, where $\mathbb{D} = \{jane, bobo\}$ and $\mathbb{P} = \{applicant(\cdot), accepted(\cdot), rejected(\cdot), person(\cdot), hired(\cdot)\}$.

Then $SE^A(P)$ as well as $SE^A(Q)$ consists of elements $(X, Y)_C$ according to the following table:

X	Y	C
$\emptyset$	$\emptyset$	$\{j, b\}$
$\emptyset$ $\{ap(b), ac(b)\}$	$\{ap(b), ac(b)\}$	$\{j, b\}$
$\emptyset$ $\{ap(b), re(b)\}$	$\{ap(b), re(b)\}$	$\{j, b\}$
$\emptyset$ $\{ap(j), ac(j)\}$	$\{ap(j), ac(j)\}$	$\{j, b\}$
$\emptyset$ $\{ap(j), re(j)\}$	$\{ap(j), re(j)\}$	$\{j, b\}$
$\emptyset$ $\{ap(j)\}$ $\{ap(b)\}$ $\{ap(j), ap(b), ac(j), ac(b)\}$	$\{ap(j), ap(b), ac(j), ac(b)\}$	$\{j, b\}$
$\emptyset$ $\{ap(j)\}$ $\{ap(b)\}$ $\{ap(j), ap(b), re(j), re(b)\}$	$\{ap(j), ap(b), re(j), re(b)\}$	$\{j, b\}$
$\emptyset$ $\{ap(j)\}$ $\{ap(b)\}$ $\{ap(j), ap(b), ac(j), re(b)\}$	$\{ap(j), ap(b), ac(j), re(b)\}$	$\{j, b\}$
$\emptyset$ $\{ap(j)\}$ $\{ap(b)\}$ $\{ap(j), ap(b), ac(b), re(j)\}$	$\{ap(j), ap(b), ac(b), re(j)\}$	$\{j, b\}$
$\emptyset$	$\emptyset$	$\{j\}$
$\emptyset$ $\{ap(j), ac(j)\}$	$\{ap(j), ac(j)\}$	$\{j\}$
$\emptyset$ $\{ap(j), re(j)\}$	$\{ap(j), re(j)\}$	$\{j\}$
$\emptyset$	$\emptyset$	$\{b\}$
$\emptyset$ $\{ap(b), ac(b)\}$	$\{ap(b), ac(b)\}$	$\{b\}$
$\emptyset$ $\{ap(b), re(b)\}$	$\{ap(b), re(b)\}$	$\{b\}$

The RSE-models of P and Q are the same and thus we should be able to replace P with Q in M.

Replacing P with Q in M yields

$$M' = \left\{ \begin{array}{l} accepted(x) \vee rejected(x) \leftarrow applicant(x), \\ applicant(x) \leftarrow person(x), not\ hired(x), \\ person(jane) \leftarrow, \\ person(bob) \leftarrow \end{array} \right\}.$$

Now, according to Theorem 4 and Definition 7, $AS(M) = AS(M')$ has to hold and it can easily be verified that it does.

CHAPTER 4

Generalised Correspondence Checking

In this chapter we generalize some characteristic structures which were introduced by Eiter et al. [8]. Those structures are then linked to correspondence checking in the non-ground case.

4.1 Basic Definitions and Properties

We start with some basic definitions and properties. The first one adds *projection* to generalised relativised strong equivalence.

Definition 8. *Let P and Q be logic programs over $\mathcal{V} = (\mathbb{P}, \mathbb{D})$ and let $A, B \subseteq \mathbb{P}$ be a set of predicates. Then, P and Q are strongly equivalent relative to A under projection B iff $AS(P \cup R) \equiv_{\mathcal{V}}^{B} AS(Q \cup R)$, for any program $R \in \mathcal{P}_{\mathcal{V}}^{A}$.*

Just like without projection, relativised strong equivalence with projection as defined in Definition 8 can also be expressed as a correspondence problem.

Theorem 5. *Two programs P and Q over $\mathcal{V} = (\mathbb{P}, \mathbb{D})$ are strongly equivalent relative to $A \subseteq \mathbb{P}$ under projection $B \subseteq \mathbb{P}$ iff the equivalence problem $(P, Q, \mathcal{P}_{\mathcal{V}}^{A}, \equiv_{\mathcal{V}}^{B})$ holds.*

Obviously if $B = \mathbb{P}$, then we obtain relative strong equivalence without projection.

Eiter et al. [8] defined a useful property for sets of SE-models. The following definition is a generalisation of that property.

Definition 9. *A set S of (R)SE-interpretations is* complete*, if for each $(X, Y)_C \in S$, also $(Y, Y)_C \in S$ as well as $(X, Z)_C \in S$, for any $Z \supseteq Y$ with $(Z, Z)_C \in S$.*

It can be shown that the set $SE(P)$ of a program P is always complete. [8] Furthermore, we introduce the following definition.

Definition 10. *A set S of (R)SE-interpretations is called* over C *if for each $(X,Y)_{C'} \in S$, $C' = C$.*

That definition enables us to restrict a set of SE-models to a particular set of grounding constants. Think of a complete set of SE-models S over C. Then, the grounding of a program P with respect to C is semantically given by S.
In order to show the relationship between our structures and correspondence problems we need some auxiliary results. Those results are all lifted from the previously established propositional case [8].

Definition 11. *For a set S of (R)SE-interpretations, a vocabulary $\mathcal{V}$, and an interpretation Y, we define $\sigma^A_{Y,\mathcal{V}}(S) = \{(X,Z)_C \in S \mid Z \equiv^A_{\mathcal{V}} Y\}$.*

Proposition 1. *Let P be a program over $\mathcal{V} = (\mathbb{P}, \mathbb{D})$, $Y \subseteq HB_{P,C}$ an interpretation, $C \subseteq \mathbb{D}$, and $A, B \subseteq \mathbb{P}$. Then, $Y|_{HB_{B,\mathbb{D}}} \notin AS(P)|_{HB_{B,\mathbb{D}}}$ iff, for each $(Z,Z)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(P))$, there exists a non-total $(X,Z)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(P))$.*

Proof. We start by proving the only-if-direction. Assume a $(Z,Z)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(P))$ for which there is no non-total $(X,Z)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(P))$. We show $Y|_{HB_{B,\mathbb{D}}} \in AS(P)|_{HB_{B,\mathbb{D}}}$.
From Definition 11, we get that $(Z,Z)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(P))$ implies $(Z,Z)_C \in SE^A(P)$ and $Z \equiv^B_{\mathcal{V}} Y$. Furthermore, by Definition 7, we know that $(Z,Z)_C \in SE^A(P)$ implies $Z \models grd(P,C)$ and for all $Z' \subset Z$ with $Z' \equiv^A_{\mathcal{V}} Z$, $Z' \not\models grd(P,C)$.
We now show that $Z \in AS(P)$ holds. Towards a contradiction, assume $Z \notin AS(P)$. Then, either

(i) $Z \not\models grd(P, HU_P)$; or

(ii) there is an $X \subset Z$ with $X \models grd(P, HU_P)^Z$.

If (i), then we have a contradiction because according to Lemma 1, $Z \models grd(P,C)$ and $C \subseteq \mathbb{D}$ imply $Z \models grd(P,\mathbb{D})$, which in turn implies $Z \models grd(P, HU_P)$.

So, assume (ii) holds. Then, by Definition 7, $(X,Z)_C \in SE^A(P)$ has to hold because (i), (ii), and (iii) of the definition hold (for (iii) just set $X' = X$). But since we have assumed above that there is no such $(X,Z)_C \in SE^A(P)$, we have a contradiction. Hence $Z \in AS(P)$.
Now, $Z \in AS(P)$ implies $Z|_{HB_{B,\mathbb{D}}} \notin AS(P)|_{HB_{B,\mathbb{D}}}$ by definition and since $Z \equiv^B_{\mathcal{V}} Y$, we obtain $Y|_{HB_{B,\mathbb{D}}} \notin AS(P)|_{HB_{B,\mathbb{D}}}$.

