NINTH INTERNATIONAL CONFERENCE ON
LOGIC PROGRAMMING AND NONMONOTONIC REASONING (LPNMR 2007)

Workshop on Correspondence and Equivalence for Nonmonotonic Theories (CENT 2007)

Working Notes

David Pearce, Axel Polleres, Agustín Valverde, Stefan Woltran (editors)

Preface

This volume consists of the contributions presented at the Workshop Correspondence and Equivalence for Nonmonotonic Theories, CENT 2007, colocated with LPNMR 2007 in Tempe, Arizona, USA on May 14 2007.

The systematic study of intertheory relations such as strong and uniform equivalence has recently become an active sub-area of research in the field of LPNMR. Various kinds of correspondence relations that may hold between logic programs or between nonmonotonic theories have been analysed and shown to be of practical relevance for theory or program transformation, optimisation and modularity. Several systems for verifying such relations have already been implemented. The papers in this volume explore this topic further and take it in several new directions.

We would like to express our warm thanks to the LPNMR programme chairs, Gerhard Brewka and John Schlipf, for agreeing to host this event in Tempe. A special thanks also to goes to Chitta Baral, the local organiser of LPNMR, for his help in accommodating the workshop and printing these proceedings. Finally, we thank the contributors for their efforts to improve our understanding of this developing area and the programme committee members for their efforts to help improve the quality of our research.

April 2007

David Pearce
Axel Polleres
Agustín Valverde
Stefan Woltran

Workshop Organization

Steering committee

David Pearce	(*Universidad Rey Juan Carlos, Spain*)
Axel Polleres	(*Universidad Rey Juan Carlos, Spain*)
Agustín Valverde	(*Universidad de Málaga, Spain*)
Stefan Woltran	(*Vienna University of Technology, Austria*)

Program Committee

Wolfgang Faber	(*University of Calabria, Italy*)
Katsumi Inoue	(*National Institute of Informatics, Japan*)
Vladimir Lifschitz	(*University of Texas at Austin, USA*)
Fangzhen Lin	(*Hong Kong University of Science and Technology, China*)
Emilia Oikarinen	(*Helsinki University of Technology, Finland*)
Riccardo Rosati	(*Universita di Roma "La Sapienza", Italy*)
Hans Tompits	(*Vienna University of Technology*)

Additional Referees

Pedro Cabalar
Johannes Oetsch

Table of Contents

Yet Another Proof of the Strong Equivalence Between Propositional Theories and Logic Programs

Joohyung Lee and Ravi Palla

School of Computing and Informatics
Arizona State University, Tempe, AZ, USA
{joolee, Ravi.Palla}@asu.edu

Abstract. Recently, the stable model semantics was extended to the syntax of arbitrary propositional formulas, which are beyond the traditional rule form. Cabalar and Ferraris, as well as Cabalar, Pearce, and Valverde, showed that any propositional theory under the stable model semantics can be turned into a logic program. In this note, we present yet another proof of this result. Unlike the other approaches that are based on the logic of here-and-there, our proof uses familiar properties of classical logic. Based on this idea, we present a prototype implementation for computing stable models of propositional theories using the answer set solver DLV. We also note that every first-order formula under the stable model semantics is strongly equivalent to a prenex normal form whose matrix has the form of a logic program.

1 Introduction

Recently, the stable model semantics was extended to the syntax of arbitrary propositional formulas, which are beyond the traditional rule form [1, 2]. Ferraris [2] showed that nonmonotone aggregates can be naturally expressed in the extended syntax. On the other hand, Cabalar and Ferraris [3] showed that every propositional theory under the stable model semantics is strongly equivalent [4] to a logic program. They provided two proofs based on the logic of here-and-there, one by syntactic transformation, and the other by constructing a logic program using countermodels of the theory. An approach similar to the first proof was taken in [5], where the authors presented a set of rules for rewriting a propositional theory into a disjunctive logic program. These rules are an extension of the rules for turning a program with nested expressions into a logic program [6], which led to an implementation NLP [7]. The system is essentially a preprocessor to the answer set solver DLV [1] for handling programs with nested expressions.

In this note, we present yet another proof of the theorem on strong equivalence between propositional theories and logic programs. Unlike the other approaches that are based on the logic of here-and-there, our proof is based on an operator that characterizes strong equivalence in terms of classical logic, using an extended signature with two groups of atoms, the original one corresponding to the "there" world, and a group of newly introduced atoms referring to the "here" world. This not only shows that the reduction is possible, but also tells us how to *generate* strongly equivalent logic programs based on equivalence in classical logic.

[1] http://www.dbai.tuwien.ac.at/proj/dlv/ .

The reduction idea has led us to develop a prototype implementation, which we call F2LP,[2] that computes the stable models of an arbitrary propositional theory. Similar to NLP, the system turns a propositional theory into a disjunctive logic program and calls DLV.

We also apply the reduction idea to first-order formulas under the new definition of stable model semantics, recently proposed in [8]. We show that any first-order theory under the stable model semantics is strongly equivalent to a prenex normal form whose matrix has the form of a logic program. Thus the syntactic difference of arbitrarily nested connectives and quantifiers is not essential between the language proposed in [8] and logic programs. On the other hand, since the prenex normal form may contain existential quantifiers, it is different from a logic program, where all variables are assumed to be universally quantified.

In the next section, we review the definition of stable models for arbitrary propositional formulas as well as the definition of strong equivalence between propositional formulas, and present how to find a logic program that is strongly equivalent to a given formula. In Section 3, we present a simpler transformation, and in Section 4, we extend the reduction idea to arbitrary first-order formulas and note that every first-order theory is strongly equivalent to a prenex normal form. In Section 5, we present a prototype implementation of computing the stable models of propositional theories.

2 Reducing propositional formulas to logic programs

We first review the definition of a stable model proposed in [8], by restricting attention to the propositional case. This definition is essentially the same as the encoding of formulas of equilibrium logic by quantified Boolean formulas given in [9], and is equivalent to the fixpoint definition of a stable model proposed in [2].

Let F be a propositional formula and σ a signature consisting of all atoms $p_1, \dots, p_n$ occurring in F. By $\mathrm{SM}[F]$ we denote the second-order propositional sentence

$$F \wedge \forall \mathbf{u}((\mathbf{u} < \mathbf{p}) \rightarrow \neg F^*(\mathbf{u})),$$

where $\mathbf{p}$ stands for the tuple $p_1, \dots, p_n$, $\mathbf{u}$ is a tuple of n distinct propositional variables $u_1, \dots, u_n$, equation $\mathbf{u} < \mathbf{p}$ stands for

$$(u_1 \rightarrow p_1) \wedge \cdots \wedge (u_n \rightarrow p_n) \wedge \neg((p_1 \rightarrow u_1) \wedge \cdots \wedge (p_n \rightarrow u_n))$$

as in the definition of circumscription, and $F^*(\mathbf{u})$ is defined recursively, as follows:

- $p_i^* = u_i$;
- $\bot^* = \bot$;
- $(F \odot G)^* = F^* \odot G^*$, where $\odot \in \{\wedge, \vee\}$;
- $(F \rightarrow G)^* = (F^* \rightarrow G^*) \wedge (F \rightarrow G)$.

We regard $\neg F$ as shorthand for $F \rightarrow \bot$. Note that $\neg$ corresponds to *not* in the logic program syntax. For instance, the rule

$$p \leftarrow \textit{not}\ q$$

[2] `http://peace.eas.asu.edu/f2lp` .

is identified with the formula

$$\neg q \rightarrow p\ .$$

The operator $F \mapsto F^*(\mathbf{u})$ replaces each atom with the corresponding propositional variable, and commutes with all propositional connectives except implication. If, in the definition of this operator, we drop the second conjunctive term in the clause for implication, then $F^*(\mathbf{u})$ will turn into the formula $F(\mathbf{u})$ referred to in the definition of circumscription [10, 11]. A model of F is *stable* if it satisfies $\mathrm{SM}[F]$.

According to [12, Section 2.6], a (propositional) formula F is said to be strongly equivalent to a formula G if any formula F' that contains an occurrence of F has the same stable models as the formula G' obtained from F' by replacing that occurrence with G. This condition is more general than the original definition from [4] not only because it is applicable to arbitrary formulas, but also because F is allowed here to be any subformula of F', not necessarily a "subconjunction."

Our reduction idea is based on the following proposition from [8], which generalizes the main theorem from [13], stating that the strong equivalence between two formulas F and G can be characterized in terms of equivalence (in classical logic) between F^* and G^*. Let σ' be a signature consisting of distinct atoms $\{p'_1, \dots, p'_n\}$ that are disjoint from σ, and let $\mathbf{p}'$ stand for the tuple $p'_1, \dots, p'_n$. Formula $F^*(\mathbf{p}')$ is obtained from $F^*(\mathbf{u})$ by substituting the atoms $\mathbf{p}'$ for propositional variables $\mathbf{u}$. Thus $F^*(\mathbf{p}')$ is a transformation of F in signature $\sigma \cup \sigma'$. Equation $\mathbf{p}' \leq \mathbf{p}$ stands for

$$(p'_1 \rightarrow p_1) \wedge \cdots \wedge (p'_n \rightarrow p_n)$$

as in the definition of circumscription.

Proposition 1 *[8, Proposition 5] Formulas F and G of signature σ are strongly equivalent iff*

$$\mathbf{p}' \leq \mathbf{p} \rightarrow (F^*(\mathbf{p}') \leftrightarrow G^*(\mathbf{p}')) \qquad (1)$$

is a tautology.

As usual, a formula F is in *negation normal form* if, for every subformula $G \rightarrow H$ of F, formula G is an atom, and H is $\bot$. An occurrence of a formula G in a formula F is *positive* if the number of implications in F containing the occurrence of G in the antecedent is even, and *negative* otherwise.

Definition 1. *An implication $F \rightarrow G$ of signature $\sigma \cup \sigma'$ is called a* canonical implication *if F and G are formulas in negation normal form such that every occurrence of atoms from σ' is positive, and every occurrence of atoms from σ is negative.*

For example,

$$p' \wedge q \rightarrow r$$

is not canonical, while

$$(p' \vee (\neg q \wedge r')) \rightarrow (s' \wedge \neg p) \qquad (2)$$

is canonical.

Given a formula F of signature $\sigma \cup \sigma'$, by $R(F)$ we denote the formula of signature σ that is obtained from F by dropping all occurrences of $'$ in F. Note that $R(F)$, where F is a canonical implication, can be identified with a logic program with nested expressions [6], by identifying '$\neg$' with *not*, '$\wedge$' with ',', and '$\vee$' with ';'. For instance, in logic programming notation, when F is (2), $R(F)$ can be written as

$$s,\ \textit{not}\, p\ \leftarrow\ p\ ;\ (\textit{not}\, q,\ r)\ .$$

The following proposition tells us how to obtain a logic program that is strongly equivalent to a given formula.

Proposition 2 *Given a formula F, if G is a conjunction of canonical implications that is equivalent to F^*, then F and $R(G)$ are strongly equivalent.*[3]

The proof of Proposition 2 uses the observation that

$$\mathbf{p}' \leq \mathbf{p} \rightarrow (F^* \leftrightarrow (R(G))^*) \tag{3}$$

is a tautology. In view of Proposition 1, it follows that F and $R(G)$ are strongly equivalent. The fact that every propositional theory is strongly equivalent to a logic program follows from the fact that every formula F^* can be equivalently rewritten as a conjunction of canonical implications. One way to do this is by forming a conjunctive normal form (CNF) of $F^*(\mathbf{p}')$, and then converting each of its clauses into a canonical implication as follows. Given a clause C of signature $\sigma\cup\sigma'$, by $\mathit{Tr}(C)$ we denote an implication whose antecedent is the conjunction of

– all p' where $\neg p' \in C$, and
– all $\neg p$ where $p \in C$,

and whose consequent is the disjunction of

– all p' where $p' \in C$, and
– all $\neg p$ where $\neg p \in C$.

For instance, if C is $(p'\vee\neg q'\vee r\vee\neg s)$, then $\mathit{Tr}(C)$ is $(q'\wedge\neg r \rightarrow p'\vee\neg s)$. We can take G in the statement of Proposition 2 to be the conjunction of $\mathit{Tr}(C)$ for all clauses C in a conjunctive normal form of F^*. In view of Proposition 1, it follows that every formula is strongly equivalent to a logic program whose rules have the form

$$a_1;\dots;a_k;\textit{not}\ a_{k+1};\dots;\textit{not}\ a_l \leftarrow a_{l+1},\dots,a_m,\textit{not}\ a_{m+1},\dots,\textit{not}\ a_n$$

($0 \leq k \leq l \leq m \leq n$) where all a_i are atoms.

Example 1 $F = (p \rightarrow q) \rightarrow r$.

$$\begin{aligned}((p \rightarrow q) \rightarrow r)^* &= (((p' \rightarrow q') \wedge (p \rightarrow q)) \rightarrow r') \wedge ((p \rightarrow q) \rightarrow r)\\ &\leftrightarrow (p' \vee p \vee r') \wedge (\neg q' \vee p \vee r') \wedge (p' \vee \neg q \vee r') \wedge (\neg q' \vee \neg q \vee r')\\ &\qquad \wedge (p \vee r) \wedge (\neg q \vee r)\ .\end{aligned}$$

[3] For convenience, we will often drop "$(\mathbf{p}')$" from $F^*(\mathbf{p}')$ when there is no confusion.

Under the assumption that $(p', q', r') \leq (p, q, r)$, the formula can be simplified to

$$(p \vee r') \wedge (p' \vee \neg q \vee r') \wedge (\neg q' \vee r') \wedge (\neg q \vee r)\ .$$

Applying *Tr* to each clause yields the following formula G:

$$(\neg p \rightarrow r') \wedge (p' \vee \neg q \vee r') \wedge (q' \rightarrow r') \wedge (\neg r \rightarrow \neg q). \tag{4}$$

Thus $R(G)$ is

$$(\neg p \rightarrow r) \wedge (p \vee \neg q \vee r) \wedge (q \rightarrow r) \wedge (\neg r \rightarrow \neg q)\ . \tag{5}$$

In logic programming notation, (5) can be written as follows:

$$\begin{array}{rcl} r & \leftarrow & \textit{not}\ p \\ p\ ;\ \textit{not}\ q\ ;\ r & & \\ r & \leftarrow & q \\ \textit{not}\ q & \leftarrow & \textit{not}\ r\ . \end{array} \tag{6}$$

Proposition 2 tells us that logic program (6) is strongly equivalent to $(p \rightarrow q) \rightarrow r$.

Example 2 $F = p \rightarrow ((q \rightarrow r) \vee s)$.

$$\begin{aligned} (p \rightarrow ((q \rightarrow r) \vee s))^* &= (p' \rightarrow (((q' \rightarrow r') \wedge (q \rightarrow r)) \vee s')) \wedge (p \rightarrow (q \rightarrow r) \vee s) \\ &\leftrightarrow (\neg p' \vee (((\neg q' \vee r') \wedge (\neg q \vee r)) \vee s')) \wedge (\neg p \vee (\neg q \vee r) \vee s) \\ &\leftrightarrow (\neg p' \vee \neg q' \vee r' \vee s') \wedge (\neg p' \vee \neg q \vee r \vee s') \wedge (\neg p \vee \neg q \vee r \vee s)\ . \end{aligned}$$

Applying *Tr* to each clause yields the following formula G:

$$(p' \wedge q' \rightarrow r' \vee s') \wedge (p' \wedge \neg r \rightarrow \neg q \vee s') \wedge (\neg r \wedge \neg s \rightarrow \neg p \vee \neg q)\ . \tag{7}$$

Thus $R(G)$ is

$$(p \wedge q \rightarrow r \vee s) \wedge (p \wedge \neg r \rightarrow \neg q \vee s) \wedge (\neg r \wedge \neg s \rightarrow \neg p \vee \neg q)\ . \tag{8}$$

In logic programming notation, (8) can be written as follows:

$$\begin{array}{rcl} r\ ;\ s & \leftarrow & p,\ q \\ \textit{not}\ q\ ;\ s & \leftarrow & p,\ \textit{not}\ r \\ \textit{not}\ p\ ;\ \textit{not}\ q & \leftarrow & \textit{not}\ r,\ \textit{not}\ s\ . \end{array} \tag{9}$$

Proposition 2 tells us that logic program (9) is strongly equivalent to formula $p \rightarrow ((q \rightarrow r) \vee s)$.

3 Simpler Transformation

The following observation shows how to disregard some redundancies with the translation introduced in the previous section.

Proposition 3 *Let F be a propositional formula of signature σ. Under the assumption $\mathbf{p}' \leq \mathbf{p}$, if F^* is equivalent to $G \wedge H$ where G is a conjunction of canonical implications and H is a formula of signature σ that is entailed by $R(G)$, then F^* is equivalent to $(R(G))^*$.*

Example 1$'$. $F = (p \rightarrow q) \rightarrow r$ as in Example 1. Note that in (4), the last implication $(\neg r \rightarrow \neg q)$ is entailed by

$$R((\neg p \rightarrow r') \wedge (p' \vee \neg q \vee r') \wedge (q' \rightarrow r')).$$

Therefore, by Proposition 3, F^* is equivalent to

$$((\neg p \rightarrow r) \wedge (p \vee \neg q \vee r) \wedge (q \rightarrow r))^*.$$

In other words, in view of Proposition 1, F is strongly equivalent to the first three rules of (6).

Example 2$'$. $F = p \rightarrow ((q \rightarrow r) \vee s)$ as in Example 2. Note that in (7), the last implication is entailed by

$$R((p' \wedge q' \rightarrow r' \vee s') \wedge (p' \wedge \neg r \rightarrow \neg q \vee s'))\ .$$

Therefore in view of Proposition 3, F^* is equivalent to

$$((p \wedge q \rightarrow r \vee s) \wedge (p \wedge \neg r \rightarrow \neg q \vee s))^*.$$

In other words, in view of Proposition 1, F is strongly equivalent to the first two rules of (9).

Based on Proposition 3, we consider the following definition which leads to a simpler transformation than the one given in Proposition 2.

Definition 2. *For any formula F of signature σ, $F^\diamond(\mathbf{u})$ is defined as follows:*

- $p_i^\diamond = u_i$;
- $\bot^\diamond = \bot$;
- $(F \vee G)^\diamond = F^* \vee G^*$;
- $(F \wedge G)^\diamond = F^\diamond \wedge G^\diamond$;
- $(F \rightarrow G)^\diamond = (F^* \rightarrow G^*)$.

Note that $F^\diamond$ is different from F^* when we identify F with a conjunction $F_1 \wedge \cdots \wedge F_n$ $(n \geq 1)$, $F^\diamond$ is

$$F_1^\diamond \wedge \cdots \wedge F_n^\diamond$$

where

$$F_i^\diamond = \begin{cases} G^* \rightarrow H^* & \text{if } F_i \text{ is } G \rightarrow H, \\ F_i^* & \text{otherwise.} \end{cases}$$

The following proposition tells us that, in Proposition 2, $F^\diamond$ can be considered in place of F^*.

Proposition 4 *Given a formula F, if G is a conjunction of canonical implications that is equivalent to $F^{\diamond}$, then F and $R(G)$ are strongly equivalent.*

Example 1$''$ $F = (p \rightarrow q) \rightarrow r$ as in Example 1. Under the assumption that $(p', q', r') \leq (p, q, r)$,

$$\begin{aligned} F^{\diamond}(p', q', r') &= ((p' \rightarrow q') \wedge (p \rightarrow q)) \rightarrow r' \\ &\leftrightarrow (p \vee r') \wedge (p' \vee \neg q \vee r') \wedge (\neg q' \vee r') \\ &\leftrightarrow (\neg p \rightarrow r') \wedge (p' \vee \neg q \vee r') \wedge (q' \rightarrow r') \ . \end{aligned}$$

Thus F is strongly equivalent to

$$(\neg p \rightarrow r) \wedge (p \vee \neg q \vee r) \wedge (q \rightarrow r) \ ,$$

which is the same as in Example 1$'$.

Example 2$''$ $F = p \rightarrow ((q \rightarrow r) \vee s)$ as in Example 2. Under the assumption that $(p', q', r', s') \leq (p, q, r, s)$,

$$\begin{aligned} F^{\diamond}(p', q', r', s) &= p' \rightarrow (((q' \rightarrow r') \wedge (q \rightarrow r)) \vee s') \\ &\leftrightarrow (\neg p' \vee \neg q' \vee r' \vee s') \wedge (\neg p' \vee \neg q \vee r \vee s') \\ &\leftrightarrow (p' \wedge q' \rightarrow r' \vee s') \wedge (p' \wedge \neg r \rightarrow \neg q \vee s') \ . \end{aligned}$$

Thus F is strongly equivalent to

$$(p \wedge q \rightarrow r \vee s) \wedge (p \wedge \neg r \rightarrow \neg q \vee s) \ ,$$

which is the same as in Example 2$'$.

Due to lack of space, we do not provide a detailed comparison between our translation method and the others. However, we note that Proposition 2 not only shows that the reduction is possible, but also tells us how to generate strongly equivalent logic programs of preferably smaller size, based on the notion of equivalence in classical logic. This is in contrast with the other approaches that are based on syntactic rewriting rules under the logic of here-and-there. For instance, given a formula

$$((p \rightarrow q) \rightarrow r) \rightarrow r$$

our translation yields the following program:

$$\begin{aligned} q\ ;\ r\ ;\ \textit{not}\ r &\leftarrow p \\ \textit{not}\ p &\leftarrow \textit{not}\ q \ . \end{aligned}$$

On the other hand, the following program is obtained according to Section 3 of [5].

$$\begin{aligned} \textit{not}\ p\ ;\ r &\leftarrow \textit{not}\ q \\ r &\leftarrow r \\ q\ ;\ r\ ;\ \textit{not}\ r &\leftarrow p \\ \textit{not}\ p\ ;\ r\ ;\ \textit{not}\ r &\leftarrow \textit{not}\ q \ . \end{aligned}$$

However, clearly, any translation according to Proposition 4 (or Proposition 2) involves an exponential blowup in size in the worst case. Indeed, it is shown in [5] that there is no polynomial translation from propositional theories to logic programs if we do not introduce new atoms, and that there is one if we allow them.

4 Prenex Normal Form of First-Order Formulas

The translation from an arbitrary propositional theory into a logic program shows that their syntactic difference is not essential, which allows existing answer set solvers to compute the stable models of arbitrary propositional formulas. Can the result be extended to first-order formulas, of which the stable model semantics is presented in [8]?

We begin with a review of the stable model semantics presented in [8], which extends the definition of a stable model reviewed in Section 2 to first-order sentences. Given a first-order sentence F, by SM$[F]$ we denote the second-order sentence

$$F \wedge \forall \mathbf{u}((\mathbf{u} < \mathbf{p}) \rightarrow \neg F^*(\mathbf{u})),$$

where **p** stands for the tuple of all predicate constants $p_1, \dots, p_n$ occurring in F, **u** is a tuple of n distinct predicate variables $u_1, \dots, u_n$, equation $\mathbf{u} < \mathbf{p}$ is defined as in circumscription [11], and $F^*(\mathbf{u})$ is defined recursively, as follows:

- $p_i(t_1, \dots, t_m)^* = u_i(t_1, \dots, t_m)$;
- $(t_1 = t_2)^* = (t_1 = t_2)$;
- $\bot^* = \bot$;
- $(F \odot G)^* = F^* \odot G^*$, where $\odot \in \{\wedge, \vee\}$;
- $(F \rightarrow G)^* = (F^* \rightarrow G^*) \wedge (F \rightarrow G)$;
- $(QxF)^* = QxF^*$, where $Q \in \{\forall, \exists\}$.

A model of F is *stable* if it satisfies SM$[F]$. For the definition of strong equivalence extended to first-order formulas, we refer the reader to Section 4 of [8].

Proposition 1 can be extended to the case where F and G are first-order formulas [8, Proposition 5]. Using the proposition, one can prove that every first-order formula is strongly equivalent to a prenex normal form. The following proposition is essentially Theorem 6.4 of [14].

Proposition 5 *Every first-order formula is strongly equivalent to a prenex normal form.*

The proposition follows from the fact that usual prenex normal form conversion rules for first-order logic (e.g., [15, Lemma 2.29]) preserves strong equivalence. Alternative to the proof in [14], this fact can be proved using [8, Proposition 5]. For instance, $\forall x F(x) \rightarrow G$ is strongly equivalent to $\exists x(F(x) \rightarrow G)$, where x is not free in G. Consider

$$\begin{aligned}(\forall x F(x) \rightarrow G)^* &= (\forall x F(x) \rightarrow G) \wedge (\forall x F^*(x) \rightarrow G^*) \\ &\Leftrightarrow \exists x(F(x) \rightarrow G) \wedge \exists x(F^*(x) \rightarrow G^*) \end{aligned} \qquad (10)$$

and

$$(\exists x(F(x) \rightarrow G))^* = \exists x((F(x) \rightarrow G) \wedge (F^*(x) \rightarrow G^*)) \,. \qquad (11)$$

Note that (10) and (11) are not (classically) equivalent in general, but they are equivalent under the assumption $\mathbf{p}' \leq \mathbf{p}$, where $\mathbf{p}$ is the tuple of all predicate constants occurring in $F(x)$ and G, and $\mathbf{p}'$ is the tuple of new, pairwise distinct predicate constants of the same length as $\mathbf{p}$. Therefore, by [8, Proposition 5], we conclude that $\forall x F(x) \rightarrow G$ is strongly equivalent to $\exists x(F(x) \rightarrow G)$.

Also, Proposition 2 can be straightforwardly extended to quantifier-free first-order formulas as follows. A first-order formula F is in *negation normal form* if, for every subformula $G \rightarrow H$ of F,

– formula G is an atomic formula, and
– formula H is $\bot$.

For any clause C in a CNF of a quantifier-free first order formula, $Tr(C)$ from Section 2 can be extended in a straightforward way. The equality can be placed either in the consequent or the antecedent (properly negated).

Corollary 1 *Any first-order formula is strongly equivalent to a prenex normal form whose matrix is a conjunction of implications $F \rightarrow G$ where F and G are formulas in negation normal form.*

The matrix of a prenex normal form indicated in Corollary 1 is in the form of a logic program. Thus, similar to the propositional case, the syntactic difference of arbitrarily nested connectives and quantifiers is not essential between the new language proposed in [8] and logic programs. On the other hand, since the prenex normal form may contain existential quantifiers, it is different from a logic program, where all variables are assumed to be universally quantified. For instance, according to [8], the stable models of formula $\exists x\ p(x)$ represent that p is a singleton, as in circumscription. This has no counterpart in logic programs, since their stable models are limited to Herbrand interpretations. For a related discussion, see [16].

5 Implementation

Our implementation, which we call F2LP, turns an arbitrary propositional theory into a logic program and calls DLV to compute its stable models. When the input is already in the syntax of DLV input language, its operation is just as what DLV does. The system is available at

`http://peace.eas.asu.edu/f2lp .`

The ASCII representations of propositional connectives used in the syntax of F2LP are summarized in the following chart:

Symbol	$\neg$	$\wedge$	$\vee$	$\rightarrow$	$\bot$	$\top$
ASCII representation	`not`	`&`	`|`	`->`	`false`	`true`

Example 1 is written in the syntax of F2LP as follows:

```
(p->q)->r.
```

F2LP turns this formula into the following DLV input:

```
r :- not p.
p | r | q_bar :-.
r :- q.
q_bar :- not q.
:- q, q_bar.
```

Note that this program is slightly different from the logic program shown in Example 1′ (the first three rules of (6)). This is because DLV, like most other answer set solvers, does not allow negation as failure in the head of a rule. However, it can be simulated by introducing new atoms (Section 4 of [17]). The method replaces the occurrence of *not* p in the head of a rule with a new atom $\overline{p}$, and adds rules $\overline{p} \leftarrow$ *not* p and $\leftarrow p, \overline{p}$. The stable models of the program correspond to the stable models of the original program by disregarding the presence of the new atoms. In the example above, `q_bar` is a new atom, and the last two rules are added. After F2LP calls DLV to compute the stable models, it removes all occurrences of the new atoms ("`_bar`") from the stable models returned by DLV.

Example 2 is written in our syntax as follows:

```
p -> ((q->r) | s).
```

This is turned into the following DLV input by F2LP:

```
r | s :- p, q.
q_bar | s :- p, not r.
q_bar :- not q.
:- q, q_bar.
```

6 Conclusion

Our contributions in this note are as follows. First, we presented a new proof of the theorem on strong equivalence between propositional theories and logic programs. Unlike the other approaches that are based on the logic of here-and-there, our proof relies on familiar properties of classical logic. Due to this fact, our proof indicates how corresponding logic programs can be generated using equivalent transformations in classical logic. Second, using the same reduction idea, we showed that arbitrary first-order formulas under the stable model semantics, recently proposed in [8], can be turned into a prenex normal form whose matrix has the form of a logic program. Third, we presented a prototype implementation for computing the stable models of arbitrary propositional formulas based on the reduction method.

For future work, we plan to investigate how the methods of obtaining minimally equivalent theories in classical logic can be applied to finding minimally equivalent logic programs. Recently, Cabalar *et al.* [18] proposed two notions of minimal logic programs. It would be interesting to see how these approaches are related.

Acknowledgements

We are grateful to Paolo Ferraris, Vladimir Lifschitz and the anonymous referees for useful comments on this paper. The authors were partially supported by DTO AQUAINT.

A Appendix: Proof of Proposition 2

Due to lack of space, we present the proof of Proposition 2 only, which follows immediately from Proposition 1 and the following proposition.

Proposition 6 *Let F be a formula of signature σ and G a conjunction of canonical implications that is equivalent to F^*. Then*

$$\mathbf{p}' \leq \mathbf{p} \rightarrow (F^* \leftrightarrow (R(G))^*)$$

is a tautology.

The proof of Proposition 6 uses the following lemmas, most of which can be proven by induction.

Lemma 1. *For any formula F of signature σ, the formula*

$$\mathbf{p}' \leq \mathbf{p} \rightarrow (F^*(\mathbf{p}') \rightarrow F)$$

is logically valid.

Lemma 2. *Every formula F is equivalent to $R(F^*)$.*

Lemma 3. *For any two formulas F and G of signature $\sigma \cup \sigma'$,*

$$(F \leftrightarrow G) \rightarrow (R(F) \leftrightarrow R(G))$$

is a tautology.

Proof. Assume that $F \leftrightarrow G$ holds for all interpretations of $\sigma \cup \sigma'$, which includes the interpretations I such that $p^I = (p')^I$ for all $p \in \mathbf{p}$. It is clear that $F^I = R(F)^I$ and $G^I = R(G)^I$, from which $R(F)^I = R(G)^I$ follows. Since I range over all interpretations of σ, it follows that $R(F) \leftrightarrow R(G)$. ∎

Lemma 4. *For any canonical implication F of signature $\sigma \cup \sigma'$,*

$$(\mathbf{p}' \leq \mathbf{p}) \rightarrow ((F \wedge R(F)) \leftrightarrow (R(F))^*)$$

is a tautology.