We proceed with the if-direction. What we need to show is that $Y|_{HB_{B,\mathbb{D}}} \in AS(P)|_{HB_{B,\mathbb{D}}}$ implies the existence of a total $(Z,Z)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(P))$ for which there is no non-total $(X,Z)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(P))$. Towards a contradiction, assume that $Y|_{HB_{B,\mathbb{D}}} \in AS(P)|_{HB_{B,\mathbb{D}}}$ and there is no total $(Z,Z)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(P))$ for which there is no $(X,Z)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(P))$.
By definition, $Y|_{HB_{B,\mathbb{D}}} \in AS(P)|_{HB_{B,\mathbb{D}}}$ implies that there is a $Z \in AS(P)$ such that $Z \equiv^B_{\mathcal{V}} Y$. From the definition of answer sets, we get that $Z \in AS(P)$ implies $Z \models grd(P, HU_P)$,

$X \not\models grd(P, HU_P)^Z$ for each $X \subset Z$, and $Z \subseteq HB_P$.
By Lemma 1, we obtain

(i) $Z \models grd(P, C)$, and

(ii) $X \not\models grd(P, C)^Z$ for each $X \subset Z$.

$(Z, Z)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(P))$ obviously follows from (i), (ii), and $Z \equiv^B_{\mathcal{V}} Y$.
According to our assumption, $(Z, Z)_C$ implies a $X \subset Z$ such that $(X, Z)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(P))$. But, by Definition 7, $(X, Z)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(P))$ implies the existence of an $X' \subset Z$ with $X' \equiv^A_{\mathcal{V}} X$ such that $X' \models grd(P, C)^Z$ holds. This is a contradiction to (ii), and hence we have proven the if-direction. □

Proposition 2. *For any program P over the predicates A and constants $\mathbb{D}$,*

(i) $(X, Y)_C \in SE(P)|_{HB_{A,\mathbb{D}}}$ implies $(X', Y')_C \in SE(P)$, for any $X' \subseteq Y'$ with $X'|_{HB_{A,\mathbb{D}}} = X$ and $Y'|_{HB_{A,\mathbb{D}}} = Y$; and

(ii) $(X, Y)_C \in SE(P)$ implies $(X|_{HB_{A,\mathbb{D}}}, Y|_{HB_{A,\mathbb{D}}})_C \in SE(P)|_{HB_{A,\mathbb{D}}}$.

Proof. Towards a contradiction, assume that (i) does not hold. Then, there is some $(X, Y)_C \in SE(P)|_{HB_{A,\mathbb{D}}}$ such that $(X', Y')_C \notin SE(P)$ with $X' \subseteq Y'$, $X'|_{HB_{A,\mathbb{D}}} = X$, and $Y'|_{HB_{A,\mathbb{D}}} = Y$.
By definition, $(X', Y')_C \notin SE(P)$ implies either

(a) $Y' \not\models grd(P, C)$; or

(b) $X' \not\models grd(P, C)^{Y'}$.

If (a), then there has to be a rule $r \in grd(P, C)$ such that $Y' \not\models r$. Furthermore, from $(X, Y)_C \in SE(P)|_{HB_{A,\mathbb{D}}}$, it follows that there is some $Y'' \models grd(P, C)$ with $Y''|_{HB_{A,\mathbb{D}}} = Y$. $Y''|_{HB_{A,\mathbb{D}}} = Y$ obviously implies $Y'' \equiv^A_{\mathcal{V}} Y'$, since $Y'|_{HB_{A,\mathbb{D}}} = Y$.
Now, $Y \not\models r$ implies $B^+(r) \subseteq Y'$, $B^-(r) \cap Y' = \emptyset$, and $H(r) \cap Y' = \emptyset$. Since r only contains predicates from A, we obtain $B^+(r) \subseteq Y'|_{HB_{A,\mathbb{D}}}$, $B^-(r) \cap Y'|_{HB_{A,\mathbb{D}}} = \emptyset$, and $H(r) \cap Y'|_{HB_{A,\mathbb{D}}} = \emptyset$. From $Y'' \equiv^A_{\mathcal{V}} Y'$, we get that $Y'|_{HB_{A,\mathbb{D}}} \subseteq Y''$ and thus $Y'' \not\models r$ has to hold. Hence there can be no Y'' such that $Y'' \models grd(P, C)$ and $Y''|_{HB_{A,\mathbb{D}}} = Y$ hold and therefore $(X, Y)_C \notin SE(P)|_{HB_{A,\mathbb{D}}}$ holds. The latter is obviously a contradiction.

If (b), then there has to be a rule $r \in grd(P, C)^{Y'}$ such that $X' \not\models r$. We know from $(X, Y)_C \in SE(P)|_{HB_{A,\mathbb{D}}}$ that there has to be an $X'' \models grd(P, C)^{Y'}$ with $X''|_{HB_{A,\mathbb{D}}} = X$. $X''|_{HB_{A,\mathbb{D}}} = X$ obviously implies $X'' \equiv^A_{\mathcal{V}} X'$, since $X'|_{HB_{A,\mathbb{D}}} = X$.
$X' \not\models r$ implies $B^+(r) \subseteq X'$ and $H(r) \cap X' = \emptyset$. Since r only contains predicates from A, this is equivalent to $B^+(r) \subseteq X'|_{HB_{A,\mathbb{D}}}$ and $H(r) \cap X'|_{HB_{A,\mathbb{D}}} = \emptyset$. Now, from $X'' \equiv^A_{\mathcal{V}} X'$, we get $X'|_{HB_{A,\mathbb{D}}} \subseteq X''$ and therefore $X'' \not\models r$. But this would mean that there can be no $X'' \models grd(P, C)^{Y'}$ with $X'' \equiv^A_{\mathcal{V}} X$ and thus $(X, Y)_C \notin SE(P)|_{HB_{A,\mathbb{D}}}$, which would be a contradiction.

Condition (ii) of the proposition follows trivially from the definition of $S|_A$. □

Lemma 5. *Let P be a program over $\mathcal{V} = (\mathbb{P}, \mathbb{D})$ and let Q be a program over $(A, \mathbb{D})$ with $A \subseteq \mathbb{P}$. Then, $SE^A(P) \cap SE(Q) = SE^A(P \cup Q)$.*

Proof. First we are going to show that $(X,Y)_C \notin SE^A(P \cup Q)$ implies $(X,Y)_C \notin SE^A(P)$ or $(X,Y)_C \notin SE(Q)$.
So, suppose $(X,Y)_C \notin SE^A(P \cup Q)$. Then, by Definition 7 either

(i) $Y \not\models grd(P \cup Q, C)$; or

(ii) there exists a $Y' \subset Y$ with $Y' \equiv^A_{\mathcal{V}} Y$ such that $Y' \models grd(P \cup Q, C)^Y$; or

(iii) $X \subset Y$ and for all $X' \subseteq Y$ with $X' \equiv^A_{\mathcal{V}} X$, $X' \not\models grd(P \cup Q, C)^Y$ holds.

If (i) holds then either $Y \not\models grd(P, C)$ or $Y \not\models grd(Q, C)$. So either by Definition 7, $(X,Y)_C \notin SE^A(P)$ holds, or $(X,Y)_C \notin SE(Q)$ holds by definition of an SE-model.

If (ii) holds then $Y' \models grd(P, C)^Y$ holds as well, and since we have $Y' \subset Y$ and $Y' \equiv^A_{\mathcal{V}} Y$, $(X,Y)_C \notin SE^A(P)$ has to hold by (ii) of Definition 7.

If (iii) applies then either $X' \not\models grd(P, \mathbb{D})^Y$ or $X' \not\models grd(Q, \mathbb{D})^Y$ holds for every $X' \subseteq Y$ with $X' \equiv^A_{\mathcal{V}} X$.
In the first case, $(X,Y)_C \notin SE^A(P)$ has to hold since (iii) of Definition 7 cannot be satisfied for any $(X,Y)_C$.
In the second case, we know from $X' \not\models grd(Q, C)^Y$ that $(X',Y)_C \notin SE(Q)$ holds. From Proposition 2, we know that $(X',Y')_C \notin SE(Q)$ implies $(X,Y)_C \notin SE(Q)|_{HB_{A,\mathbb{D}}}$, for any $X' \subseteq Y'$ with $X'|_{HB_{A,\mathbb{D}}} = X$ and $Y'|_{HB_{A,\mathbb{D}}} = Y$; and $(X|_{HB_{A,\mathbb{D}}}, Y|_{HB_{A,\mathbb{D}}})_C \notin SE(Q)|_{HB_{A,\mathbb{D}}}$ implies $(X,Y)_C \notin SE(Q)$. Now, from (iii), we know that $X' \subseteq Y$ and $X'|_{HB_{A,\mathbb{D}}} = X|_{HB_{A,\mathbb{D}}}$. Therefore, $(X',Y)_C \notin SE(Q)$ implies $(X|_{HB_{A,\mathbb{D}}}, Y|_{HB_{A,\mathbb{D}}})_C \notin SE(Q)|_{HB_{A,\mathbb{D}}}$ since obviously $Y|_{HB_{A,\mathbb{D}}} = Y|_{HB_{A,\mathbb{D}}}$. By Proposition 2, $(X|_{HB_{A,\mathbb{D}}}, Y|_{HB_{A,\mathbb{D}}})_C \notin SE(Q)|_{HB_{A,\mathbb{D}}}$ implies $(X,Y)_C \notin SE(Q)$.