Proof of Proposition 6. Assume $\mathbf{p}' \leq \mathbf{p}$ and $F^* \leftrightarrow G$. By Lemma 1, $F^* \rightarrow F$ holds, so that F^* is equivalent to $G \wedge F$. Since F is equivalent to $R(F^*)$ according to Lemma 2, $G \wedge F$ is equivalent to $G \wedge R(F^*)$, which, in turn, is equivalent to $G \wedge R(G)$ according to Lemma 3. By Lemma 4, it follows that $G \wedge R(G)$ is equivalent to $(R(G))^*$. ∎

References

1. Pearce, D.: A new logical characterization of stable models and answer sets. In Dix, J., Pereira, L., Przymusinski, T., eds.: Non-Monotonic Extensions of Logic Programming (Lecture Notes in Artificial Intelligence 1216), Springer-Verlag (1997) 57–70
2. Ferraris, P.: Answer sets for propositional theories. In: Proceedings of International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR). (2005) 119–131
3. Cabalar, P., Ferraris, P.: Propositional theories are strongly equivalent to logic programs. Submitted for publication (2005)
4. Lifschitz, V., Pearce, D., Valverde, A.n.: Strongly equivalent logic programs. ACM Transactions on Computational Logic **2** (2001) 526–541
5. Cabalar, P., Pearce, D., Valverde, A.n.: Reducing propositional theoreis in equilibrium logic to logic programs. In: Proceedings of 12th Portuguese Conference on Artificial Intelligence (EPIA 2005). (2005) 4–17
6. Lifschitz, V., Tang, L.R., Turner, H.: Nested expressions in logic programs. Annals of Mathematics and Artificial Intelligence **25** (1999) 369–389
7. Sarsakov, V., Schaub, T., Tompits, H., Woltran, S.: nlp: A compiler for nested logic programming. In Lifschitz, V., Niemelä, I., eds.: Proceedings of the Seventh International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'04). Volume 2923 of Lecture Notes in Computer Science., Springer-Verlag Heidelberg (2003) 361 – 364
8. Ferraris, P., Lee, J., Lifschitz, V.: A new perspective on stable models. In: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI). (2007)
9. Pearce, D., Tompits, H., Woltran, S.: Encodings for equilibrium logic and logic programs with nested expressions. In: Proceedings of Portuguese Conference on Artificial Intelligence (EPIA). (2001) 306–320
10. McCarthy, J.: Circumscription—a form of non-monotonic reasoning. Artificial Intelligence **13** (1980) 27–39,171–172
11. Lifschitz, V.: Circumscription. In Gabbay, D., Hogger, C., Robinson, J., eds.: The Handbook of Logic in AI and Logic Programming. Volume 3. Oxford University Press (1994) 298–352
12. Ferraris, P., Lifschitz, V.: Mathematical foundations of answer set programming. In: We Will Show Them! Essays in Honour of Dov Gabbay. King's College Publications (2005) 615–664
13. Lin, F.: Reducing strong equivalence of logic programs to entailment in classical propositional logic. In: Proceedings of International Conference on Principles of Knowledge Representation and Reasoning (KR). (2002) 170–176
14. Pearce, D., n Valverde, A.: A first order nonmonotonic extension of constructive logic. Studia Logica **80** (2005) 323–348
15. Mendelson, E.: Introduction to Mathematical Logic. Wadsworth & Brooks (1987) Third edition.
16. Texas Action Group: Technical discussions: Do we need existential quantifiers in logic programming? (2007)
`http://www.cs.utexas.edu/users/vl/tag/discussions.html` .
17. Janhunen, T.: On the effect of default negation on the expressiveness of disjunctive rules. In: Proc. LPNMR 2001. (2001) 93–106
18. Cabalar, P., Pearce, D., Valverde, A.n.: Minimal logic programs. Unpublished draft (2007)

A Common View on Strong, Uniform, and Other Notions of Equivalence in Answer-Set Programming[⋆]

Stefan Woltran

Institut für Informationssysteme 184/3, Technische Universität Wien,
Favoritenstraße 9-11, A-1040 Vienna, Austria
e-mail: stefan@kr.tuwien.ac.at

Abstract. Logic programming under the answer-set semantics nowadays deals with numerous different notions of equivalence between programs. This is due to the fact that equivalence for substitution (known as strong equivalence), which holds between programs P and Q iff P can faithfully be replaced by Q within any context R, is a different concept than ordinary equivalence between P and Q, which holds if P and Q have the same answer sets. Notions inbetween strong and ordinary equivalence have therefore been obtained by either restricting the syntactic structure of R or bounding the set of atoms allowed to occur in R (relativized equivalence). For the former approach, however, it turned out that any "reasonable" syntactic restriction to R either coincides with strong equivalence or collapses to uniform equivalence where R ranges over arbitrary sets of facts. In this paper, we propose a parameterization for equivalence notions which takes care of both such kinds of restrictions simultaneously by bounding, on the one hand, the atoms which are allowed to occur in the rule heads of R and, on the other hand, the atoms which are allowed to occur in the rule bodies of R. We introduce a semantical characterization including known ones as SE-models or UE-models as special cases. Moreover, we provide complexity bounds for the problem in question.

1 Introduction

Starting with the seminal paper on strong equivalence between logic programs by Lifschitz, Pearce, and Valverde [7], a new research direction in logic programming under the answer-set semantics has been established. This is due to fact that strong equivalence between programs P and Q, which holds iff P can faithfully be replaced by Q in any program, is a different concept than deciding whether P and Q have the same answer sets, i.e., (ordinary) equivalence between P and Q holds. Formally, P and Q are strongly equivalent iff, for each further so-called context program R, $P \cup R$ and $Q \cup R$ possess the same answer sets. That difference between strong and ordinary equivalence motivated investigations of equivalence notions inbetween (see, e.g., [4]). Basically this was done in two ways, viz. to bound the actually allowed context programs R by (i) restricting their syntax; or (ii) restricting their language. For Case (i), it turned out that any "reasonable" (i.e., where the restriction is defined rule-wise, for instance only allowing

[⋆] Supported by the Austrian Science Fund (FWF) under project P15068-INF.

for Horn rules) attempt either coincides with strong equivalence itself, or reduces to uniform equivalence [2], which is to test whether, for each set F of facts, $P \cup F$ and $Q \cup F$ possess the same answer sets. Case (ii), where the atoms allowed to occur in R are given by an alphabet $\mathcal{A}$ yields in general different concepts for different $\mathcal{A}$ and thus is known as strong equivalence relative to $\mathcal{A}$ [12]. Finally a combination of both approaches leads to the concept of uniform equivalence relative to $\mathcal{A}$ [12].[1]

In this paper, we propose a fine-grained framework to define notions of equivalence where the aforementioned restrictions are simultaneously taken into account. This is accomplished by restricting, on the one hand, the atoms which are allowed to occur in the rule heads of the context programs and, on the other hand, the atoms which are allowed to occur in the rule bodies of the context programs. More formally, for given programs P, Q, and given sets $\mathcal{H}$, $\mathcal{B}$ of atoms, we want to decide whether the answer sets of $P \cup R$ and $Q \cup R$ coincide for each program R, where each rule in R has its head atoms from $\mathcal{H}$ and its body atoms from $\mathcal{B}$. We will show that this new notion includes all of the previously mentioned; for instance, setting $\mathcal{B} = \emptyset$, i.e., disallowing any atom to occur in bodies, will be shown to coincide with (relativized) uniform equivalence; while the parameterization $\mathcal{H} = \mathcal{B}$ amounts to (relativized) strong equivalence by definition.

The main contribution of the paper is to provide a general semantical characterization for the new equivalence notion. Moreover, we show that our characterization includes as special cases known concepts as SE-models [11] or UE-models [2]. Finally, we address the computational complexity of the introduced equivalence problems and propose a prototypical implementation.

2 Background

Throughout the paper we assume an arbitrary finite but fixed universe $\mathcal{U}$ of atoms. Subsets of $\mathcal{U}$ are either called interpretations or alphabets: We use the latter term to restrict the syntax of programs, while the former is used when talking about semantics. For an interpretation Y and an alphabet $\mathcal{A}$, we write $Y|_{\mathcal{A}}$ instead of $Y \cap \mathcal{A}$.

A propositional disjunctive logic program (or simply, a program) is a finite set of rules of form

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

$n > 0$, $n \geq m \geq l$, and where all a_i are propositional atoms in $\mathcal{U}$ and not denotes default negation; for $n = l = 1$, we usually identify the rule (1) with the atom a_1, and call it a *fact*. A rule of the form (1) is called a *constraint* if $l = 0$, *positive* if $m = n$ and *unary* if it is either a fact or of the form $a \leftarrow b$. A program is positive (resp., unary) iff all its rules are positive (resp., unary). If all atoms occurring in a program P are from a given alphabet $\mathcal{A} \subseteq \mathcal{U}$ of atoms, we say that P is a program *over* (alphabet) $\mathcal{A}$. The class of all logic programs over universe $\mathcal{U}$ is denoted by $\mathcal{C}_{\mathcal{U}}$.

For a rule r of form (1), we identify its head by $H(r) = \{a_1, \ldots, a_l\}$ and its body via $B^+(r) = \{a_{l+1}, \ldots, a_m\}$ and $B^-(r) = \{a_{m+1}, \ldots, a_n\}$. We shall write rules of

[1] A further approach is to additionally restrict the alphabet over which the answer sets of $P \cup R$ and $Q \cup R$ compared. This kind of *projection* was investigated in [5, 8, 10], but we do not consider it in this work.

form (1) also as $H(r) \leftarrow B^+(r), not\, B^-(r)$. Moreover, we also use $B(r) = B^+(r) \cup B^-(r)$. Finally, for a program P, $\alpha(P) = \bigcup_{r \in P} \alpha(r)$, for $\alpha \in \{H, B, B^+, B^-\}$.

The relation $Y \models P$ between an interpretation Y and a program P is defined as usual, i.e., $Y \models P$ holds if for each $r \in P$, $Y \models r$. The latter holds iff $H(r) \cap Y \neq \emptyset$, whenever jointly $B^+(r) \subseteq Y$ and $B^-(r) \cap Y = \emptyset$ hold. If $Y \models P$ holds, Y is called a model of P. Following Gelfond and Lifschitz [6], an interpretation Y, is an *answer set* of a program P iff it is a minimal (wrt set inclusion) model of the *reduct* $P^Y = \{H(r) \leftarrow B^+(r) \mid Y \cap B^-(r) = \emptyset\}$. The set of all answer sets of a program P is denoted by $\mathcal{AS}(P)$.

Finally, we briefly review some prominent notions of equivalence [7, 2, 12, 4], which have been studied under the answer-set semantics: For a given alphabet $\mathcal{A} \subseteq \mathcal{U}$, we call programs $P, Q \in \mathcal{C}_\mathcal{U}$, *strongly equivalent relative to* $\mathcal{A}$, iff, for any program R over $\mathcal{A}$, it holds that $\mathcal{AS}(P \cup R) = \mathcal{AS}(Q \cup R)$. P, Q are *uniformly equivalent relative to* $\mathcal{A}$, iff, for any set $F \subseteq \mathcal{A}$ of facts, $\mathcal{AS}(P \cup F) = \mathcal{AS}(Q \cup F)$. If, $\mathcal{A} = \mathcal{U}$, strong (resp., uniform) equivalence relative to $\mathcal{A}$ collapses to (unrelativized) strong (resp., uniform) equivalence [7, 2]; if $\mathcal{A} = \emptyset$, we obtain *ordinary equivalence*, i.e., $\mathcal{AS}(P) = \mathcal{AS}(Q)$.

In case of strong equivalence (also in the relativized case), it was shown that the syntactic class of *counterexamples*, i.e., programs R, such that $\mathcal{AS}(P \cup R) \neq \mathcal{AS}(Q \cup R)$, can always be restricted to the class of unary programs. Hence, the next result comes by mere surprise, but provides insight with respect to the alphabets in the rules' heads and bodies.

Lemma 1. *Let P, Q, $R \in \mathcal{C}_\mathcal{U}$ be programs, and Y be an interpretation, such that $Y \in \mathcal{AS}(P \cup R)$ and $Y \notin \mathcal{AS}(Q \cup R)$. Then there exists a program R', such that R' is positive, $H(R') \subseteq H(R)$, $B(R') \subseteq B(R)$, $Y \in \mathcal{AS}(P \cup R')$, and $Y \notin \mathcal{AS}(Q \cup R')$.*

The result can be checked by using $R' = R^Y$.

As we will see later, Lemma 1 can even be strengthened to unary programs. However, already the present result shows that whenever a counterexample R for an equivalence problem exists, then we can find a simpler (positive) one, which is given over the same alphabets in the heads, and respectively, bodies.

3 The General Framework

Lemma 1 suggests to study equivalence problems along a parameterization via two alphabets. To this end, we first introduce classes of programs as follows.

Definition 1. *For any alphabets $\mathcal{H}, \mathcal{B} \subseteq \mathcal{U}$, the class $\mathcal{C}_{\langle \mathcal{H}, \mathcal{B} \rangle}$ of programs is defined as $\{P \in \mathcal{C}_\mathcal{U} \mid H(P) \subseteq \mathcal{H}, B(P) \subseteq \mathcal{B}\}$.*

With this concept of program classes at hand, we now define equivalence notions which are more fine-grained than the ones previously introduced.

Definition 2. *Let $\mathcal{H}, \mathcal{B} \subseteq \mathcal{U}$ be alphabets, and $P, Q \in \mathcal{C}_\mathcal{U}$ be programs. The $\langle \mathcal{H}, \mathcal{B} \rangle$-*equivalence problem *between P and Q, in symbols $P \equiv_{\langle \mathcal{H}, \mathcal{B} \rangle} Q$, is to decide whether, for each $R \in \mathcal{C}_{\langle \mathcal{H}, \mathcal{B} \rangle}$, $\mathcal{AS}(P \cup R) = \mathcal{AS}(Q \cup R)$. If $P \equiv_{\langle \mathcal{H}, \mathcal{B} \rangle} Q$ holds, we say that P and Q are $\langle \mathcal{H}, \mathcal{B} \rangle$-*equivalent.

The class $\mathcal{C}_{\langle\mathcal{H},\mathcal{B}\rangle}$ is also called the *context* of an $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence problem, and a program $R \in \mathcal{C}_{\langle\mathcal{H},\mathcal{B}\rangle}$, where $\mathcal{AS}(P \cup R) \neq \mathcal{AS}(Q \cup R)$ holds, is called a *counterexample* to the $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence problem between P and Q.

Example 1. Consider $P = \{a \vee b \leftarrow;\ a \leftarrow b\}$ and $Q = \{a \leftarrow \mathit{not}\, b;\ b \leftarrow \mathit{not}\, a;\ a \leftarrow b\}$. It is known that these programs are not strongly equivalent, since adding any R which closes the cycle between a and b yields $\mathcal{AS}(P \cup R) \neq \mathcal{AS}(Q \cup R)$. In particular, for $R = \{b \leftarrow a\}$, we get $\mathcal{AS}(P \cup R) = \{\{a,b\}\}$, while $\mathcal{AS}(Q \cup R) = \emptyset$. However, P and Q are uniformly equivalent. In our setting, we are able to "approximate" equivalence notions which hold between P and Q. It can be shown that, for instance, $P \equiv_{\langle\{a,b\},\{b\}\rangle} Q$ or $P \equiv_{\langle\{a\},\{a,b\}\rangle} Q$ holds (basically since $b \leftarrow a$ does not occur in any program in $\mathcal{C}_{\langle\{a,b\},\{b\}\rangle}$, or $\mathcal{C}_{\langle\{a\},\{a,b\}\rangle}$). But $P \equiv_{\langle\{b\},\{a,b\}\rangle} Q$ and likewise $P \equiv_{\langle\{a,b\},\{a\}\rangle} Q$ do not hold, since $\{b \leftarrow a\}$ is contained in the context $\mathcal{C}_{\langle\{b\},\{a,b\}\rangle}$, resp., $\mathcal{C}_{\langle\{a,b\},\{a\}\rangle}$. $\diamond$

Observe that the concept of $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence captures other equivalence notions as follows: $\langle\mathcal{A},\mathcal{A}\rangle$-equivalence coincides with strong equivalence relative to $\mathcal{A}$; and, in particular, $\langle\mathcal{U},\mathcal{U}\rangle$-equivalence amounts to strong equivalence. Later we will see that $\langle\mathcal{A},\emptyset\rangle$-equivalence coincides with uniform equivalence relative to $\mathcal{A}$; and, in particular, $\langle\mathcal{U},\emptyset\rangle$-equivalence amounts to uniform equivalence. Note that the relation to uniform equivalence is not immediate since $\langle\mathcal{A},\emptyset\rangle$-equivalence deals with sets of *disjunctive* facts, i.e., rules of the form $a_1 \vee \cdots \vee a_l \leftarrow$, rather than sets of (simple) facts $a \leftarrow$.

The following result shows some general properties for $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence.

Proposition 1. *Let $\mathcal{H},\mathcal{B} \subseteq \mathcal{U}$ and $P,Q \in \mathcal{C}_\mathcal{U}$, such that $P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ holds. Then, also $(P \cup R) \equiv_{\langle\mathcal{H}',\mathcal{B}'\rangle} (Q \cup R)$ holds, for each $R \in \mathcal{C}_{\langle\mathcal{H},\mathcal{B}\rangle}$, $\mathcal{H}' \subseteq \mathcal{H}$, and $\mathcal{B}' \subseteq \mathcal{B}$.*

A central aspect in equivalence checking is the quest for semantical characterizations assigned to a *single* program. The following formal approach captures this aim.

Definition 3. *A* semantical characterization *for an $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence problem is a function $\sigma_{\langle\mathcal{H},\mathcal{B}\rangle} : \mathcal{C}_\mathcal{U} \to 2^{2^\mathcal{U} \times 2^\mathcal{U}}$, such that, for any $P,Q \in \mathcal{C}_\mathcal{U}$, $P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ holds iff $\sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P) = \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$.*

We will review known characterizations for special cases (as, for instance, SE-models [11] and UE-models [2]) later. Finally, we also introduce containment problems.

Definition 4. *Let $\mathcal{H},\mathcal{B} \subseteq \mathcal{U}$ be alphabets, and $P,Q \in \mathcal{C}_\mathcal{U}$ be programs. The* $\langle\mathcal{H},\mathcal{B}\rangle$-containment problem *for P in Q, in symbols $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$, is to decide whether, for each $R \in \mathcal{C}_{\langle\mathcal{H},\mathcal{B}\rangle}$, $\mathcal{AS}(P \cup R) \subseteq \mathcal{AS}(Q \cup R)$. A counterexample to $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$, is any program $R \in \mathcal{C}_{\langle\mathcal{H},\mathcal{B}\rangle}$, such that $\mathcal{AS}(P \cup R) \not\subseteq \mathcal{AS}(Q \cup R)$.*

Proposition 2. *$P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ holds iff $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ and $Q \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} P$ jointly hold.*

4 Characterizations for $\langle\mathcal{H},\mathcal{B}\rangle$-Equivalence

Towards the semantical characterization for $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence problems, we first introduce the notion of a witness, which is assigned to $\langle\mathcal{H},\mathcal{B}\rangle$-containment problems taking both compared programs into account. Afterwards, we will derive the desired semantical characterization of $\langle \mathcal{H}, \mathcal{B} \rangle$-models which are assigned to single programs and satisfy the conditions in Definition 3.

To start with, we introduce the following partial order on interpretations and state a technical lemma.

Definition 5. *Given alphabets $\mathcal{H}, \mathcal{B} \subseteq \mathcal{U}$, we define the relation $\preceq^{\mathcal{B}}_{\mathcal{H}} \subseteq \mathcal{U} \times \mathcal{U}$ between interpretations as follows: $V \preceq^{\mathcal{B}}_{\mathcal{H}} Z$ iff $V|_{\mathcal{H}} \subseteq Z|_{\mathcal{H}}$ and $Z|_{\mathcal{B}} \subseteq V|_{\mathcal{B}}$.*

Observe that if $V \preceq^{\mathcal{B}}_{\mathcal{H}} Z$ holds, then either $V|_{\mathcal{H} \cup B} = Z|_{\mathcal{H} \cup B}$, or one of $V|_{\mathcal{H}} \subset Z|_{\mathcal{H}}$, $Z|_{\mathcal{B}} \subset V|_{\mathcal{B}}$ holds. We write $V \prec^{\mathcal{B}}_{\mathcal{H}} Z$, in case $V \preceq^{\mathcal{B}}_{\mathcal{H}} Z$ and $V|_{\mathcal{H} \cup B} \neq Z|_{\mathcal{H} \cup B}$.

Lemma 2. *Let $\mathcal{H}, \mathcal{B} \subseteq \mathcal{U}$ be alphabets, P a positive program with $H(P) \subseteq \mathcal{H}$, $B(P) \subseteq \mathcal{B}$, and $Z, V \subseteq \mathcal{U}$ interpretations. Then, $V \models P$ and $V \preceq^{\mathcal{B}}_{\mathcal{H}} Z$ imply $Z \models P$.*

Proof. Towards a contradiction, suppose $V \models P$, $V|_{\mathcal{H}} \subseteq Z|_{\mathcal{H}}$, $Z|_{\mathcal{B}} \subseteq V|_{\mathcal{B}}$, as well as $Z \not\models P$ hold. If $Z \not\models P$, then there exists a rule $r \in P$, such that $B^+(r) \subseteq Z$ and $Z \cap H(r) = \emptyset$. Since $H(r) \subseteq \mathcal{H}$, we get from $V|_{\mathcal{H}} \subseteq Z|_{\mathcal{H}}$, that $V \cap H(r) = \emptyset$. Moreover, since $B^+(r) \subseteq \mathcal{B}$, we have $B^+(r) \subseteq Z|_{\mathcal{B}} \subseteq V|_{\mathcal{B}}$, and thus $B^+(r) \subseteq V$. Hence $V \not\models r$ which yields $V \not\models P$. Contradiction. □

4.1 Witnesses for Containment Problems

Definition 6. *A* witness *for (violating) a containment problem $P \subseteq_{\langle \mathcal{H}, \mathcal{B} \rangle} Q$ is a pair of interpretations (X, Y) with $X \subseteq Y \subseteq \mathcal{U}$, such that*

(i) $Y \models P$ and for each $Y' \subset Y$, $Y' \models P^Y$ implies $Y'|_{\mathcal{H}} \subset Y|_{\mathcal{H}}$;
(ii) if $Y \models Q$ then $X \subset Y$, $X \models Q^Y$, and for each X' with $X \preceq^{\mathcal{B}}_{\mathcal{H}} X' \subset Y$, $X' \not\models P^Y$.

The aim of a witness (X, Y) for (violating) $P \subseteq_{\langle \mathcal{H}, B \rangle} Q$ is, roughly speaking, as follows: Set X is used to characterize a counterexample R, such that set Y behaves as a witnessing answer set, i.e., $Y \in \mathcal{AS}(P \cup R)$ and $Y \notin \mathcal{AS}(Q \cup R)$. Property (i) ensures that Y can become such an answer set of an extended P. To this end, it is not only necessary that $Y \models P$. It also has to be guaranteed that no $Y' \subset Y$, with $Y'|_{\mathcal{H}} = Y|_{\mathcal{H}}$ satisfies $Y' \models P^Y$, otherwise Y can never become an answer of $P \cup R$, no matter which $R \in \mathcal{C}_{\langle \mathcal{H}, \mathcal{B} \rangle}$ is added to P. Property (ii) ensures that the program R is obtained from X in such a way, that Y does not become an answer set of $Q \cup R$, but Y still can become an answer set of $P \cup R$. We can focus on a positive program R (cf. Lemma 1), and R can be constructed in such a way, that it rules out all possible models $X' \subset Y$ of P^Y, as long as $X \not\preceq^{\mathcal{B}}_{\mathcal{H}} X'$ holds. The latter is due to the fact that each positive $R \in \mathcal{C}_{\langle \mathcal{H}, \mathcal{B} \rangle}$ suitably applies here to Lemma 2.

Example 2. We already have mentioned that $P = \{a \vee b \leftarrow;\ a \leftarrow b\}$ and $Q = \{a \leftarrow not\, b;\ b \leftarrow not\, a;\ a \leftarrow b\}$ are not $\langle \mathcal{H}, \mathcal{B} \rangle$-equivalent for $\mathcal{H} = \{b\}$ and $\mathcal{B} = \{a, b\}$. We show that there exists a witness for $P \subseteq_{\langle \mathcal{H}, \mathcal{B} \rangle} Q$. First, let us compute the programs' models (over $\{a, b\}$) as well as the models of their reducts. Observe that P and Q have the same models $Y_1 = \{a, b\}$ and $Y_2 = \{a\}$. For the positive program P we are done, since all reducts coincide with P, and thus possess the same models. For Q, however, observe that $Q^{Y_1} = \{a \leftarrow b\}$ has models $\emptyset$, $\{a\}$, and $\{a,b\}$, while $Q^{Y_2} = \{a;\ a \leftarrow b\}$ has models $\{a\}$ and $\{a,b\}$. We show that for $X = \emptyset$, (X, Y_1) is a witness for $P \subseteq_{\langle \mathcal{H},\mathcal{B}\rangle} Q$. Clearly, Condition (i) from Definition 6 holds, since $Y_1 \models P$ and $Y_2|_{\mathcal{H}} \subset Y_1|_{\mathcal{H}}$. Concerning Condition (ii), we have $Y_1 \models Q$, $X \subset Y_1$, and $X \models Q^{Y_1}$. The only X' (over $\{a,b\}$) such that $X \preceq^{\mathcal{B}}_{\mathcal{H}} X'$ holds is X itself, since $\mathcal{B} = \{a,b\}$ and thus $X' \subseteq X$ has to be satisfied. It thus remains to check $X \not\models P^{Y_1}$, which is the case. Hence, $(\emptyset, \{a,b\})$ is a witness for $P \subseteq_{\langle \{b\},\{a,b\}\rangle} Q$. By similar arguments (in particular, since also $\{b\} \not\models P^{Y_1}$), $(\emptyset, \{a,b\})$ is a witness also for $P \subseteq_{\langle \{a,b\},\{a\}\rangle} Q$. $\diamond$

We now formally proof that the existence of witnesses for a containment problem $P \subseteq_{\langle \mathcal{H},\mathcal{B}\rangle} Q$ in fact shows that $P \subseteq_{\langle \mathcal{H},\mathcal{B}\rangle} Q$ does not hold. As a by-product we obtain that there are always counterexamples to $P \subseteq_{\langle \mathcal{H},\mathcal{B}\rangle} Q$ of a simple syntactic form.

Lemma 3. *The following propositions are equivalent for any $P, Q \in \mathcal{C}_{\mathcal{U}}$, $\mathcal{H}, \mathcal{B} \subseteq \mathcal{U}$:*

(1) $P \subseteq_{\langle \mathcal{H},\mathcal{B}\rangle} Q$ does not hold;
(2) there exists a unary program $R \in \mathcal{C}_{\langle \mathcal{H},\mathcal{B}\rangle}$, such that $\mathcal{AS}(P \cup R) \not\subseteq \mathcal{AS}(Q \cup R)$;
(3) there exists a witness for $P \subseteq_{\langle \mathcal{H},\mathcal{B}\rangle} Q$.

Proof. We show that (1) implies (3) and (3) implies (2). (2) implies (1) obviously holds by definition of $\langle \mathcal{H},\mathcal{B}\rangle$-containment problems.

(1) implies (3): If $P \subseteq_{\langle \mathcal{H},\mathcal{B}\rangle} Q$ does not hold, there exists a program R, and an interpretation Y, such that $Y \in \mathcal{AS}(P \cup R)$ and $Y \notin \mathcal{AS}(Q \cup R)$. By Lemma 1, we can wlog assume that R is positive. Moreover, we know $H(R) \subseteq \mathcal{H}$ and $B(R) \subseteq \mathcal{B}$. Starting from $Y \in \mathcal{AS}(P \cup R)$, we first show that Property (i) from Definition 6 holds. We have $Y \models P \cup R$, and thus, $Y \models P$ as well as $Y \models R$ holds. It remains to show that for each $Y' \subset Y$, $Y' \models P^Y$ implies $Y'|_{\mathcal{H}} \subset Y|_{\mathcal{H}}$. Towards a contradiction, now suppose there exists an $Y' \subset Y$ such that $Y' \models P^Y$ and $Y'|_{\mathcal{H}} \not\subset Y|_{\mathcal{H}}$. Since $Y' \subset Y$, we have $Y'|_{\mathcal{H}} = Y|_{\mathcal{H}}$, and thus, $Y|_{\mathcal{H}} \subseteq Y'|_{\mathcal{H}}$. Moreover, $Y'|_{\mathcal{B}} \subseteq Y|_{\mathcal{B}}$ holds, and we get $Y \preceq^{\mathcal{B}}_{\mathcal{H}} Y'$. By $Y \models R$ and Lemma 2 this yields $Y' \models R$. But then $Y' \models (P^Y \cup R) = (P \cup R)^Y$, a contradiction to $Y \in \mathcal{AS}(P \cup R)$.

It remains to establish Property (ii) in Definition 6. From $Y \notin \mathcal{AS}(Q \cup R)$, we either get $Y \not\models Q \cup R$ or existence of an X such that $X \models (Q \cup R)^Y = (Q^Y \cup R)$. We already know that $Y \models R$. Hence, in the former case, i.e., $Y \not\models Q \cup R$, we get $Y \not\models Q$. Then, for any $X \subseteq Y$, (X, Y) is a witness for $P \subseteq_{\langle \mathcal{H},\mathcal{B}\rangle} Q$, and we are done. For the remaining case, where $X \models Q^Y$ and $X \models R$, we suppose towards a contradiction, that there exists an $X' \subset Y$, such that $X' \models P^Y$ and $X \preceq^{\mathcal{B}}_{\mathcal{H}} X'$ hold. The latter together with $X \models R$ yields $X' \models R$, following Lemma 2. Together with $X' \models P^Y$, we thus get $X' \models (P^Y \cup R) = (P \cup R)^Y$. Since $X' \subset Y$ this is in contradiction to $Y \in \mathcal{AS}(P \cup R)$. Thus (X, Y) is a witness for for $P \subseteq_{\langle \mathcal{H},\mathcal{B}\rangle} Q$.

(3) implies (2): Let (X, Y) be a witness for $P \subseteq_{\langle \mathcal{H},\mathcal{B}\rangle} Q$. We use the unary program

$$R = X|_{\mathcal{H}} \cup \{a \leftarrow b \mid a \in (Y \setminus X)|_{\mathcal{H}}, b \in (Y \setminus X)|_{\mathcal{B}}\}$$

and show $Y \in \mathcal{AS}(P \cup R) \setminus AS(Q \cup R)$. We first show $Y \in \mathcal{AS}(P \cup R)$. Since (X, Y) is a witness for $P \subseteq_{\langle \mathcal{H},\mathcal{B}\rangle} Q$, we know $Y \models P$. $Y \models R$ is easily checked and thus $Y \models P \cup R$. It remains to show that no $Z \subset Y$ satisfies $Z \models (P \cup R)^Y = P^Y \cup R$. Towards a contradiction suppose such a Z exists. Hence, $Z \models P^Y$ and $Z \models R$. By $Z \models R$, $X|_{\mathcal{H}} \subseteq Z|_{\mathcal{H}}$ has to hold. Since (X, Y) is a witness for $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$, $Z|_{\mathcal{H}} \subset Y|_{\mathcal{H}}$ holds, otherwise Property (i) in Definition 6 is violated. Hence, $X|_{\mathcal{H}} \subseteq Z|_{\mathcal{H}} \subset Y|_{\mathcal{H}}$ holds. We have $X \subset Y$ and, moreover, get $Z|_{\mathcal{B}} \not\subseteq X|_{\mathcal{B}}$ from Property (ii) in Definition 6, since $Z \models P^Y$ and $X|_{\mathcal{H}} \subseteq Z|_{\mathcal{H}}$ already hold. Now, $Z|_{\mathcal{B}} \subseteq Y|_{\mathcal{B}}$ by assumption, hence there exists an atom $b \in (Y \setminus X)|_{\mathcal{B}}$ contained in Z. We already know that $X|_{\mathcal{H}} \subseteq Z|_{\mathcal{H}} \subset Y|_{\mathcal{H}}$ has to hold. Hence, there exists at least one $a \in (Y \setminus X)|_{\mathcal{H}}$, not contained in Z. But then, we derive that $Z \not\models \{a \leftarrow b\}$. Since $a \leftarrow b \in R$, this is a contradiction to $Z \models R$.

It remains to show $Y \notin \mathcal{AS}(Q \cup R)$. If $Y \not\models Q$, we are done. So let $Y \models Q$. Since (X, Y) is a witness for $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$, we get $X \models Q^Y$ and $X \subset Y$. It is easy to see that $X \models R$ holds. Thus $X \models (Q^Y \cup R) = (Q \cup R)^Y$; $Y \notin \mathcal{AS}(Q \cup R)$ follows. □

As an immediate consequence of Lemma 3 and Proposition 2, we get that $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence problems which do not hold always possess simple counterexamples. As a special case we obtain the already mentioned fact that $\langle\mathcal{H},\emptyset\rangle$-equivalence amounts to uniform equivalence relative to $\mathcal{H}$.

Corollary 1. *For any $\mathcal{H}, \mathcal{B} \in \mathcal{U}$ and programs $P, Q \in \mathcal{C}_{\mathcal{U}}$, $P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ does not hold iff there exists a unary program $R \in \mathcal{C}_{\langle\mathcal{H},\mathcal{B}\rangle}$, such that $\mathcal{AS}(P \cup R) \neq \mathcal{AS}(Q \cup R)$; if $\mathcal{B} = \emptyset$, then $P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ does not hold iff there exists a set $F \subseteq \mathcal{H}$ of facts, such that $\mathcal{AS}(P \cup F) \neq \mathcal{AS}(Q \cup F)$.*

4.2 Introducing $\langle\mathcal{H},\mathcal{B}\rangle$-models

Next, we present the desired semantical characterization for $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence, which we call $\langle\mathcal{H},\mathcal{B}\rangle$-models. First, we introduce two further properties.