Now, what remains to be proven is that $(X,Y)_C \in SE^A(P \cup Q)$ implies $(X,Y)_C \in SE^A(P)$ and $(X,Y)_C \in SE(Q)$.
Suppose $(X,Y)_C \in SE^A(P \cup Q)$. Then, by Definition 7 the following propositions hold:

(i) $Y \models grd(P \cup Q, C)$,

(ii) for all $Y' \subset Y$ with $Y' \equiv^A_{\mathcal{V}} Y$, $Y' \not\models grd(P \cup Q, C)^Y$ holds, and

(iii) $X \subset Y$ implies the existence of an $X' \subseteq Y$ with $X' \equiv^A_{\mathcal{V}} X$ such that $X' \models grd(P \cup Q, C)^Y$.

Condition (i) implies $Y \models grd(Q, C)$, which in turn implies $Y \models grd(Q, C)^Y$. Therefore, $(Y,Y)_C \in SE(Q)$ holds.
If $X \subset Y$, (iii) implies $X' \models grd(Q, C)^Y$ and therefore $(X',Y)_C \in SE(Q)$. By Proposition 2, $(X',Y)_C \in SE(Q)$ implies $(X'|_{HB_{A,\mathbb{D}}}, Y|_{HB_{A,\mathbb{D}}})_C \in SE(Q)|_{HB_{A,\mathbb{D}}}$ and since $X|_{HB_{A,\mathbb{D}}} = X'|_{HB_{A,\mathbb{D}}}$ holds we get $(X|_{HB_{A,\mathbb{D}}}, Y|_{HB_{A,\mathbb{D}}})_C \in SE(Q)|_{HB_{A,\mathbb{D}}}$. Now, also by Proposition 2, $(X|_{HB_{A,\mathbb{D}}}, Y|_{HB_{A,\mathbb{D}}})_C \in SE(Q)|_{HB_{A,\mathbb{D}}}$ implies $(X,Y)_C \in SE(Q)$.

From Proposition 2 we know that $(Y,Y)_C \in SE(Q)$ (which we have established above) implies $(Y|_{HB_{A,\mathbb{D}}}, Y|_{HB_{A,\mathbb{D}}})_C \in SE(Q)|_{HB_{A,\mathbb{D}}}$ and that $(Y|_{HB_{A,\mathbb{D}}}, Y|_{HB_{A,\mathbb{D}}})_C \in SE(Q)|_{HB_{A,\mathbb{D}}}$ implies $(Y',Y')_C \in SE(Q)$ for any Y' with $Y|_{HB_{A,\mathbb{D}}} = Y'|_{HB_{A,\mathbb{D}}}$. Therefore, $Y' \models grd(Q,C)^Y$ holds for every $Y' \equiv^A_{\mathcal{V}} Y$. Together with (ii), the above means that for every $Y' \subset Y$ with $Y' \equiv^A_{\mathcal{V}} Y$, $Y' \not\models grd(P,C)^Y$.
(iii) implies that if $X \subset Y$, there exists an $X' \subseteq Y$ with $X' \equiv^A_{\mathcal{V}} X$ such that $X' \models grd(P,C)^Y$.
Combining those last two observations with (i), we easily obtain $(X,Y)_C \in SE^A(P)$. □

The next proposition is not really a lifting to the non-ground setting. That is, because reconstructing a non-ground program from a set of SE-models is not trivial. Our generalised approach takes a complete set of SE-models over a certain set of constants and computes a ground program which is semantically equal to the original non-ground program for the given constants. Nonetheless, the construction of the program is very similar to the one given by Eiter et al. [8].

Proposition 3. *Let $\mathcal{V} = (\mathbb{P},\mathbb{D})$ be a vocabulary, $A \subseteq \mathbb{P}$ a set of predicates, $C \subseteq \mathbb{D}$ a set of constants, and S a complete set of SE-models over C. Then, there exists a program $P_{S,A,C} \in \mathcal{P}^A_{\mathcal{V}}$ such that $SE(P_{S,A,C}) \equiv^A_{\mathcal{V}} S$.*

Proof. Consider the program $P_{S,A,C} = M \cup N$, where

$$M = \{\leftarrow Y, not(HB_{A,\mathbb{D}} \setminus Y) \mid Y \subseteq HB_{A,\mathbb{D}}, (Y,Y)_C \notin S|_{HB_{A,\mathbb{D}}}\}$$

and

$$N = \{ \bigvee_{x \in (Y \setminus X)} x \leftarrow X, not(HB_{A,\mathbb{D}} \setminus Y) \mid X \subset Y, (Y,Y)_C \in S|_{HB_{A,\mathbb{D}}}, (X,Y)_C \notin S|_{HB_{A,\mathbb{D}}}\}.$$

We want to show that $S \equiv^A_{\mathcal{V}} SE(P_{S,A,C})$.
We start by showing that $S \sqsubseteq^A_{\mathcal{V}} SE(P_{S,A,C})$ holds. Consider a $(X,Y)_C \notin SE(P_{S,A,C})$. We have two cases: Either

(1) $(Y,Y)_C \notin SE(P_{S,A,C})|_{HB_{A,\mathbb{D}}}$, or

(2) $(Y,Y)_C \in SE(P_{S,A,C})|_{HB_{A,\mathbb{D}}}$.

Assume (1) holds. If $Y \not\subseteq HB_{A,\mathbb{D}}$, then $(X,Y)_C \notin S|_{HB_{A,\mathbb{D}}}$ is trivially true. So $Y \subseteq HB_{A,\mathbb{D}}$, which imples $(Y,Y)_C \notin SE(P_{S,A,C})$. The latter implies $Y \not\models P_{S,A,C}$ or $Y \not\models P^Y_{S,A,C}$.
If $Y \not\models P_{S,A,C}$ or $Y \not\models P^Y_{S,A,C}$, then there has to be a rule $r \in P_{S,A,C}$ such that $Y \subseteq B^+(r)$, $Y \cap B^-(r) = \emptyset$, and $Y \cap H(r) = \emptyset$.
If $r \in M$, then $B^+(r) \subseteq HB_{A,\mathbb{D}}$ and $Y \cap (HB_{A,\mathbb{D}} \setminus B^+(r)) = \emptyset$. Together with $Y \subseteq HB_{A,\mathbb{D}}$ this implies $Y = B^+(r)$ and thus $(Y,Y)_C \notin S|_{HB_{A,\mathbb{D}}}$. Since S is complete we obtain $(X,Y)_C \notin S|_{HB_{A,\mathbb{D}}}$.
Now, assume $r \in N$. Let us write $H(r) = (Y' \setminus X')$, $B^+(r) = X'$, and $B^-(r) =$

$(HB_{A,\mathbb{D}} \setminus Y')$ for some sets X' and Y'. The following statements obviously hold: $Y \supseteq X'$, $Y \cap (HB_{A,\mathbb{D}} \setminus Y') = \emptyset$, and $Y \cap (Y' \setminus X')$. $Y \cap (HB_{A,\mathbb{D}} \setminus Y') = \emptyset$ implies $Y \subseteq Y'$ and $Y \supseteq X'$ in addition with $Y \cap (Y' \setminus X')$ implies $Y = X'$. Therefore, there is a $Z \supset Y$ such that $(Z,Z)_C \in S|_{HB_{A,\mathbb{D}}}$ and $(Y,Z)_C \in S|_{HB_{A,\mathbb{D}}}$. By completeness of S, we get $(Y,Y)_C \in S|_{HB_{A,\mathbb{D}}}$ and $(X,Y)_C \in S|_{HB_{A,\mathbb{D}}}$.