Definition 7. *Given $\mathcal{H} \subseteq \mathcal{U}$, an interpretation Y is an $\mathcal{H}$-*total model *for P iff $Y \models P$ and for all $Y' \subset Y$, $Y' \models P^Y$ implies $Y'|_{\mathcal{H}} \subset Y|_{\mathcal{H}}$.*

Definition 8. *Given $\mathcal{H}, \mathcal{B} \subseteq \mathcal{U}$, a pair (X, Y) of interpretations is called $\preceq^{\mathcal{B}}_{\mathcal{H}}$-maximal for P iff $X \models P^Y$ and, for each X' with $X \prec^{\mathcal{B}}_{\mathcal{H}} X' \subset Y$, $X' \not\models P^Y$.*

Observe that Y being an $\mathcal{H}$-total model for P matches Property (i) from Definition 6 and follows the same intuition. Being $\preceq^{\mathcal{B}}_{\mathcal{H}}$-maximal refers to being maximal (wrt subset inclusion) in the atoms from $\mathcal{H}$ and simultaneously minimal in the atoms from $\mathcal{B}$.

Definition 9. *Given $\mathcal{H}, \mathcal{B} \subseteq \mathcal{U}$, and interpretations $X \subseteq Y \subseteq \mathcal{U}$, a pair (X, Y) is an $\langle\mathcal{H},\mathcal{B}\rangle$-*model *of a program $P \in \mathcal{C}_{\mathcal{U}}$ iff Y is an $\mathcal{H}$-total model for P and, if $X \subset Y$, there exists an $X' \subset Y$ with $X'|_{\mathcal{H}\cup\mathcal{B}} = X$, such that (X', Y) is $\preceq^{\mathcal{B}}_{\mathcal{H}}$ maximal for P. The set of all $\langle\mathcal{H},\mathcal{B}\rangle$-models of a program P is denoted by $\sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P)$.*

Moreover, we call a pair (X, Y) *total* if $X = Y$, otherwise it is called *non-total*. Observe that each non-total $\langle\mathcal{H},\mathcal{B}\rangle$-model (X, Y) satisfies $X \subseteq Y|_{\mathcal{H}\cup\mathcal{B}}$ and $X|_{\mathcal{H}} \subset Y|_{\mathcal{H}}$.

Example 3. In Example 1, we already mentioned that $P = \{a \vee b \leftarrow;\ a \leftarrow b\}$ and $Q = \{a \leftarrow not\, b;\ b \leftarrow not\, a;\ a \leftarrow b\}$ are $\langle\{a, b\}, \{b\}\rangle$-equivalent. Hence, fix $\mathcal{H} = \{a, b\}$, $\mathcal{B} = \{b\}$, and let us compute the $\langle\mathcal{H},\mathcal{B}\rangle$-models of P, and resp., Q. In Example 2 we already have obtained the models of these programs as well as their reducts. There, we have seen that $Y_1 = \{a, b\}$ and $Y_2 = \{a\}$ are the models of both P and Q. Since $\mathcal{H} = \{a, b\}$, both are $\mathcal{H}$-total models for P and Q. So, (Y_1, Y_1) and (Y_2, Y_2) are the total $\langle\mathcal{H}, \mathcal{B}\rangle$-models of both programs. It remains to check whether the non-total $\langle\mathcal{H}, \mathcal{B}\rangle$-models of P and Q coincide. First observe that (Y_2, Y_1) is $\langle\mathcal{H}, \mathcal{B}\rangle$-model of both P and Q, as well. The interesting candidate is $(\emptyset, Y_1)$ since $\emptyset$ is model of Q^{Y_1} but not of P^{Y_1}. Hence, $(\emptyset, Y_1)$ cannot be $\langle\mathcal{H}, \mathcal{B}\rangle$-model of P. But $(\emptyset, Y_1)$ is also not $\langle\mathcal{H}, \mathcal{B}\rangle$-model of Q, since there exists an interpretation X' satisfying $\emptyset \prec^{\mathcal{B}}_{\mathcal{H}} X' \subset Y$, which is model of Q^{Y_1}, viz. $X' = \{a\}$. In fact, $\emptyset|_{\mathcal{H}} \subset X'|_{\mathcal{H}}$ and $\emptyset|_{\mathcal{B}} = X'|_{\mathcal{B}}$ hold.

For $\mathcal{H} = \{a\}$ and $\mathcal{B} = \{a, b\}$, one can show that (Y_2, Y_2) is the only $\langle\mathcal{H}, \mathcal{B}\rangle$-model (over $\{a, b\}$) of P as well as of Q, since Y_1 is no $\mathcal{H}$-total model in this setting. $\diamond$

Before stating our main theorem, we require one further lemma.

Lemma 4. *Let $P, Q \in \mathcal{C}_{\mathcal{U}}$, $\mathcal{H}, \mathcal{B} \subseteq \mathcal{U}$, and Y be an interpretation. Then, $(Y, Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P) \setminus \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$ iff there is a witness (X, Y) to $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ with $X|_{\mathcal{H}} = Y|_{\mathcal{H}}$.*

Proof. For the only-if direction, we directly obtain from $(Y, Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P)$, that Property (i) in Definition 6 holds. To show the remaining Property (ii), observe that from $(Y, Y) \notin \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$, we either have $Y \not\models Q$ or existence of some $Y' \subset Y$, such that $Y' \models Q^Y$ and $Y'|_{\mathcal{H}} = Y|_{\mathcal{H}}$. In the former case, we are already done, and get that any (X, Y) with $X \subseteq Y$ is a witness for $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$, in particular for $X|_{\mathcal{H}} = Y|_{\mathcal{H}}$. It remains to show that, in case $Y \models Q$, and for some $Y' \subset Y$ with $Y'|_{\mathcal{H}} = Y|_{\mathcal{H}}$, $Y' \models Q^Y$, each X' with $Y' \preceq^{\mathcal{B}}_{\mathcal{H}} X' \subset Y$ satisfies $X' \not\models P^Y$. By definition, this would make (Y', Y) a witness for $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$. Towards a contradiction, suppose such an X' exists. But then, from $Y' \preceq^{\mathcal{B}}_{\mathcal{H}} X' \subset Y$ and $Y'|_{\mathcal{H}} = Y|_{\mathcal{H}}$, we get $Y'|_{\mathcal{H}} = X'|_{\mathcal{H}} = Y|_{\mathcal{H}}$. Thus, Y cannot be an $\mathcal{H}$-total model of P; a contradiction to $(Y, Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P)$.

For the if-direction, let (X, Y) be a witness for $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$. Property (i) in Definition 6 yields $(Y, Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P)$. It remains to show $(Y, Y) \notin \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$. Now, (X, Y) being a witness implies that either $Y \not\models Q$ or $X \models Q^Y$, where $X|_{\mathcal{H}} = Y|_{\mathcal{H}}$ and $X \subset Y$ hold. Both cases prevent (Y, Y) from being $\langle\mathcal{H}, \mathcal{B}\rangle$-model of Q. $\square$

Theorem 1. *For any programs $P, Q \in \mathcal{C}_{\mathcal{U}}$ and alphabets $\mathcal{H}, \mathcal{B} \subseteq \mathcal{U}$, $P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ holds iff $\sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P) = \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$.*

Proof. If-direction: Suppose that either $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ or $Q \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} P$ does not hold. Let us wlog assume $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ does not hold (the other case is symmetric). By Lemma 3, then a witness (X, Y) to $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ exists. By Property (i) in Definition 6, we immediately get $(Y, Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P)$. In case $(Y, Y) \notin \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$ we are already done. So suppose $(Y, Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$. Hence, we can assume $Y \models Q$, and by Lemma 4, $X|_{\mathcal{H}} \neq Y|_{\mathcal{H}}$. Since (X, Y) is a witness for $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$, we get that $X \models Q^Y$ holds, and for each X' with $X \preceq^{\mathcal{B}}_{\mathcal{H}} X' \subset Y$, $X' \not\models P^Y$. Consider now an arbitrary pair (Z, Y) of interpretations with $Z \subset Y$ which is $\preceq^{\mathcal{B}}_{\mathcal{H}}$-maximal for Q. Then $X \preceq^{\mathcal{B}}_{\mathcal{H}} Z$ has to hold and since $(Y, Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$, Y is an $\mathcal{H}$-total model of Q, and we obtain $(Z|_{\mathcal{H}\cup\mathcal{B}}, Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$. On that other hand, $(Z|_{\mathcal{H}\cup\mathcal{B}}, Y) \notin \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P)$ holds. This is a consequence of the observation that for each X' with $X \preceq^{\mathcal{B}}_{\mathcal{H}} X' \subset Y$, $X' \not\models P^Y$, (since (X, Y) is a witness for $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$), and by the fact that $X \preceq^{\mathcal{B}}_{\mathcal{H}} Z$.

Only-if direction: Wlog assume $(X,Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P) \setminus \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$; again, the other case is symmetric. From $(X,Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P)$, $(Y,Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P)$ follows by Definition 9. Hence, if $(Y,Y) \notin \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$, we are done, since we know from Lemma 4 that then, there exists a witness for $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ and we get by Lemma 3, that $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ does not hold. Consequently, $P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ cannot hold as well. Thus, let $X \subset Y$, and $(Y,Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$. We distinguish between two cases: First suppose there exists an X' with $X \preceq^{\mathcal{B}}_{\mathcal{H}} X' \subset Y$, such that $(X',Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$. Since $(X,Y) \notin \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$, by definition of $\langle\mathcal{H},\mathcal{B}\rangle$-models, $X \prec^{\mathcal{B}}_{\mathcal{H}} X'$ has to hold, and there exists a $Z \subset Y$ with $Z|_{\mathcal{H}\cup\mathcal{B}} = X'$, such that $Z \models Q^Y$. We show that (Z,Y) is a witness for $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$. Since $(Y,Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P)$, Property (i) in Definition 6 holds. We know $Z \models Q^Y$, and since $(X,Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P)$, we get by definition of $\langle\mathcal{H},\mathcal{B}\rangle$-models, that, for each X'' with $X \prec^{\mathcal{B}}_{\mathcal{H}} X'' \subset Y$, $X'' \not\models P^Y$. Now since $X \prec^{\mathcal{B}}_{\mathcal{H}} Z$, Property (ii) in Definition 6 holds for Z (instead of X) as well. This shows that (Z,Y) is a witness for $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$. So suppose, for each X' with $X \preceq^{\mathcal{B}}_{\mathcal{H}} X' \subset Y$, $(X',Y) \notin \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$ holds. We have $(X,Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(P)$, thus there exists a $Z \subset Y$, with $Z|_{\mathcal{H}\cup\mathcal{B}} = X$, such that $Z \models P^Y$. We show that (Z,Y) is a witness for the reverse problem, $Q \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} P$. From $(Y,Y) \in \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$, we get that Property (i) in Definition 6 is satisfied for Q and Y. Moreover, we have $Z \models P^Y$. It remains to show that, for each X'' with $X \preceq^{\mathcal{B}}_{\mathcal{H}} X'' \subset Y$, $X'' \not\models Q^Y$. This holds by assumption, i.e., $(X',Y) \notin \sigma_{\langle\mathcal{H},\mathcal{B}\rangle}(Q)$, for each X' with $X \preceq^{\mathcal{B}}_{\mathcal{H}} X' \subset Y$. Hence, both cases yield a witness, either for $P \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ or $Q \subseteq_{\langle\mathcal{H},\mathcal{B}\rangle} P$. By Lemma 3 and Proposition 2, $P \equiv_{\langle\mathcal{H},\mathcal{B}\rangle} Q$ does not hold. □

5 Special Cases

In this section, we analyze how $\langle\mathcal{H}, B\rangle$-models behave for special instantiations of $\mathcal{H}$ and $\mathcal{B}$. We first consider the case where either $\mathcal{H} = \mathcal{U}$ or $\mathcal{B} = \mathcal{U}$. We call the former scenario *body-relativized* and the latter *head-relativized*. Then, we sketch more general settings where the only restriction is that either $\mathcal{H} \subseteq \mathcal{B}$ or $\mathcal{B} \subseteq \mathcal{H}$ holds.

5.1 Body-Relativized and Head-Relativized Equivalence

First, we consider $\langle\mathcal{U},\mathcal{B}\rangle$-equivalence problems, where $\mathcal{U}$ is fixed by the universe, but $\mathcal{B}$ can be arbitrarily chosen. Note that $\langle\mathcal{U},\mathcal{B}\rangle$-equivalence ranges from strong (setting $\mathcal{B} = \mathcal{U}$) to uniform equivalence (setting $\mathcal{B} = \emptyset$ and cf. Corollary 1) and thus provides a common view on these two important problems, as well as on problems "inbetween" them. Second, head-relativized equivalence problems, $P \equiv_{\langle\mathcal{H},\mathcal{U}\rangle} Q$, have as special cases once more strong equivalence (now by setting $\mathcal{H} = \mathcal{U}$) but also the case where $\mathcal{H} = \emptyset$ is of interest, since it amounts to check whether P and Q possess the same answer sets under any addition of constraints. It is quite obvious that this holds iff P and Q are ordinarily equivalent, since constraints can only "rule out" answer sets. That observation is also reflected in Corollary 1, since the only unary program in $\mathcal{C}_{\langle\emptyset,\mathcal{U}\rangle}$ is the empty program.

The following result simplifies the definition of $\preceq^{\mathcal{B}}_{\mathcal{H}}$ within these settings.

Proposition 3. *For interpretations $V, Z \subseteq \mathcal{U}$ and an alphabet $\mathcal{A} \subseteq \mathcal{U}$, it holds that (i) $V \preceq^{\mathcal{A}}_{\mathcal{U}} Z$ iff $V \subseteq Z$ and $V|_{\mathcal{A}} = Z|_{\mathcal{A}}$; and (ii) $V \preceq^{\mathcal{U}}_{\mathcal{A}} Z$ iff $Z \subseteq V$ and $V|_{\mathcal{A}} = Z|_{\mathcal{A}}$.*

Thus, maximizing wrt $\preceq^{\mathcal{B}}_{\mathcal{H}}$ becomes in case of $\mathcal{H} = \mathcal{U}$ a form of $\subseteq$-maximization; and in case of $\mathcal{B} = \mathcal{U}$ a form of $\subseteq$-minimization. Obviously, both neutralize themselves for $\mathcal{B} = \mathcal{H} = \mathcal{U}$, i.e., in the strong equivalence setting, where $V \preceq^{\mathcal{U}}_{\mathcal{U}} Z$ iff $V = Z$.

For body-relativized equivalence, our characterization now simplifies as follows.

Corollary 2. *A pair (X, Y) of interpretations is an $\langle \mathcal{U}, \mathcal{B} \rangle$-model of $P \in \mathcal{C}_{\mathcal{U}}$ iff $X \subseteq Y$, $Y \models P$, $X \models P^Y$, and for all X' with $X \subset X' \subset Y$ and $X'|_{\mathcal{B}} = X|_{\mathcal{B}}$, $X' \not\models P^Y$.*

Observe that for the notions inbetween strong and uniform equivalence the maximality test, which tests if each X' with $X \subset X' \subset Y$ and $X'|_{\mathcal{B}} = X|_{\mathcal{B}}$ yields $X' \not\models P^Y$, gets more localized the more atoms are contained in $\mathcal{B}$. In particular, for $\mathcal{B} = \mathcal{U}$ it disappears and we end up with a very simple condition for $\langle \mathcal{U}, \mathcal{U} \rangle$-models which exactly matches the definition of SE-models by Turner [11]: a pair (X, Y) of interpretations is an SE-model of a program P iff $X \subseteq Y$, $Y \models P$, and $X \models P^Y$.

For $\mathcal{B} = \emptyset$, on the other hand, we observe that $X'|_{\mathcal{B}} = X|_{\mathcal{B}}$ always holds for $\mathcal{B} = \emptyset$. Thus, a pair (X, Y) is a $\langle \mathcal{U}, \emptyset \rangle$-model of a program P, if $X \subseteq Y$, $Y \models P$, $X \models P^Y$, and for all X' with $X \subset X' \subset Y$, $X' \not\models P^Y$. These conditions are now exactly the ones given for UE-models following [2]. Hence, Corollary 2 provides a common view on the characterizations of uniform and strong equivalence.

For head-relativized equivalence notions, simplifications are as follows.

Corollary 3. *A pair (X, Y) of interpretations is an $\langle \mathcal{H}, \mathcal{U} \rangle$-model of $P \in \mathcal{C}_{\mathcal{U}}$ iff $X \subseteq Y$, Y is an $\mathcal{H}$-total model for P, $X \models P^Y$, and for each $X' \subset X$ with $X'|_{\mathcal{H}} = X|_{H}$, $X' \not\models P^Y$.*

In the case of $\mathcal{H} = \mathcal{U}$, $\langle \mathcal{H}, \mathcal{U} \rangle$-models again reduce to SE-models. The other special case is $\mathcal{H} = \emptyset$. Recall that $\langle \emptyset, \mathcal{U} \rangle$-equivalence amounts to ordinary equivalence. $\langle \emptyset, \mathcal{U} \rangle$-models thus characterize answer sets as follows: First, Y is an $\emptyset$-total model for P, iff no $X \subset Y$ satisfies $X \models P^Y$. Moreover, this requires that all $\langle \emptyset, \mathcal{U} \rangle$-models are total. So, the condition in Corollary 3 for $X \subset Y$ is immaterial and we have a one-to-one correspondence between $\langle \emptyset, \mathcal{U} \rangle$-models and answer sets of a program.

5.2 $\mathcal{B} \subseteq \mathcal{H}$ - and $\mathcal{H} \subseteq \mathcal{B}$ - Equivalence

Due to lack of space, we just highlight a few results here, in order to establish a connection between $\langle \mathcal{H}, \mathcal{B} \rangle$-models and relativized SE- and UE-models, as defined in [12].

Proposition 4. *For interpretations $V, Z \subseteq \mathcal{U}$ and alphabets $\mathcal{H}, \mathcal{B} \subseteq \mathcal{U}$ with $\mathcal{B} \subseteq \mathcal{H}$ (resp., $\mathcal{H} \subseteq \mathcal{B}$), $V \preceq^{\mathcal{B}}_{\mathcal{H}} Z$ iff $V|_{\mathcal{H}} \subseteq Z|_{\mathcal{H}}$ and $V|_{\mathcal{B}} = Z|_{\mathcal{B}}$ (resp., iff $Z|_{\mathcal{B}} \subseteq V|_{\mathcal{B}}$ and $V|_{\mathcal{H}} = Z|_{\mathcal{H}}$). Moreover, if $\mathcal{A} = \mathcal{H} = \mathcal{B}$, $V \preceq^{\mathcal{B}}_{\mathcal{H}} Z$ iff $V|_{\mathcal{A}} = Z|_{\mathcal{A}}$.*

Observe that $\preceq^{\mathcal{A}}_{\mathcal{A}}$-maximality (in the sense of Definition 8) of a pair (X, Y) for P reduces to test $X \models P^Y$. Thus, to make $(X|_{\mathcal{A}}, Y)$ an $\langle \mathcal{A}, \mathcal{A} \rangle$-model of P, we just additionally need $\mathcal{A}$-totality of Y. In other words, we obtain the following criteria.

Corollary 4. *Given $\mathcal{A} \subseteq \mathcal{U}$, a pair (X, Y) of interpretations is an $\langle \mathcal{A}, \mathcal{A} \rangle$-model of a program $P \in \mathcal{C}_{\mathcal{U}}$, iff (1) $X = Y$ or $X \subset Y|_{\mathcal{A}}$, (2) $Y \models P$ and for each $Y' \subset Y$, $Y' \models P^Y$ implies $Y'|_{\mathcal{A}} \subset Y|_{\mathcal{A}}$; and (3) if $X \subset Y$ then there exists an $X' \subseteq Y$ with $X'|_{\mathcal{A}} = X$, such that $X' \models P^Y$.*

This exactly matches the definition of $\mathcal{A}$-SE-models according to [12]. Finally, if we switch from $\langle\mathcal{A},\mathcal{A}\rangle$-equivalence to $\langle\mathcal{A},\emptyset\rangle$-equivalence (i.e., from relativized strong to relativized uniform equivalence) we obtain the following result for $\langle\mathcal{A},\emptyset\rangle$-models which coincides with an explicit definition of $\mathcal{A}$-UE-models according to [12].

Corollary 5. *Given $\mathcal{A}\subseteq\mathcal{U}$, a pair (X,Y) of interpretations is an $\langle\mathcal{A},\emptyset\rangle$-model of $P\in\mathcal{C}_{\mathcal{U}}$, iff (1) and (2) from Corollary 4 hold, and if $X\subset Y$ then there exists $X'\subseteq Y$ such that $X'|_{\mathcal{A}}=X$, $X'\models P^Y$, and for each $X''\subset Y$ with $X'|_{\mathcal{A}}\subset X''|_{\mathcal{A}}$, $X''\not\models P^Y$.*

6 Computational Issues

Former results on uniform [2] or relativized [12] equivalence show that these problems are, in general, Π_2^P-hard for disjunctive logic programs. Hence, $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence is Π_2^P-hard as well. However, Π_2^P-membership still holds in the view of Corollary 1. In particular, it is sufficient to guess an interpretation Y and a unary program $R\in\mathcal{C}_{\langle\mathcal{H},\mathcal{B}\rangle}$, and then to check whether Y is contained in either $\mathcal{AS}(P\cup R)$ or $\mathcal{AS}(Q\cup R)$, but not in both. Answer-set checking is in coNP, and since one can safely restrict Y and R to contain only atoms which also occur in P or Q, this algorithm for disproving $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence runs in nondeterministic polynomial time with access to an NP-oracle. Thus, that problem is in Σ_2^P, and consequently $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence is in Π_2^P.

Concerning implementation, we briefly discuss an approach which makes use of Corollary 1 in a similar manner and compiles $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence into ordinary equivalence for which a dedicated system exists [9]; a similar method was also discussed in [12, 10]. The idea hereby is to incorporate the guess of the unary context programs over the specified alphabets in both programs accordingly. To this end, let, for an $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence problem between programs P and Q, f as well as $c_{a,b}$ and $\bar{c}_{a,b}$ for each $a\in\mathcal{H}$, $b\in\mathcal{B}\cup\{f\}$, be new distinct atoms, not occurring in $P\cup Q$. Then, $P\equiv_{\langle\mathcal{H},B\rangle}Q$ holds iff $P^+_{\langle\mathcal{H},B\rangle}$ and $Q^+_{\langle\mathcal{H},B\rangle}$ are ordinarily equivalent, where, for $R\in\{P,Q\}$,

$$R^+_{\langle\mathcal{H},B\rangle}=R\cup\Big\{c_{a,b}\vee\bar{c}_{a,b}\leftarrow;\ a\leftarrow b,c_{a,b}\mid a\in\mathcal{H},b\in\mathcal{B}\cup\{f\}\Big\}\cup\{f\leftarrow\}.$$

In fact, the role of atoms $c_{a,f}$ is to guess a set of facts $F\subseteq\mathcal{H}$, while atoms $c_{a,b}$ with $b\neq f$ guess a subset of unary rules $a\leftarrow b$ with $a\in\mathcal{H}$ and $b\in\mathcal{B}$.

7 Conclusion

The aim of this work is to provide a general and uniform characterization for different equivalence problems, which have been handled by inherently different concepts, so far. We have introduced an equivalence notion parameterized by two alphabets to restrict the atoms allowed to occur in the heads, and respectively, bodies of the context programs. We showed that our approach captures the most important equivalence notions studied, including strong and uniform equivalence as well as relativized notions thereof.

Figure 1 gives an overview of $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence and its special cases, i.e., relativized uniform equivalence (RUE), relativized strong equivalence (RSE), body-relativized equivalence (BRE), and head-relativized equivalence (HRE). On the bottom line

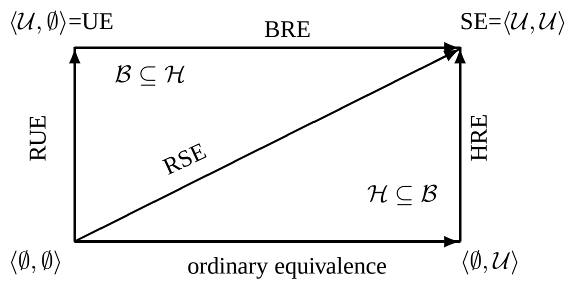
$\langle\mathcal{U},\emptyset\rangle$=UE
BRE
SE=$\langle\mathcal{U},\mathcal{U}\rangle$
$\mathcal{B}\subseteq\mathcal{H}$
RUE
RSE
HRE
$\mathcal{H}\subseteq\mathcal{B}$
$\langle\emptyset,\emptyset\rangle$
ordinary equivalence
$\langle\emptyset,\mathcal{U}\rangle$

Fig. 1. The landscape of $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence with either $\mathcal{H}\subseteq\mathcal{B}$ or $\mathcal{B}\subseteq\mathcal{H}$.

we have ordinary equivalence, while the top-left corner amounts to uniform equivalence (UE) and the top-right corner to strong equivalence (SE).

Future work includes the study of further properties of $\langle\mathcal{H},\mathcal{B}\rangle$-equivalence, as well as potential applications, which include relations to open logic programs [1] and new concepts for program simplification [3]. Also an extension in the sense of [5], where a further alphabet is used to specify the atoms which have to coincide in comparing the answer sets is considered. While [5] provides a characterization for relativized *strong* equivalence with projection, recent work [8] addresses the problem of relativized *uniform* equivalence with projection. Our results may be a basis to provide a common view on these two recent characterizations, as well.

References

1. P. Bonatti. Reasoning with Open Logic Programs. In *Proc. LPNMR'01*, vol. 2173 of *LNCS*, pg. 147–159. Springer, 2001.
2. T. Eiter and M. Fink. Uniform Equivalence of Logic Programs under the Stable Model Semantics. In *Proc. ICLP'03*, vol. 2916 in LNCS, pg. 224–238. Springer, 2003.
3. T. Eiter, M. Fink, H. Tompits, and S. Woltran. Simplifying Logic Programs Under Uniform and Strong Equivalence. In *Proc. LPNMR'03*, vol. 2923 of *LNCS*, pg. 87–99. Springer, 2004.
4. T. Eiter, M. Fink, and S. Woltran. Semantical Characterizations and Complexity of Equivalences in Stable Logic Programming. Technical Report INFSYS RR-1843-05-01, TU Wien, Austria, 2005. Accepted for publication in *ACM Transactions on Computational Logic*.
5. T. Eiter, H. Tompits, and S. Woltran. On Solution Correspondences in Answer Set Programming. In *Proc. IJCAI'05*, pg. 97–102. Professional Book Center, 2005.
6. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.
7. V. Lifschitz, D. Pearce, and A. Valverde. Strongly Equivalent Logic Programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.
8. J. Oetsch, H. Tompits, and S. Woltran. Facts do not Cease to Exist Because They are Ignored: Relativised Uniform Equivalence with Answer-Set Projection. In *Proc. CENT'07*, 2007.
9. E. Oikarinen and T. Janhunen. Verifying the Equivalence of Logic Programs in the Disjunctive Case. In *Proc. LPNMR'03*, vol. 2923 of *LNCS*, pg. 180–193. Springer, 2004.
10. E. Oikarinen and T. Janhunen. Modular Equivalence for Normal Logic Programs. In *Proc. ECAI'06*, pg. 412–416. IOS Press, 2006.
11. H. Turner. Strong Equivalence Made Easy: Nested Expressions and Weight Constraints. *Theory and Practice of Logic Programming*, 3(4-5):602–622, 2003.
12. S. Woltran. Characterizations for Relativized Notions of Equivalence in Answer Set Programming. In *Proc. JELIA'04*, vol. 3229 of *LNCS*, pg. 161–173. Springer, 2004.

Alternative Characterizations for Program Equivalence under Answer-Set Semantics: Preliminary Report[⋆]

Martin Gebser[1], Torsten Schaub[1], Hans Tompits[2], and Stefan Woltran[2]

[1] Institut für Informatik, Universität Potsdam,
August-Bebel-Straße 89, D-14482 Potsdam, Germany
{gebser,torsten}@cs.uni-potsdam.de
[2] Institut für Informationssysteme 184/3, Technische Universität Wien,
Favoritenstraße 9-11, A-1040 Vienna, Austria
{tompits,stefan}@kr.tuwien.ac.at

Abstract. Logic programs under answer-set semantics constitute an important tool for declarative problem solving. In recent years, two research issues received growing attention. On the one hand, concepts like loops and elementary sets have been proposed in order to extend Clark's completion for computing answer sets of logic programs by means of propositional logic. On the other hand, different concepts of program equivalence, like strong or uniform equivalence, have been studied in the context of program optimization and modular programming. In this paper, we bring these two lines of research together and provide alternative characterizations for different conceptions of equivalence in terms of unfounded sets, along with the related concepts of loops and elementary sets. Our results yield new insights into the model theory of equivalence checking. We further exploit these characterizations to develop novel encodings of program equivalence in terms of propositional logic.

1 Introduction

The increasing success of answer-set programming [1] as a tool for declarative problem solving has produced the need to optimize logic programs in various ways, while leaving their semantics unaffected. Different scenarios have led to different criteria of when a program's semantics is preserved. Formally, this is reflected by different definitions of program equivalence (see below). For instance, in solving, one is usually interested in program modifications preserving all answer sets, while program optimization requires a stronger definition, guaranteeing that replacing one subprogram by another preserves answer sets, no matter how the encompassing program looks like.

In what follows, we elaborate upon the model theory underlying program equivalence, dealing primarily with the well-known concepts of SE- and UE-models [2, 3]. In particular, we provide a new perspective on these semantic structures by using *unfounded sets* [4] and related constructs like *elementary sets* [5] and *loops* [6, 7]. Recall that SE- and UE-models are defined as pairs (X, Y), where Y is a model of a given logic program P and X is a model of the reduct P^Y. The major difference between this

[⋆] Partially supported by the Austrian Science Fund (FWF) under project P15068-INF.

characterization and our approach is that we refer to $(Y \setminus X)$ rather than to X itself. As it turns out, an explicit reference to the reduct and its models is not required, rather, the respective unfoundedness property possessed by $(Y \setminus X)$ allows us to characterize and distinguish SE- and UE-models.

2 Background

A propositional *disjunctive logic program* is a finite set of rules of the form

$$a_1 \vee \cdots \vee a_k \leftarrow a_{k+1}, \ldots, a_m, not\, a_{m+1}, \ldots, not\, a_n, \tag{1}$$

where $1 \leq k \leq m \leq n$ and every a_i ($1 \leq i \leq n$) is a propositional atom from some universe $\mathcal{U}$; not denotes default negation. A rule as in (1) is called a *fact* if $k = n = 1$; it is said to be *positive* if $m = n$. For a rule r, $H(r) = \{a_1, \ldots, a_k\}$ is the *head* of r, $B(r) = \{a_{k+1}, \ldots, a_m, not\, a_{m+1}, \ldots, not\, a_n\}$ is the *body* of r, $B^+(r) = \{a_{k+1}, \ldots, a_m\}$ is the *positive body* of r, and $B^-(r) = \{a_{m+1}, \ldots, a_n\}$ is the *negative body* of r. We sometimes denote a rule r by $H(r) \leftarrow B(r)$.

The (*positive*) *dependency graph* of a program P is $(\mathcal{U}, \{(a, b) \mid r \in P, a \in H(r), b \in B^+(r)\})$. A nonempty set $U \subseteq \mathcal{U}$ is a *loop* of P if the subgraph of the dependency graph of P induced by U is strongly connected. Similar to Lee [7], we consider every singleton as a loop. A program P is *tight* [8, 9] if every loop of P is a singleton.

As usual, an interpretation Y is a set of atoms (i.e., a subset of $\mathcal{U}$). For a rule r, we write $Y \models r$ iff $H(r) \cap Y \neq \emptyset$, $B^+(r) \not\subseteq Y$, or $B^-(r) \cap Y \neq \emptyset$. An interpretation Y is a *model* of a program P, denoted by $Y \models P$, iff $Y \models r$ for every $r \in P$. The *reduct* of P with respect to Y is $P^Y = \{H(r) \leftarrow B^+(r) \mid r \in P, B^-(r) \cap Y = \emptyset\}$. An interpretation Y is an *answer set* of P iff Y is a minimal model of P^Y.