If (2) holds, then $X \not\models P^Y_{S,A,C}$ has to hold as well because $Y \models P_{S,A,C}$ follows from $(Y,Y)_C \in SE(P_{S,A,C})|_{HB_{A,\mathbb{D}}}$ and otherwise $(X,Y)_C \notin SE(P_{S,A,C})|_{HB_{A,\mathbb{D}}}$ would not hold. $X \not\models P^Y_{S,A,C}$ implies a rule $r^Y \in P^Y_{S,A,C}$ such that $X \supseteq B^+(r^Y)$, $Y \cap B^-(r^Y) = \emptyset$, and $X \cap H(r^Y) = \emptyset$. $X \subseteq Y$ and $X \supseteq B^+(r^Y)$ imply $Y \subseteq B^+(r^Y)$ and since $B^-(r^Y) = (HB_{A,\mathbb{D}} \setminus B^+(r^Y)$ as well as $B^+(r^Y) \subseteq HB_{A,\mathbb{D}}$ hold, we obtain $Y = B^+(r^Y)$. $Y = B^+(r^Y)$ obviously implies $(Y,Y)_C \notin S|_{HB_{A,\mathbb{D}}}$ by the construction of $P_{S,A,C}$, but this would also imply $Y \not\models P_{S,A,C}$ and thus $(Y,Y)_C \notin SE(P_{S,A,C})|_{HB_{A,\mathbb{D}}}$. Since we have assumed $(Y,Y)_C \in SE(P_{S,A,C})|_{HB_{A,\mathbb{D}}}$, $r^Y \in N$ has to hold.
Condition $r^Y \in N$ implies $(X \cap (Y' \setminus X')) = \emptyset$, $Y \cap (HB_{A,\mathbb{D}} \setminus Y') = \emptyset$, and $X \supseteq X'$, where X' and Y' are the X and Y occuring in N. $X \subseteq Y$ implies $X \cap (HB_{A,\mathbb{D}} \setminus Y') = \emptyset$ and since $X \subseteq HB_{A,\mathbb{D}}$ it follows that $X \subseteq Y'$. $X \cap (Y' \setminus X') = \emptyset$ in addition with $X \subseteq Y'$ implies $X \subseteq X'$ and thus $X = X'$. Now, by construction of N, there has to be a $Z \supset X$ such that $(Z,Z)_C \in S|_{HB_{A,\mathbb{D}}}$ and $(X,Z)_C \notin S|_{HB_{A,\mathbb{D}}}$. Hence, we obtain $(X,Y)_C \notin S|_{HB_{A,\mathbb{D}}}$ by completeness of S.

Now we show $SE(P_{S,A,C}) \sqsubseteq^A_{\mathcal{V}} S$.
Consider a $(X,Y)_C$ with $(X,Y)_C \notin S|_{HB_{A,\mathbb{D}}}$. Again, we have two cases: Either

(1) $(Y,Y)_C \notin S|_{HB_{A,\mathbb{D}}}$, or

(2) $(Y,Y)_C \in S|_{HB_{A,\mathbb{D}}}$.

If (1) holds and $Y \not\subseteq HB_{A,\mathbb{D}}$, $(X,Y)_C \notin SE(P_{S,A,C})|_{HB_{A,\mathbb{D}}}$ is trivially true. So, let us assume $Y \subseteq HB_{A,\mathbb{D}}$. By construction there has to be a rule

$$r = (\leftarrow Y, not(HB_{A,\mathbb{D}} \setminus Y))$$

in $P_{S,A,C}$. Obviously, $Y \models B(r)$ holds and thus $Y \not\models P_{S,A,C}$. Hence $(Y,Y)_C \notin SE(P_{S,A,C})$ and also $(Y,Y)_C \notin SE(P_{S,A,C})|_{HB_{A,\mathbb{D}}}$. By completeness of S, we again obtain $(X,Y)_C \notin SE(P_{S,A,C})|_{HB_{A,\mathbb{D}}}$.

If (2), then by construction the rule

$$r = (\bigvee_{x \in (Y \setminus X)} x \leftarrow X, not(HB_{A,\mathbb{D}} \setminus Y))$$

has to be in $P_{S,A,C}$. The reduct of the rule is given by

$$r^Y = (\bigvee_{x \in (Y \setminus X)} x \leftarrow X),$$

so $X \supseteq B(r^Y)$ obviously holds. But since $(Y \setminus X) \cap X = \emptyset$ and $X \cap H(r^Y) = \emptyset$, it follows that $X \not\models r^Y$. This in turn implies $X \not\models P^Y_{S,A,C}$ and therefore $(X,Y)_C \notin SE(P_{S,A,C})$. $(X,Y)_C \notin SE(P_{S,A,C})$ of course implies $(X,Y)_C \notin SE(P_{S,A,C})|_{HB_{A,\mathbb{D}}}$. □

The last proposition of this section follows quite naturally from the definition of $\equiv^B_{\mathcal{V}}$, which is why we omit the proof in this case.

Proposition 4. *Let $\Pi = (P, Q, \mathcal{P}^A_{\mathcal{V}}, \equiv^B_{\mathcal{V}})$ be an equivalence problem. Then, Π holds iff $(P, Q, \mathcal{P}^A_{\mathcal{V}}, \sqsubseteq^B_{\mathcal{V}})$ and $(Q, P, \mathcal{P}^A_{\mathcal{V}}, \sqsubseteq^B_{\mathcal{V}})$ jointly hold.*

4.2 Spoilers

Now we come to the generalisation of a *spoiler*. The structure was introduced by Eiter et al. [8] in the propositional case and its existence for a certain inclusion problem *disproves* the correspondence.

Definition 12. *Let P and Q be programs over $\mathcal{V} = (\mathbb{P}, \mathbb{D})$, $Y \subseteq HB_{\mathbb{P},C}$ an interpretation, $C \subseteq \mathbb{D}$ a set of constants, and $S \subseteq \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q))$. Then, the pair $(Y, S)_C$ is a* spoiler *for the correspondence problem $\Pi = (P, Q, \mathcal{P}^A_{\mathcal{V}}, \sqsubseteq^B_{\mathcal{V}})$ iff*

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

(ii) *for each $(Z, Z)_C \in S$, some non-total $(X, Z)_C \in S$ exists,*

(iii) $(Z, Z)_C \in S$ *iff* $(Z, Z)_C \in \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q))$, *and*

(iv) $(X, Z)_C \in S$ *implies* $(X, Y)_C \notin SE^A(P)$.

Intuitively, the interpretation Y in a spoiler $(Y, S)_C$ is an answer set of $P \cup R$ but not of $Q \cup R$, where R is semantically given by S.
The following theorem forms the link between correspondence problems and spoilers. The proof is analogous to the one for the propositional case.

Theorem 6. *Let P and Q be programs over $\mathcal{V} = (\mathbb{P}, \mathbb{D})$, and $A, B \subseteq \mathbb{P}$ sets of predicates. Then, the inclusion problem $(P, Q, \mathcal{P}^A_{\mathcal{V}}, \sqsubseteq^B_{\mathcal{V}})$ holds iff there is no spoiler for it.*

Proof. We start with the if-direction. Suppose some $R \in \mathcal{P}^A_{\mathcal{V}}$ such that $Y|_{HB_{B,\mathbb{D}}} \in AS(P \cup R)|_{HB_{B,\mathbb{D}}}$ and $Y|_{HB_{B,\mathbb{D}}} \notin AS(Q \cup R)|_{HB_{B,\mathbb{D}}}$. Without loss of generality, assume $Y \in AS(P \cup R)$. By definition, $Y \in AS(P \cup R)$ implies $Y \models grd(P \cup R, HU_{P\cup R})$ and $Y' \not\models grd(P \cup R, HU_{P\cup R})^Y$ for each $Y' \subset Y$. Since $HU_{P\cup R} \subseteq \mathbb{D}$ obviously holds, we get by Lemma 1 that $Y \models grd(P \cup R, C)$ and $Y' \not\models grd(P \cup R, C)^Y$ hold for any $C \subseteq \mathbb{D}$. Thus, (i), (ii), and (iii) of Definition 7 hold for $(Y, Y)_C$ and therefore $(Y, Y)_C \in SE^A(P \cup R)$. Furthermore, since $Y' \not\models grd(P \cup R, C)^Y$ holds for each $Y' \subset Y$, by (iii) of Definition 7, $(X, Y)_C \in SE^A(P \cup R)$ implies $X = Y$. By Proposition 1, $Y|_{HB_{B,\mathbb{D}}} \notin AS(Q \cup R)|_{HB_{B,\mathbb{D}}}$ implies that $\sigma^B_{Y,\mathcal{V}}(SE^A(Q \cup R))$ is non-total. $(J, J)_C \in \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q \cup R))$ implies $(J, J)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(Q \cup R))$, since $J \equiv^{A\cup B}_{\mathcal{V}} Y$ implies $J \equiv^B_{\mathcal{V}} Y$ by definition. From the non-totality of $\sigma^B_{Y,\mathcal{V}}(SE^A(Q \cup R))$, we get that $(J, J)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(Q \cup R))$ implies the existence of some $(I, J)_C \in \sigma^B_{Y,\mathcal{V}}(SE^A(Q \cup R))$. Since $(J, J)_C \in \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q \cup R))$ implies $J \equiv^{A\cup B}_{\mathcal{V}} Y$, $(I, J)_C \in \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q \cup R))$ holds and $\sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q \cup R))$ is thereby non-total as well. To summarise, the following observations hold:

(a) $(Y,Y)_C \in SE^A(P \cup R)$,

(b) $(X,Y)_C \in SE^A(P \cup R)$ implies $X = Y$, and

(c) $\sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q \cup R))$ is non-total.

From Lemma 5, it follows that $\sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q \cup R)) = \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q) \cap SE(R))$ and by Definition 11, $\sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q) \cap SE(R)) = \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q)) \cap \sigma^{A\cup B}_{Y,\mathcal{V}}(SE(R))$. Consider a $(Z,Z) \in \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q \cup R))$. By Definition 11, $Z \equiv^{A\cup B}_{\mathcal{V}} Y$ and thus $Z \equiv^{A}_{\mathcal{V}} Y$. From (a) and Lemma 5 we know that $(Y,Y)_C \in (SE^A(P) \cap SE(R))$ and therefore $(Y,Y)_C \in SE(R)$ and $(Y,Y)_C \in SE^A(P)$ hold. By Proposition 2, $(Y|_{HB_{A,\mathbb{D}}}, Y|_{HB_{A,\mathbb{D}}})_C \in SE(R)|_{HB_{A,\mathbb{D}}}$, which in turn implies $(Z,Z)_C \in SE(R)$ since $Z|_{HB_{A,\mathbb{D}}} = Y|_{HB_{A,\mathbb{D}}}$ and obviously $Z \subseteq Z$. So, for each $(Z,Z)_C \in \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q \cup R))$, $(Z,Z)_C \in SE(R)$ holds. So suppose $(Z,Z)_C \in \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q))$ and $(Z,Z)_C \notin \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q \cup R))$. Since $\sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q \cup R)) = \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q)) \cap \sigma^{A\cup B}_{Y,\mathcal{V}}(SE(R))$, the above would imply $(Z,Z)_C \notin \sigma^{A\cup B}_{Y,\mathcal{V}}(SE(R))$. We know that $Z \equiv^{A\cup B}_{\mathcal{V}} Y$, so $(Z,Z)_C \notin SE(R)$ holds. The latter is obviously a contradiction and thus $(Z,Z)_C \in \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q \cup R))$ implies $(Z,Z)_C \in \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q))$ for any Z. Since every $(Z,Z)_C \in \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q \cup R))$ is non-total, $\sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q))$ is non-total as well.
Now let $\mathcal{S}_Q$ be $\mathcal{S}_Q = \sigma^{A\cup B}_{Y,\mathcal{V}}(SE^A(Q))$. It remains to show that there is at least one non-total $S \subseteq \mathcal{S}_Q$ (to satisfy (ii)), for which (iii) and (iv) hold. Towards a contradiction, assume this is not the case. Such an S cannot exist iff there is a $(Z,Z)_C \in \mathcal{S}_Q$ for which (iv) cannot hold. In other words, there is a $(Z,Z)_C \in \mathcal{S}_Q$ such that $(X,Z)_C \in \mathcal{S}_Q$ implies $(X,Y)_C \in SE^A(P)$ for any $X \subset Z$. We now show that $(X,Z)_C \in SE^A(Q \cup R)$ implies $(X,Z)_C \in SE^A(P \cup R)$ for any $X \subset Y$. $(Z,Z)_C \in \mathcal{S}_Q$ implies $Z \equiv^{A\cup B}_{\mathcal{V}} Y$, which implies $Z \equiv^{A}_{\mathcal{V}} Y$. By Lemma 5, $SE^A(Q \cup R) = SE^A(Q) \cap SE(R)$, thus $(X,Z)_C \in SE^A(Q \cup R)$ implies $(X,Z)_C \in SE^A(Q)$ and $(X,Z)_C \in SE(R)$. Furthermore since $Z \equiv^{A}_{\mathcal{V}} Y$, $(X,Z) \in \mathcal{S}_Q$ holds. From the above we know that $(X,Z)_C \in \mathcal{S}_Q$ implies $(X,Y)_C \in SE^A(P)$. By Definition 7, $(X,Z)_C \in SE^A(Q \cup R)$ implies either $X = Z$ or $X \subset Z|_{HB_{A,\mathbb{D}}}$ and since $X \subset Z$, $X \subset Z|_{HB_{A,\mathbb{D}}}$ holds. Additionally, $Z \equiv^{A}_{\mathcal{V}} Y$ implies $X \subset Y|_{HB_{A,\mathbb{D}}}$ and thus $X \subset Y$ holds. By Proposition 2, we get that $(X,Z)_C \in SE(R)$ implies $(X|_{HB_{A,\mathbb{D}}}, Z|_{HB_{A,\mathbb{D}}})_C \in SE(R)|_{HB_{A,\mathbb{D}}}$, which in turn implies $(X,Y)_C \in SE(R)$, since $X \subset Y$ and $Z \equiv^{A}_{\mathcal{V}} Y$ both hold. Now we have established that $(X,Z)_C \in SE^A(Q \cup R)$ implies $(X,Y)_C \in SE^A(P)$ and $(X,Y)_C \in SE(R)$. By Lemma 5, we obtain $(X,Y)_C \in SE^A(P \cup R)$. From (b), we know that $(X,Y)_C \in SE^A(P \cup R)$ implies $X = Y$, but since $X \subset Y$, $(X,Y)_C \notin SE^A(P \cup R)$ holds. The latter implies $(X,Z)_C \notin SE^A(Q \cup R)$, which is a contradiction to (c). Therefore, there has to be a non-total $S \subseteq \mathcal{S}_Q$ such that (iii) and (iv) hold.

Now we continue with proving the only-if-direction.
Let $(Y,S)_C$ be a spoiler for $(P,Q,\mathcal{P}^A_{\mathcal{V}},\sqsubseteq^B_{\mathcal{V}})$.

If $S = \emptyset$, set $R = Y|_{HB_{A,\mathbb{D}}} \cup \{\leftarrow y \mid y \in (HB_{A,\mathbb{D}} \setminus Y)\}$. $Z \models R$ holds iff:

(1) $Z \supseteq Y|_{HB_{A,\mathbb{D}}}$, and

(2) $Z \cap (HB_{A,\mathbb{D}} \setminus Y) = \emptyset$.