Two programs, P and Q, are *ordinarily equivalent* iff their answer sets coincide. Furthermore, P and Q are *strongly equivalent* [10] (resp., *uniformly equivalent* [3]) iff, for every program (resp., set of facts) R, $P \cup R$ and $Q \cup R$ are ordinarily equivalent.

For interpretations X, Y, the pair (X, Y) is an *SE-interpretation* iff $X \subseteq Y$. Given an SE-interpretation (X, Y) and a program P, (X, Y) is an *SE-model* [2] of P iff $Y \models P$ and $X \models P^Y$. An SE-model (X, Y) is a *UE-model* [3] of P iff there is no SE-model (Z, Y) of P such that $X \subset Z \subset Y$. The set of all SE-models (resp., UE-models) of P is denoted by $SE(P)$ (resp., $UE(P)$). Two programs P and Q are strongly (resp., uniformly) equivalent iff $SE(P) = SE(Q)$ (resp., $UE(P) = UE(Q)$) [2, 3].

Example 1. Consider $P = \{a \vee b \leftarrow\}$ and $Q = \{a \leftarrow not\, b;\ b \leftarrow not\, a\}$. Clearly, both programs are ordinarily equivalent as $\{a\}$ and $\{b\}$ are their respective answer sets. However, they are not strongly equivalent. Indeed, since P is positive, we have that $SE(P) = \{(a, a), (b, b), (ab, ab), (a, ab), (b, ab)\}$.[1] For Q, we have to take the reduct into account. In particular, we have $Q^{\{a,b\}} = \emptyset$, and so any interpretation is a model of $Q^{\{a,b\}}$. Hence, each pair (X, ab) with $X \subseteq \{a, b\}$ is an SE-model of Q. We thus have $SE(Q) = \{(a, a), (b, b), (ab, ab), (a, ab), (b, ab), (\emptyset, ab)\}$. That is, $SE(P) \neq SE(Q)$,

[1] Whenever convenient, we use strings like ab as a shorthand for $\{a, b\}$. As a convention, we let universe $\mathcal{U}$ be the set of atoms occurring in the programs under consideration.

so P and Q are not strongly equivalent. A witness for this is $R = \{a \leftarrow b;\ b \leftarrow a\}$, as $P \cup R$ has $\{a, b\}$ as its (single) answer set, while $Q \cup R$ has no answer set.

Concerning uniform equivalence, observe first that $UE(P) = SE(P)$. This is not the case for Q, where the SE-model $(\emptyset, ab)$ drops out since there exist further SE-models (Z, ab) of Q with $\emptyset \subset Z \subset \{a, b\}$, viz. (a, ab) and (b, ab). One can check that $(\emptyset, ab)$ is in fact the only pair in $SE(Q)$ that is no UE-model of Q. So, $UE(Q) = SE(Q) \setminus \{(\emptyset, ab)\} = SE(P) = UE(P)$. Thus, P and Q are uniformly equivalent. $\Diamond$

We conclude this section with the following known properties. First, for any program P and any interpretation Y, the following statements are equivalent: (i) $Y \models P$; (ii) $Y \models P^Y$; (iii) $(Y, Y) \in SE(P)$; and (iv) $(Y, Y) \in UE(P)$. Second, if $Y \models P$, Y is an answer set of P iff, for each SE-model (resp., UE-model) (X, Y) of P, $X = Y$.

3 Model-Theoretic Characterizations by Unfounded Sets

In this section, we exploit the notion of an unfounded set [4] and provide alternative characterizations of models for logic programs and program equivalence. Roughly speaking, the aim of unfounded sets is to collect atoms that cannot be derived from a program with respect to a fixed interpretation. Given the closed-world reasoning flavor of answer sets, such atoms are considered to be false. However, we shall relate here unfounded sets also to SE- and UE-models, and thus to concepts that do not fall under the closed-world assumption (since they implicitly deal with program extensions). For the case of uniform equivalence, we shall also employ the recent concept of elementarily unfounded sets [5], which via elementary sets decouple the idea of (minimal) unfounded sets from fixed interpretations. Finally, we link our results to loops.

Given a program P and an interpretation Y, a set $U \subseteq \mathcal{U}$ is *unfounded* [4] for P with respect to Y if, for each $r \in P$, at least one of the following conditions holds:

1. $H(r) \cap U = \emptyset$,
2. $H(r) \cap (Y \setminus U) \neq \emptyset$,
3. $B^+(r) \not\subseteq Y$ or $B^-(r) \cap Y \neq \emptyset$, or
4. $B^+(r) \cap U \neq \emptyset$.

Note that the empty set is unfounded for any program P with respect to any interpretation, since the first condition, $H(r) \cap \emptyset = \emptyset$, holds for all $r \in P$.

Example 2. Consider the following program:

$$P = \left\{ \begin{array}{lllllll} r_1: & a \vee b \leftarrow & \quad & r_3: & c \leftarrow a & \quad r_5: & c \leftarrow b, d \\ r_2: & b \vee c \leftarrow & \quad & r_4: & d \leftarrow not\ b & \quad r_6: & d \leftarrow c, not\ a \end{array} \right\}.$$

Let $U = \{c, d\}$. We have $H(r_1) \cap U = \{a, b\} \cap \{c, d\} = \emptyset$, that is, r_1 satisfies Condition 1. For r_5 and r_6, $B^+(r_5) \cap U = \{b, d\} \cap \{c, d\} \neq \emptyset$ and $B^+(r_6) \cap U = \{c\} \cap \{c, d\} \neq \emptyset$. Hence, both rules satisfy Condition 4. Furthermore, consider the interpretation $Y = \{b, c, d\}$. We have $H(r_2) \cap (Y \setminus U) = \{b, c\} \cap \{b\} \neq \emptyset$, thus r_2 satisfies Condition 2. Finally, for r_3 and r_4, $B^+(r_3) = \{a\} \not\subseteq \{b, c, d\} = Y$ and $B^-(r_4) \cap Y = \{b\} \cap \{b, c, d\} \neq \emptyset$, that is, both rules satisfy Condition 3. From the fact that each rule in P satisfies at least one of the unfoundedness conditions, we conclude that $U = \{c, d\}$ is unfounded for P with respect to $Y = \{b, c, d\}$. $\Diamond$

The basic relation between unfounded sets and answer sets is as follows.

Proposition 1 ([11]). *Let P be a program and Y an interpretation. Then, Y is an answer set of P iff $Y \models P$ and no nonempty subset of Y is unfounded for P with respect to Y.*

Example 3. Program P in Example 2 has two answer sets: $\{a, c, d\}$ and $\{b\}$. For the latter, we just have to check that $\{b\}$ is not unfounded for P with respect to $\{b\}$ itself, which holds in view of either rule r_1 or r_2. To verify via unfounded sets that $Y = \{a, c, d\}$ is an answer set of P, we have to check all nonempty subsets of Y. For instance, take $U = \{c, d\}$. We have already seen that r_1, r_5, and r_6 satisfy Condition 1 or 4, respectively; but the remaining rules r_2, r_3, and r_4 violate all four unfoundedness conditions for U with respect to Y. $\diamondsuit$

We next detail the relation between unfounded sets and models of logic programs as well as of their reducts. First, we have the following relationships between models and unfounded sets.

Lemma 1. *Let P be a program and Y an interpretation. Then, the following statements are equivalent: (a) $Y \models P$; (b) every set $U \subseteq \mathcal{U} \setminus Y$ is unfounded for P with respect to Y; and (c) every singleton $U \subseteq \mathcal{U} \setminus Y$ is unfounded for P with respect to Y.*

Proof. (*a*) $\Rightarrow$ (*b*): Assume that some set $U \subseteq \mathcal{U} \setminus Y$ is not unfounded for P with respect to Y. Then, for some rule $r \in P$, we have

(α) $H(r) \cap U \neq \emptyset$,
(β) $H(r) \cap (Y \setminus U) = \emptyset$,
(γ) $B^+(r) \subseteq Y$ and $B^-(r) \cap Y = \emptyset$, and
(δ) $B^+(r) \cap U = \emptyset$.

Since $U \cap Y = \emptyset$ by the assumption, we conclude from (β) that $H(r) \cap Y = \emptyset$. Since (γ) holds in addition, we have $Y \not\models r$ and thus $Y \not\models P$.

(*b*) $\Rightarrow$ (*c*) is trivial.

(*c*) $\Rightarrow$ (*a*): Assume $Y \not\models P$. Then, there is a rule $r \in P$ such that $Y \not\models r$, that is, $H(r) \cap Y = \emptyset$ and (γ) hold. By the definition of rules, $H(r) \neq \emptyset$. So, consider any $a \in H(r)$ and the singleton $U = \{a\}$. Clearly, (α) holds for r, and (β) holds by $H(r) \cap Y = \emptyset$. Finally, since $B^+(r) \subseteq Y$ and $a \notin Y$, (δ) holds as well. That is, there is a singleton $U \subseteq \mathcal{U} \setminus Y$ that is not unfounded for P with respect to Y. $\square$

We further describe the models of a program's reduct by unfounded sets.

Lemma 2. *Let P be a program, Y an interpretation such that $Y \models P$, and $U \subseteq \mathcal{U}$. Then, $(Y \setminus U) \models P^Y$ iff U is unfounded for P with respect to Y.*

Proof. ($\Rightarrow$) Assume that U is not unfounded for P with respect to Y. Then, for some rule $r \in P$, (α)–(δ) from the proof of Lemma 1 hold. Clearly, $B^-(r) \cap Y = \emptyset$ implies $(H(r) \leftarrow B^+(r)) \in P^Y$. From $B^+(r) \subseteq Y$ and (δ), we conclude $B^+(r) \subseteq (Y \setminus U)$. Together with ($\beta$), we obtain $(Y \setminus U) \not\models (H(r) \leftarrow B^+(r))$ and thus $(Y \setminus U) \not\models P^Y$.

($\Leftarrow$) Assume $(Y \setminus U) \not\models P^Y$. Then, there is a rule $r \in P$ such that $(Y \setminus U) \not\models \{r\}^Y$. We conclude that r satisfies (β), $B^+(r) \subseteq (Y \setminus U)$, and $B^-(r) \cap Y = \emptyset$. Since $B^+(r) \subseteq$

$(Y \setminus U)$ immediately implies $B^+(r) \subseteq Y$, (γ) holds. Moreover, $B^+(r) \subseteq (Y \setminus U)$ also implies (δ). It remains to show (α). From (γ) and $Y \models r$ (which holds by the assumption $Y \models P$), we conclude $H(r) \cap Y \neq \emptyset$. Together with ($\beta$), this implies ($\alpha$). Since ($\alpha$), ($\beta$), ($\gamma$), and ($\delta$) jointly hold for some rule $r \in P$, we have that U is not unfounded for P with respect to Y. □

Example 4. For illustration, reconsider P from Example 2 and $Y = \{b, c, d\}$. For singleton $\{a\}$ and r_1, we have $H(r_1) \cap (Y \setminus \{a\}) = \{a, b\} \cap \{b, c, d\} \neq \emptyset$. Furthermore, $a \notin H(r)$ for all $r \in \{r_2, \ldots, r_6\}$. That is, $\{a\}$ is unfounded for P with respect to Y. From this, we can conclude by Lemma 1 that Y is a model of P, i.e., $Y \models P$.

As we have already seen in Example 2, $U = \{c, d\}$ is unfounded for P with respect to Y. Lemma 2 now tells us that $(Y \setminus U) = \{b\}$ is a model of $P^Y = \big\{r_1, r_2, r_3, r_5, \big(H(r_6) \leftarrow B^+(r_6)\big)\big\}$. Moreover, one can check that $\{a, c, d\}$ is as well unfounded for P with respect to Y. ◊

The last observation in Example 4 stems from a more general side effect of Lemma 2: For any program P, any interpretation Y such that $Y \models P$, and $U \subseteq \mathcal{U}$, U is unfounded for P with respect to Y iff $(U \cap Y)$ is unfounded for P with respect to Y. For models Y, this allows us to restrict our attention to unfounded sets $U \subseteq Y$.

We are now in a position to state the following alternative characterizations of SE- and UE-models.

Theorem 1. *Let P be a program, Y an interpretation such that $Y \models P$, and $U \subseteq \mathcal{U}$. Then, $(Y \setminus U, Y)$ is an SE-model of P iff $(U \cap Y)$ is unfounded for P with respect to Y.*

Theorem 2. *Let P be a program, Y an interpretation such that $Y \models P$, and $U \subseteq \mathcal{U}$. Then, $(Y \setminus U, Y)$ is a UE-model of P iff $(U \cap Y)$ is unfounded for P with respect to Y and no nonempty proper subset of $(U \cap Y)$ is unfounded for P with respect to Y.*

Note that the inherent maximality criterion of UE-models is now reflected by a *minimality condition* on (nonempty) unfounded sets. Theorem 1 and 2 allow us to characterize strong and uniform equivalence in terms of unfounded sets, avoiding an explicit use of programs' reducts. Details will be discussed in Section 4.

Example 5. Recall programs $P = \{a \vee b \leftarrow\}$ and $Q = \{a \leftarrow not\, b;\ b \leftarrow not\, a\}$ from Example 1. We have seen that the only difference in their SE-models is the pair $(\emptyset, ab)$, which is an SE-model of Q, but not of P. Clearly, $Y = \{a, b\}$ is a classical model of P and of Q, and, in view of Theorem 1, we expect that Y is unfounded for Q with respect to Y, but not for P with respect to Y. The latter is easily checked since the rule $r = (a \vee b \leftarrow)$ yields (1) $H(r) \cap Y \neq \emptyset$; (2) $H(r) \cap (Y \setminus Y) = \emptyset$; (3) $B^+(r) \subseteq Y$ and $B^-(r) \cap Y = \emptyset$; and (4) $B^+(r) \cap Y = \emptyset$. Thus, none of the four unfoundedness conditions is met. However, for $r_1 = a \leftarrow not\, b$ and $r_2 = b \leftarrow not\, a$, we have $B^-(r_i) \cap Y \neq \emptyset$, for $i \in \{1, 2\}$, and thus Y is unfounded for Q with respect to Y.

Recall that $(\emptyset, ab)$ is not a UE-model of Q. In view of Theorem 2, we thus expect that $Y = \{a, b\}$ is not a minimal nonempty unfounded set. As one can check, both nonempty proper subsets $\{a\}$ and $\{b\}$ are in fact unfounded for Q with respect to Y. ◊

In the remainder of this section, we provide a further characterization of UE-models that makes use of elementary sets [5]. This not only gives us a more intrinsic characterization of the difference $U = (Y \setminus X)$ for a UE-model (X, Y) than that stated in Theorem 2, but also yields a further direct relation to loops. We make use of this fact and provide a new result for the UE-models of tight programs.

We define a nonempty set $U \subseteq \mathcal{U}$ as *elementary* for a program P if, for each V such that $\emptyset \subset V \subset U$, there is some $r \in P$ jointly satisfying

1. $H(r) \cap V \neq \emptyset$,
2. $H(r) \cap (U \setminus V) = \emptyset$,
3. $B^+(r) \cap V = \emptyset$, and
4. $B^+(r) \cap (U \setminus V) \neq \emptyset$.

Due to Conditions 1 and 4, every elementary set is also a loop of P, but the converse does not hold in general [5].

To link elementary sets and unfounded sets together, for a program P, an interpretation Y, and $U \subseteq \mathcal{U}$, we define:

$$P_{Y,U} = \{r \in P \mid H(r) \cap (Y \setminus U) = \emptyset,\ B^+(r) \subseteq Y,\ B^-(r) \cap Y = \emptyset\}.$$

Provided that $H(r) \cap U \neq \emptyset$, a rule $r \in P_{Y,U}$ supports U with respect to Y, while no rule in $(P \setminus P_{Y,U})$ supports U. Analogously to Gebser, Lee, and Lierler [5], we say that U is *elementarily unfounded* for P with respect to Y iff (i) U is unfounded for P with respect to Y and (ii) U is elementary for $P_{Y,U}$. Any elementarily unfounded set of P with respect to Y is also elementary for P, but an elementary set U that is unfounded for P with respect to Y is not necessarily elementarily unfounded because U might not be elementary for $P_{Y,U}$ [5].

Elementarily unfounded sets coincide with minimal nonempty unfounded sets.