(1) and (2) obviously hold for Y. Therefore, $Y \models R$, and since $R = R^Y$ holds by construction, $Y \models R^Y$ holds. For any $Y' \subset Y$, (2) cannot hold and thus $Y' \not\models R^Y$ holds. From Definition 12, we know that $(Y,Y)_C \in SE^A(P)$. By Definition 7, the latter implies $Y \models grd(P, HU_P)$. Since $Y \models grd(P, HU_P)$ and $Y \models R$ both hold, $Y \models grd(P \cup R, HU_{P \cup R})$ holds as well. We know that $Y' \not\models R^Y$ holds for any $Y' \subset Y$, therefore $Y' \not\models grd(P \cup R)^Y$ has to hold and thus $Y \in AS(P \cup R)$ holds by definition. $S = \emptyset$ implies $(Y,Y)_C \notin SE^A(Q)$, by Definition 12. This implies that either $Y \not\models grd(Q,C)$ or there is a $Y' \subset Y$ with $Y' \equiv_{\mathcal{V}}^A Y$ and $Y' \models grd(Q,C)^Y$. Either way, the above implies $Y \notin AS(Q \cup R)$ and thus $Y|_{HB_{B,\mathbb{D}}} \notin AS(Q \cup R)|_{HB_{B,\mathbb{D}}}$ holds by Proposition 2. To summarise, we have

(a) $Y \in AS(P \cup R)$ and

(b) $Y|_{HB_{B,\mathbb{D}}} \notin AS(Q \cup R)|_{HB_{B,\mathbb{D}}}$,

which is what we wanted to show.

So consider $S \neq \emptyset$. We again need to show that (a) and (b) hold. Set $R = P_{S,A,C}$, where $P_{S,A,C}$ is the defined the same as in Proposition 3. We start with (a). $(Y,Y)_C \in SE^A(P)$ follows from Definition 12. Consider a $(Z,Z)_C \in S$. $(Z,Z)_C \in S$ implies $(Z,Z)_C \in \sigma_{Y,\mathcal{V}}^{A \cup B}(SE^A(Q))$ and thus $Z \equiv_{\mathcal{V}}^{A \cup B} Y$. By Proposition 3, $S|_{HB_{A,\mathbb{D}}} = SE(R)|_{HB_{A,\mathbb{D}}}$ and since $(Z,Z)_C \in S$ implies $(Z|_{HB_{A,\mathbb{D}}}, Z|_{HB_{A,\mathbb{D}}})_C \in S|_{HB_{A,\mathbb{D}}}$ by Proposition 2, $(Z|_{HB_{A,\mathbb{D}}}, Z|_{HB_{A,\mathbb{D}}})_C \in SE(R)|_{HB_{A,\mathbb{D}}}$ holds. Also by Proposition 2, the latter implies $(Y,Y)_C \in SE(R)$, since $Z \equiv_{\mathcal{V}}^A Y$ and obviously $Y \subseteq Y$. By Lemma 5, $(Y,Y)_C \in SE^A(P)$ and $(Y,Y)_C \in SE(R)$ imply $(Y,Y)_C \in SE^A(P \cup R)$. It remains to show that $(X,Y)_C \in SE^A(P \cup R)$ implies $X = Y$, because then $Y \in AS(P \cup R)$ follows from (iii) of Definition 7 and Lemma 1. So, towards a contradiction, consider a $X \subset Y$ with $(X,Y)_C \in SE^A(P \cup R)$. From Lemma 5, we get $(X,Y)_C \in SE^A(P)$ and $(X,Y)_C \in SE(R)$ and by Proposition 2, $(X,Y)_C \in SE(R)$ implies $(X|_{HB_{A,\mathbb{D}}}, Y|_{HB_{A,\mathbb{D}}})_C \in SE(R)|_{HB_{A,\mathbb{D}}}$. Since $S|_{HB_{A,\mathbb{D}}} = SE(R)|_{HB_{A,\mathbb{D}}}$ by Proposition 3, $(X|_{HB_{A,\mathbb{D}}}, Y|_{HB_{A,\mathbb{D}}})_C \in S|_{HB_{A,\mathbb{D}}}$ holds and by Proposition 2, the latter implies $(X,Z)_C \in S$ for any $Z \equiv_{\mathcal{V}}^A Y$. Since S is not empty, there exists a Z with $Z \equiv_{\mathcal{V}}^{A \cup B} Y$ and thus there has to be a $(X,Z)_C \in S$. But by (iv) of Definition 7, this would imply $(X,Y)_C \notin SE^A(P)$, which is obviously a contradiction.
To show (b), suppose a Z with $Z \equiv_{\mathcal{V}}^B Y$. We have to consider two cases: Either

(1) $Z \not\equiv_{\mathcal{V}}^A Y$; or

(2) $Z \equiv_{\mathcal{V}}^A Y$.

If (1), then $(Z,Z)_C \notin S$ since $Z \not\equiv_{\mathcal{V}}^{A \cup B} Y$. By Proposition 2, $(Z,Z)_C \notin S$ implies $(Z|_{HB_{A,\mathbb{D}}}, Z|_{HB_{A,\mathbb{D}}})_C \notin S|_{HB_{A,\mathbb{D}}}$ and because $S|_{HB_{A,\mathbb{D}}} = SE(R)|_{HB_{A,\mathbb{D}}}$ by Proposition 3, $(Z|_{HB_{A,\mathbb{D}}}, Z|_{HB_{A,\mathbb{D}}}) \notin SE(R)|_{HB_{A,\mathbb{D}}}$ holds. Also by Proposition 2, $(Z|_{HB_{A,\mathbb{D}}}, Z|_{HB_{A,\mathbb{D}}})_C \notin SE(R)|_{HB_{A,\mathbb{D}}}$ implies $(Z,Z)_C \notin SE(R)$ and thus $Z \notin AS(Q \cup R)$. Since $Z \equiv_{\mathcal{V}}^B Y$ holds, $Y|_{HB_{B,\mathbb{D}}} \notin AS(Q \cup R)|_{HB_{B,\mathbb{D}}}$ obviously holds too.

It remains to show (2). Towards a contradiction, suppose there is a Z such that $Z \in AS(Q \cup R)$ holds. $Z \in AS(Q \cup R)$ implies by definition that $Z \models grd(Q, HU_Q)$ and $Z' \not\models grd(Q, HU_Q)^Y$ hold for any $Z' \subset Z$. Hence, $(Z, Z) \in SE^A(Q)$ holds by Definition 7 and Lemma 1. Furthermore, by Definition 7, every $(X, Z)_C \in SE^A(Q)$ with $X \subset Z$ implies an $X' \equiv^A_{\mathcal{V}} X$ with $X' \subset Z$ such that $X' \models grd(Q, C)^Y$. Thus $(X, Z)_C \in SE^A(Q)$ implies $(X, Z)_C \notin SE(R)$, because otherwise $Z \in AS(Q \cup R)$ would not hold. Since $S|_{HB_{A,\mathbb{D}}} = SE(R)|_{HB_{A,\mathbb{D}}}$, $(X, Z)_C \notin SE(R)$ implies $(X, Z)_C \notin S$ by Proposition 2. The latter contradicts the non-totality of S (violating (ii) of Definition 12), and therefore $Z \notin AS(Q \cup R)$ has to hold. Hence $Y|_{HB_{B,\mathbb{D}}} \notin AS(Q \cup R)|_{HB_{B,\mathbb{D}}}$ holds. □

Example 5. *Recall program P from Example 1 and program Q from Example 3:*

$$P = \left\{ \begin{array}{l} accepted(x) \leftarrow applicant(x), not\ rejected(x), \\ rejected(x) \leftarrow applicant(x), not\ accepted(x), \\ applicant(jane) \leftarrow, \\ applicant(bob) \leftarrow \end{array} \right\},$$

$$Q = \left\{ \begin{array}{l} accepted(x) \vee rejected(x) \leftarrow applicant(x), \\ applicant(jane) \leftarrow, \\ applicant(bob) \leftarrow \end{array} \right\}.$$

Furthermore, consider the sets

$$A = B = \{accepted(\cdot),\ rejected(\cdot)\}.$$

The correspondence problem $\Pi = (Q, P, \mathcal{P}^A_{\mathcal{V}}, \sqsubseteq^B_{\mathcal{V}})$ does not hold because there exists a spoiler $(Y, S)_C$, where:

$$Y = \left\{\ ac(j), re(j), ap(j), ac(b), re(b), ap(b)\ \right\},$$

$$S = \left\{ \begin{array}{l} (\emptyset, \{ap(j), ap(b), re(j), ac(j), re(b), ac(b)\}), \\ (\{ac(j)\}, \{ap(j), ap(b), re(j), ac(j), re(b), ac(b)\}), \\ (\{ac(b)\}, \{ap(j), ap(b), re(j), ac(j), re(b), ac(b)\}), \\ (\{re(j)\}, \{ap(j), ap(b), re(j), ac(j), re(b), ac(b)\}), \\ (\{re(b)\}, \{ap(j), ap(b), re(j), ac(j), re(b), ac(b)\}), \\ (\{ac(j), re(j)\}, \{ap(j), ap(b), re(j), ac(j), re(b), ac(b)\}), \\ (\{ac(b), re(b)\}, \{ap(j), ap(b), re(j), ac(j), re(b), ac(b)\}), \\ (\{ap(j), ap(b), re(j), ac(j), re(b), ac(b)\}, \{ap(j), ap(b), re(j), ac(j), re(b), ac(b)\}) \end{array} \right\},$$

$$C = \{j, b\}.$$

An intermediate conseqence of the theorem above and Proposition 4 is the next result.

Corollary 1. *Let* $\Pi = (P, Q, \mathcal{P}^A_\mathcal{V}, \equiv^B_\mathcal{V})$ *be an equivalence problem. Then,* Π *holds iff neither* $(P, Q, \mathcal{P}^A_\mathcal{V}, \sqsubseteq^B_\mathcal{V})$ *nor* $(Q, P, \mathcal{P}^A_\mathcal{V}, \sqsubseteq^B_\mathcal{V})$ *has a spoiler.*

4.3 Counterexamples

Another structure disproving correspondence is that of a *counterexample*, which follows more or less naturally from the definition of relative strong equivalence.

Definition 13. *Let* P *and* Q *be programs over* $\mathcal{V} = (\mathbb{P}, \mathbb{D})$*,* $R \in \mathcal{P}^A_\mathcal{V}$ *a program, and* $M \in AS(P \cup R)$ *an answer set. Then, the pair* (R, M) *is a* counterexample *for the correspondence problem* $\Pi = (P, Q, \mathcal{P}^A_\mathcal{V}, \sqsubseteq^B_\mathcal{V})$ *iff*

(*i*) $M \in AS(P \cup R)$,

(*ii*) $M|_{HB_{B,\mathbb{D}}} \notin AS(Q \cup R)|_{HB_{B,\mathbb{D}}}$, *and*

(*iii*) $AS(P \cup R) \not\sqsubseteq^B_\mathcal{V} AS(Q \cup R)$.

The following theorem shows the connection between spoilers and counterexamples and follows directly from Theorem 6, Proposition 3, and Definition 12.

Theorem 7. *Suppose* $(Y, S)_C$ *is a spoiler with* $S \neq \emptyset$*. Then,* $(P_{S,A,C}, Y)$ *is a counterexample for* $\Pi = (P, Q, \mathcal{P}^A_\mathcal{V}, \sqsubseteq^B_\mathcal{V})$*, where* $P_{S,A,C}$ *is defined as in Proposition 3.*

Proof. Towards a contradiction, assume $(Y, S)_C$ is a spoiler for $\Pi = (P, Q, \mathcal{P}^A_\mathcal{V}, \sqsubseteq^B_\mathcal{V})$ but $(P_{S,A,C}, Y)$ is not a counterexample for Π.
If $(P_{S,A,C}, Y)$ is not a counterexample of Π, then either

(i) $Y \notin AS(P \cup P_{S,A,C})$,

(ii) $Y|_{HB_{B,\mathbb{D}}} \in AS(Q \cup P_{S,A,C})|_{HB_{B,\mathbb{D}}}$, or

(iii) $AS(P \cup P_{S,A,C}) \sqsubseteq^B_\mathcal{V} AS(Q \cup P_{S,A,C})$.

In the second direction of the proof of Theorem 6, we have already shown that (i) and (ii) cannot hold. That leaves (iii).
Since (i) does not hold, $Y \in AS(P \cup P_{S,A,C})$ has to hold which obviously implies $Y|_{HB_{B,\mathbb{D}}} \in AS(P \cup P_{S,A,C})|_{HB_{B,\mathbb{D}}}$. Now, since $Y|_{HB_{B,\mathbb{D}}} \in AS(P \cup P_{S,A,C})|_{HB_{B,\mathbb{D}}}$ but $Y|_{HB_{B,\mathbb{D}}} \notin AS(Q \cup P_{S,A,C})|_{HB_{B,\mathbb{D}}}$, (iii) cannot hold by definition of $\sqsubseteq^B_\mathcal{V}$. □

Example 6. *Recall the spoiler* $(Y, S)_C$ *and the correspondence problem* $\Pi = (Q, P, \mathcal{P}^A_{\mathcal{V}}, \sqsubseteq^B_{\mathcal{V}})$ *from Example 5. By Proposition 3 we get*

$$P_{S,A,C} = \left\{ \begin{array}{l} \leftarrow ac(j), \mathit{not}\ re(j), \mathit{not}\ ac(b), \mathit{not}\ re(b), \\ \leftarrow ac(b), \mathit{not}\ ac(j), \mathit{not}\ re(j), \mathit{not}\ re(b), \\ \leftarrow re(j), \mathit{not}\ ac(j), \mathit{not}\ ac(b), \mathit{not}\ re(b), \\ \leftarrow re(b), \mathit{not}\ ac(j), \mathit{not}\ ac(b), \mathit{not}\ re(j), \\ \leftarrow ac(j), re(j), \mathit{not}\ ac(b), \mathit{not}\ re(b), \\ \leftarrow re(j), ac(b), \mathit{not}\ ac(j), \mathit{not}\ re(b), \\ \leftarrow ac(b), re(b), \mathit{not}\ ac(j), \mathit{not}\ re(j), \\ \leftarrow re(j), re(b), \mathit{not}\ ac(j), \mathit{not}\ ac(b), \\ \leftarrow ac(j), ac(b), \mathit{not}\ re(j), \mathit{not}\ re(b), \\ \leftarrow ac(j), re(b), \mathit{not}\ re(j), \mathit{not}\ ac(b), \\ \leftarrow ac(j), re(j), ac(b), \mathit{not}\ re(b), \\ \leftarrow ac(j), re(j), re(b), \mathit{not}\ ac(b), \\ \leftarrow re(j), re(b), ac(b), \mathit{not}\ ac(j), \\ \leftarrow ac(j), ac(b), re(b), \mathit{not}\ re(j), \\ ac(j) \vee re(b) \leftarrow ac(b), re(j), \\ ac(j) \vee ac(b) \leftarrow re(j), re(b), \\ re(j) \vee re(b) \leftarrow ac(j), ac(b), \\ re(j) \vee ac(b) \leftarrow ac(j), re(b), \\ ac(b) \leftarrow ac(j), re(j), re(b), \\ re(b) \leftarrow ac(j), re(j), ac(b), \\ ac(j) \leftarrow ac(b), re(j), re(b), \\ re(j) \leftarrow ac(j), ac(b), re(b) \end{array} \right\}.$$

Appending $P_{S,A,C}$ *to the programs* P *and* Q*, we obtain the following answer sets:*

$$AS(P \cup P_{S,A,C}) = \emptyset$$

and

$$AS(Q \cup P_{S,A,C}) = \{\{ap(j), ap(b), ac(j), re(j), ac(b), re(b)\}\}.$$

Obviously, $Y \in AS(Q \cup P_{S,A,C})$ *and* $Y|_{HB_{B,\mathbb{D}}} \notin AS(P \cup P_{S,A,C})|_{HB_{B,\mathbb{D}}}$ *both hold. Therefore,* $(P_{S,A,C}, Y)$ *is a counterexample for* Π*.*

CHAPTER 5

Related Work

There are multiple related notions of program equivalence, among them *visible equivalence* [14], *hyperequivalence* [15], and *strong persistence* [16].
The first one, visible equivalence, was introduced by Janhunen and Oikarinen in 2007 and is an extension of ordinary equivalence with a form of projection. They restrict the equivalence to so-called *visible atoms*, but in difference to the projection used in this work, their concept requires the existence of a bijection between the visible atoms of the answer sets of the compared programs.
Hyperequivalence was introduced by Woltran and also goes back to 2007. The concept of hyperequivalence is very powerful. In fact, several other equivalences — including relative strong or uniform equivalence — can be viewed as special cases of hyperequivalence. In simple terms, hyperequivalence enables the restriction of equivalence to specific atoms in the head and the bodies of the programs being compared. So, compared to relative strong or uniform equivalence, the context class is not just restricted to a specific set of atoms, but every rule head is restricted to a set of atoms and every rule body is restricted to a possibly different set of atoms.
The last one mentioned above is also the newest one. Strong persistence was contrived by Knorr and Alferes in 2014 and is used in the context of forgetting atoms in logic programs. Strong persistence is basically the same as relative strong equivalence with projection, the only technical difference is that they only compare the original program with a version of itself where certain atoms have been "forgotten".
It should be noted that every one of the above equivalence notions were so far only defined for the propositional case.

Aside from notions of program equivalence, there is also other related work which should be mentioned. Most notably, relativised strong equivalence has been introduced for *equilibrium logic* by Pearce, Tompits, and Woltran [17]. Equilibrium logic goes back to Pearce [18] and is closely related to the the logic of here-and-there and Turner's SE-models. Models in equilibrium logic are special cases of models in the logic of here-and-there and correspond with answer sets. Hence, equilibrium logic generalises the

ASP paradigm to arbitrary propositional theories. In the second chapter of this thesis, we briefly mentioned that the concept of strong equivalence was contrived by Lifschitz, Pearce, and Valverde. In fact, they used equilibrium logic to do so. For a more thorough introduction to equilibrium logic, the reader is referred to the comprehensive overview paper by Pearce [19]. Another important topic related to this thesis are complexity results. Eiter et al. [8] have shown that checking relativised strong equivalence with projection in the propositional case is Π_4^P-complete. Furthermore, it has been shown that relativised strong equilance is undecideable in the general non-ground case [9].
Lastly, there is another work related to this thesis. Pührer and Tompits used the propositional version of relative strong equivalence with projection to elimate disjunction and negation from programs [20]. Their approach is called *casting* and is based on model-theoretic conditions that hold precisely when a "simpler" program exists. Furthermore, they provide a way to obtain the reduced program.

CHAPTER 6

Conclusion and Future Work

In this thesis, we generalised the concepts of relativised strong equivalence and relativised strong equivalence with projection to the non-ground setting. We did this by lifting and slightly modifying existing model-theoretic characterisations. In particular, we generalised the concept of relativised SE-models. Furthermore, we have extended the correspondence framework introduced by Eiter et al. [8] to cover the generalised version of relative equivalence with and without projection, although most of the groundwork was already laid down by Oetsch et al. [9]. Nonetheless, we lifted the notion of a spoiler — which refutes a correspondence between programs — to non-ground programs, and thereby paving the way for correspondence checking in the non-ground case.

There are numerous open topics left for future work. For example, in their work about relativised uniform eqivalence with projection, Oetsch et al. [9] provided translations of their model-based characterisations into *second-order logic*. Such translations would also be quite valuable for relativised strong equivalence with projection, since the existence of solvers for problems in second-order logic would open up an automated way for checking program correspondences.

Lastly, work could be done in the context of equilibrium logic and hyperequivalence. We already mentioned that Pearce et al. [17] have introduced the concept of relativised strong equivalence to equilibrium logic, but they did not cover projection. Also, equilibirium logic was recently generalised for a subset of first-order logic by Pearce and Valverde [21]. An interesting task would be the introduction of relativised strong equivalence with projection to the so-called *quantified equilibrium logic*.

As for hyperequivalence, so far it only exists in the realm of ground programs. A lifting of its concepts to the non-ground setting — analogous to this work — would be a worthwhile endeavour.

Bibliography

[1] Michael Gelfond and Vladimir Lifschitz, "The stable model semantics for logic programming", in *Proceedings of the 5th International Conference on Logic Programming*. 1988, pp. 1070–1080, MIT Press.

[2] Martin Gebser, Benjamin Kaufmann, André Neumann, and Torsten Schaub, "clasp: A conflict-driven answer set solver", in *Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning*. 2007, vol. 4483 of *LNCS*, pp. 260–265, Springer.

[3] Nicola Leone, Gerald Pfeifer, Wolfgang Faber, Thomas Eiter, Georg Gottlob, Simona Perri, and Francesco Scarcello, "The dlv system for knowledge representation and reasoning", *ACM Trans. Comput. Logic*, vol. 7, pp. 499–562, 2006.

[4] Thomas Eiter, Giovambattista Ianni, and Thomas Krennwallner, "Answer set programming: A primer", in *Reasoning Web. Semantic Technologies for Information Systems*, pp. 40–110. Springer, 2009.

[5] Vladimir Lifschitz, David Pearce, and Agustín Valverde, "Strongly equivalent logic programs", *ACM Transactions on Computational Logic*, vol. 2, pp. 526–541, 2001.

[6] Thomas Eiter, Michael Fink, Hans Tompits, and Stefan Woltran, "Strong and uniform equivalence in answer-set programming: Characterizations and complexity results for the non-ground case", in *Proceedings of the 20th National Conference on Artificial Intelligence*. 2005, vol. 2, pp. 695–700, AAAI Press.

[7] Stefan Woltran, "Characterizations for relativized notions of equivalence in answer set programming", in *Proceedings of the 9th European Conference on Logics in Artificial Intelligence*. 2004, vol. 3229 of *LNCS*, pp. 161–173, Springer.

[8] Thomas Eiter, Hans Tompits, and Stefan Woltran, "On solution correspondences in answer set programming", in *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. 2005, pp. 97–102, Professional Book.

[9] Johannes Oetsch and Hans Tompits, "Program correspondence under the answer-set semantics: The non-ground case", in *Proceedings of the 24th International Conference on Logic Programming*. 2008, vol. 5366 of *LCNS*, pp. 591–605, Springer.

[10] Teodor C. Przymusinski, "Stable semantics for disjunctive programs", *New Generation Computing*, vol. 9, pp. 401–424, 1991.

[11] Michael Gelfond and Vladimir Lifschitz, "Classical negation in logic programs and disjunctive databases", *New Generation Computing*, vol. 9, pp. 365–385, 1991.

[12] Thomas Eiter and Michael Fink, "Uniform equivalence of logic programs under the stable model semantics", in *Proceedings 19th International Conference on Logic Programming*. 2003, vol. 2916 of *LCNS*, pp. 224–238, Springer.

[13] Hudson Turner, "Strong equivalence made easy: Nested expressions and weight constraints", *Theory and Practice of Logic Programming*, vol. 3, pp. 602–622, 2003.

[14] Tomi Janhunen and Emilia Oikarinen, "Automated verification of weak equivalence within the SMODELS system", *Theory and Practice of Logic Programming*, vol. 7, pp. 697–744, 2007.

[15] Stefan Woltran, "A common view on strong, uniform, and other notions of equivalence in answer-set programming", *Theory and Practice of Logic Programming*, vol. 8, pp. 217–234, 2008.

[16] Matthias Knorr and José Júlio Alferes, "Preserving strong equivalence while forgetting", in *Proceedings of the 14th European Conference on Logics in Artificial Intelligence*. 2014, vol. 8761 of *LNCS*, pp. 412–425, Springer.

[17] David Pearce, Hans Tompits, and Stefan Woltran, "Relativised equivalence in equilibrium logic and its applications to prediction and explanation: Preliminary report", in *Proceedings of the LPNMR'07 Workshop on Correspondence and Equivalence for Nonmonotonic Theories*, 2007, vol. 265.

[18] David Pearce, "A new logical characterisation of stable models and answer sets", in *Proceedings of the 2nd International Workshop of Non-Monotonic Extensions of Logic Programming (NMELP'96)*. 1997, vol. 1216 of *LNCS*, pp. 57–70, Springer.

[19] David Pearce, "Equilibrium logic", *Annals of Mathematics and Artificial Intelligence*, vol. 47, pp. 3–41, 2006.

[20] Jörg Pührer and Hans Tompits, "Casting away disjunction and negation under a generalisation of strong equivalence with projection", in *Proceeding of the 10th International Conference on Logic Programming and Nonmonotonic Reasoning*. 2009, vol. 5753 of *LNCS*, pp. 264–276, Springer.

[21] David Pearce and Agustín Valverde, "Quantified equilibrium logic and foundations for answer set programs", in *Proceedings of the 24th International Conference on Logic Programming*. 2008, LCNS, pp. 546–560, Springer.