Proposition 2 ([5]). *Let P be a program, Y an interpretation, and $U \subseteq \mathcal{U}$. Then, U is a minimal nonempty unfounded set of P with respect to Y iff U is elementarily unfounded for P with respect to Y.*

The fact that every nonempty unfounded set contains some elementarily unfounded set, which by definition is an elementary set, allows us to derive some properties of the difference $U = (Y \setminus X)$ for SE-interpretations (X, Y). For instance, we can exploit the fact that every elementary set is also a loop in the characterization of minimal nonempty unfounded sets, where the latter are only defined with respect to interpretations.

Formally, we derive the following properties for UE-models (resp., SE-models):

Corollary 1. *Let P be a program and (X, Y) a UE-model (resp., SE-model) of P. If $X \neq Y$, then $(Y \setminus X)$ is (resp., contains) (a) an elementarily unfounded set of P with respect to Y; (b) an elementary set of P; and (c) a loop of P.*

For tight programs, i.e., programs such that every loop is a singleton, we obtain the following property:

Corollary 2. *Let P be a tight program and (X, Y) an SE-model of P. Then, (X, Y) is a UE-model of P iff $X = Y$ or $(Y \setminus X)$ is a singleton that is unfounded for P with respect to Y.*

Example 6. Recall the SE-model $(\emptyset, ab)$ of $Q = \{a \leftarrow \mathit{not}\, b;\ b \leftarrow \mathit{not}\, a\}$. The loops of Q are $\{a\}$ and $\{b\}$; thus, Q is tight. This allows us to immediately conclude that $(\emptyset, ab)$ is not a UE-model of Q, without looking for any further SE-model to rebut it. $\Diamond$

The above result shows that, for tight programs, the structure of UE-models is particularly simple, viz. they are always of the form (Y, Y) or $(Y \setminus \{a\}, Y)$, for some $a \in Y$. As we will see in the next section, this also allows for simplified encodings.

4 Characterizations for Program Equivalence

In this section, we further exploit unfounded sets to characterize different notions of program equivalence. We start by comparing two programs, P and Q, regarding their unfounded sets for deriving conditions under which P and Q are ordinarily, strongly, and uniformly equivalent, respectively. Based on these conditions, we then provide novel encodings in propositional logic.

4.1 Characterizations based on Unfounded Sets

Two programs are ordinarily equivalent if they possess the same answer sets. As Proposition 1 shows, answer sets are precisely the models of a program that do not contain any nonempty unfounded set. Hence, ordinary equivalence can be described as follows:

Theorem 3. *Let P and Q be programs. Then, P and Q are ordinarily equivalent iff, for every interpretation Y, the following two conditions are equivalent*:

1. *$Y \models P$ and no nonempty subset of Y is unfounded for P with respect to Y;*
2. *$Y \models Q$ and no nonempty subset of Y is unfounded for Q with respect to Y.*

Note that ordinarily equivalent programs are not necessarily classically equivalent, as is for instance witnessed by programs $P = \{a \vee b \leftarrow\}$ and $Q = \{a \vee b \leftarrow;\ a \leftarrow c\}$ possessing the same answer sets: $\{a\}$ and $\{b\}$. However, $\{b, c\}$ is a model of P but not of Q. In turn, for strong and uniform equivalence, classical equivalence is a necessary (but, in general, not a sufficient) condition. This follows from the fact that every model of a program participates in at least one SE-model (resp., UE-model) and is thus relevant for testing strong (resp., uniform) equivalence. Indeed, the following characterization of strong equivalence considers all classical models.

Theorem 4. *Let P and Q be programs. Then, P and Q are strongly equivalent iff, for every interpretation Y such that $Y \models P$ or $Y \models Q$, P and Q possess the same unfounded sets with respect to Y.*

Proof. ($\Rightarrow$) Assume that P and Q are strongly equivalent. Fix any interpretation Y such that $Y \models P$ (or $Y \models Q$). Then, (Y, Y) is an SE-model of P (or Q), and since P and Q are strongly equivalent, (Y, Y) is also an SE-model of Q (or P). That is, both $Y \models P$ and $Y \models Q$ hold. Fix any set $U \subseteq \mathcal{U}$. By Lemma 2 and the fact that P and Q are strongly equivalent, U is unfounded for P with respect to Y iff $(Y \setminus U, Y)$ is an SE-model of P. But the latter holds iff $(Y \setminus U, Y)$ is an SE-model of Q, which in turn holds iff U is unfounded for Q with respect to Y.

($\Leftarrow$) Assume that P and Q are not strongly equivalent. Then, without loss of generality, there is an SE-model (X, Y) of P that is not an SE-model of Q (the other case is symmetric). By the definition of SE-models, we have $Y \models P$, and by Lemma 2, $(Y \setminus X)$ is unfounded for P with respect to Y, but either $Y \not\models Q$ or $(Y \setminus X)$ is not unfounded for Q with respect to Y. If $(Y \setminus X)$ is not unfounded for Q with respect to Y, then P and Q do not possess the same unfounded sets with respect to Y. Otherwise, if $Y \not\models Q$, by Lemma 1, there is a set $U \subseteq \mathcal{U} \setminus Y$ that is not unfounded for Q with respect to Y, but U is unfounded for P with respect to Y. □

Theorem 4 shows that strong equivalence focuses primarily on the unfounded sets admitted by the compared programs. In the setting of uniform equivalence, the consideration of unfounded sets is further restricted to minimal ones (cf. Theorem 2), and by Proposition 2, these are exactly the elementarily unfounded sets.

Theorem 5. *Let P and Q be programs. Then, P and Q are uniformly equivalent iff, for every interpretation Y such that $Y \models P$ or $Y \models Q$, P and Q possess the same elementarily unfounded sets with respect to Y.*

Proof. ($\Rightarrow$) Assume that P and Q are uniformly equivalent. Fix any interpretation Y such that $Y \models P$ (or $Y \models Q$). Then, (Y, Y) a UE-model of P (or Q), and since P and Q are uniformly equivalent, (Y, Y) is also a UE-model of Q (or P). That is, both $Y \models P$ and $Y \models Q$ hold. Fix any elementarily unfounded set U for P (or Q) with respect to Y. If $U \subseteq \mathcal{U} \setminus Y$, by Lemma 1 and Proposition 2, U is a singleton that is unfounded for both P and Q with respect to Y, which implies that U is elementarily unfounded for Q (or P) with respect to Y. Otherwise, if $U \cap Y \neq \emptyset$, then Lemma 1 and Proposition 2 imply $U \subseteq Y$. By Proposition 2 and Theorem 2, $(Y \setminus U, Y)$ is a UE-model of P (or Q), and since P and Q are uniformly equivalent, $(Y \setminus U, Y)$ is also a UE-model of Q (or P). Since $\emptyset \neq U \subseteq Y$, by Theorem 2 and Proposition 2, we conclude that U is elementarily unfounded for Q (or P) with respect to Y.

($\Leftarrow$) Assume that P and Q are not uniformly equivalent. Then, without loss of generality, there is a UE-model (X, Y) of P that is not a UE-model of Q (the other case is symmetric). Since (X, Y) is also an SE-model of P, we have $Y \models P$. If $Y \not\models Q$, by Lemma 1, there is a singleton $U \subseteq \mathcal{U} \setminus Y$ that is not unfounded for Q with respect to Y, but U is unfounded for P with respect to Y. That is, U is elementarily unfounded for P with respect to Y, but not for Q with respect to Y. Otherwise, if $Y \models Q$, (Y, Y) is a UE-model both of P and of Q. We conclude that $X \subset Y$, and by Theorem 2 and Proposition 2, $(Y \setminus X)$ is elementarily unfounded for P with respect to Y. Furthermore, the fact that (X, Y) is not a UE-model of Q, by Theorem 2 and Proposition 2, implies that $(Y \setminus X)$ is not elementarily unfounded for Q with respect to Y. □

In contrast to arbitrary unfounded sets, elementarily unfounded sets exhibit a certain structure as they are in fact loops or, even more accurately, elementary sets (cf. Corollary 1). By Theorem 5, such structures alone are material to uniform equivalence.

4.2 Characterizations in Propositional Logic

We now exploit the above results about unfounded sets to encode program equivalence in propositional logic. For ordinary equivalence, we use the well-known concept of loop formulas, while for strong and uniform equivalence we directly refer to unfounded sets.

In what follows, we write for a set of default literals, like $B(r)$, and a set of atoms, like $H(r)$, $B(r) \rightarrow H(r)$ as a shorthand for

$$\big(\textstyle\bigwedge_{a \in B^+(r)} a \wedge \bigwedge_{a \in B^-(r)} \neg a\big) \rightarrow \bigvee_{a \in H(r)} a,$$

where, as usual, empty conjunctions (resp., disjunctions) are understood as $\top$ (resp., $\bot$). For instance, for a rule r of the form (1), $B(r) \rightarrow H(r)$ yields the implication

$$a_{k+1} \wedge \cdots \wedge a_m \wedge \neg a_{m+1} \wedge \cdots \wedge \neg a_n \rightarrow a_1 \vee \cdots \vee a_k.$$

Furthermore, within the subsequent encodings, an occurrence of a program P is understood as $\bigwedge_{r \in P}(B(r) \rightarrow H(r))$.

As a basis for the encodings, we use the following concept. Following Lee [7], for a program P and $U \subseteq \mathcal{U}$, the *external support formula* of U for P is

$$ES_P(U) = \textstyle\bigvee_{r \in P, H(r) \cap U \neq \emptyset, B^+(r) \cap U = \emptyset} \neg\big(B(r) \rightarrow (H(r) \setminus U)\big). \qquad (2)$$

The relation between unfounded sets and external support formulas is as follows:

Lemma 3. *Let P be a program, Y an interpretation, and $U \subseteq \mathcal{U}$. Then, U is unfounded for P with respect to Y iff $Y \not\models ES_P(U)$.*

Proof. ($\Rightarrow$) Assume that $Y \models ES_P(U)$. Then, there is a rule $r \in P$ such that

(α) $H(r) \cap U \neq \emptyset$,
(β) $B^+(r) \cap U = \emptyset$,
(γ) $B^+(r) \subseteq Y$ and $B^-(r) \cap Y = \emptyset$, and
(δ) $(H(r) \setminus U) \cap Y = H(r) \cap (Y \setminus U) = \emptyset$.

That is, U is not unfounded for P with respect to Y.

($\Leftarrow$) Assume that U is not unfounded for P with respect to Y. Then, there is a rule $r \in P$ for which (α), (β), (γ), and (δ) hold. From (γ) and (δ), we conclude

$$Y \models \neg\big(B(r) \rightarrow (H(r) \setminus U)\big),$$

which together with (α) and (β) implies $Y \models ES_P(U)$. □

For a program P and $U \subseteq \mathcal{U}$, the (conjunctive) *loop formula* [7] of U for P is

$$LF_P(U) = \big(\textstyle\bigwedge_{p \in U} p\big) \rightarrow ES_P(U).$$

With respect to an interpretation Y, the loop formula of U is violated if Y contains U as an unfounded set, otherwise, the loop formula of U is satisfied.

Proposition 3 ([7, 5]). *Let P be a program and Y an interpretation such that $Y \models P$. Then, the following statements are equivalent:*

(*a*) *Y is an answer set of P;*
(*b*) *$Y \models LF_P(U)$ for every nonempty subset U of $\mathcal{U}$;*
(*c*) *$Y \models LF_P(U)$ for every loop U of P;*
(*d*) *$Y \models LF_P(U)$ for every elementary set U of P.*

For ordinary equivalence, the following encodings (as well as different combinations thereof) can thus be obtained.

Theorem 6. *Let P and Q be programs. Let $\mathcal{L}$ and $\mathcal{E}$ denote the set of all loops and elementary sets, respectively, of P and Q. Then, the following statements are equivalent:*

(*a*) *P and Q are ordinarily equivalent;*
(*b*) $\big(P \wedge \bigwedge_{\emptyset \neq U \subseteq \mathcal{U}} LF_P(U)\big) \leftrightarrow \big(Q \wedge \bigwedge_{\emptyset \neq U \subseteq \mathcal{U}} LF_Q(U)\big)$ *is a tautology;*
(*c*) $\big(P \wedge \bigwedge_{U \in \mathcal{L}} LF_P(U)\big) \leftrightarrow \big(Q \wedge \bigwedge_{U \in \mathcal{L}} LF_Q(U)\big)$ *is a tautology;*
(*d*) $\big(P \wedge \bigwedge_{U \in \mathcal{E}} LF_P(U)\big) \leftrightarrow \big(Q \wedge \bigwedge_{U \in \mathcal{E}} LF_Q(U)\big)$ *is a tautology.*

Recall that, for tight programs, each loop (and thus, each elementary set) is a singleton. In this case, the encodings in (*c*) and (*d*) are thus polynomial in the size of the compared programs. Moreover, one can verify that they amount to checking whether the completions [12] of the compared programs are equivalent in classical logic.

For strong and uniform equivalence between P and Q, the models of P and Q along with the corresponding unfounded sets are compared, as Theorem 4 and 5 show. We thus directly consider external support formulas, rather than loop formulas.

Theorem 4 and Lemma 3 yield the following encoding for strong equivalence:

Theorem 7. *Let P and Q be programs. Then, P and Q are strongly equivalent iff $\big(P \vee Q\big) \rightarrow \big(\bigwedge_{U \subseteq \mathcal{U}} \big(ES_P(U) \leftrightarrow ES_Q(U)\big)\big)$ is a tautology.*

In order to also encode uniform equivalence, we have to single out elementarily unfounded sets. To this end, we modify the definition of the external support formula, $ES_P(U)$, and further encode the case that U is (not) a minimal nonempty unfounded set. For a program P and $U \subseteq \mathcal{U}$, we define the *minimality external support* formula as

$$ES^\star_P(U) = ES_P(U) \vee \neg\big(\bigwedge_{\emptyset \subset V \subset U} ES_P(V)\big).$$

Similar to external support formulas and unfounded sets, minimality external support formulas correspond to elementarily unfounded sets as follows.

Lemma 4. *Let P be a program, Y an interpretation, and $\emptyset \subset U \subseteq \mathcal{U}$. Then, U is elementarily unfounded for P with respect to Y iff $Y \not\models ES^\star_P(U)$.*

Proof. ($\Rightarrow$) Assume that $Y \models ES^\star_P(U)$. Then, one of the following two cases holds:

1. $Y \models ES_P(U)$: By Lemma 3, U is not unfounded for P with respect to Y, which implies that U is not elementarily unfounded for P with respect to Y.
2. $Y \not\models \big(\bigwedge_{\emptyset \subset V \subset U} ES_P(V)\big)$: For some V such that $\emptyset \subset V \subset U$, we have $Y \not\models ES_P(V)$. By Lemma 3, V is unfounded for P with respect to Y. We conclude that U is not a minimal nonempty unfounded set of P with respect to Y, and by Proposition 2, U is not elementarily unfounded for P with respect to Y.

($\Leftarrow$) Assume that $Y \not\models ES^\star_P(U)$. Then, $Y \not\models ES_P(U)$, and by Lemma 3, U is unfounded for P with respect to Y. Furthermore, $Y \models \big(\bigwedge_{\emptyset \subset V \subset U} ES_P(V)\big)$, and thus no set V such that $\emptyset \subset V \subset U$ is unfounded for P with respect to Y (again by Lemma 3). We conclude that U is a minimal nonempty unfounded set of P with respect to Y, and by Proposition 2, U is elementarily unfounded for P with respect to Y. □

Theorem 5 and Lemma 4 allow us to encode uniform equivalence as follows.

Theorem 8. *Let P and Q be programs. Let $\mathcal{L}$ and $\mathcal{E}$ denote the set of all loops and elementary sets, respectively, of P and Q. Then, the following statements are equivalent:*

(a) *P and Q are uniformly equivalent;*
(b) $\big(P \vee Q\big) \rightarrow \big(\bigwedge_{U \subseteq \mathcal{U}} \big(ES^{\star}_{P}(U) \leftrightarrow ES^{\star}_{Q}(U)\big)\big)$ *is a tautology;*
(c) $\big(P \vee Q\big) \rightarrow \big(\bigwedge_{U \in \mathcal{L}} \big(ES^{\star}_{P}(U) \leftrightarrow ES^{\star}_{Q}(U)\big)\big)$ *is a tautology;*
(d) $\big(P \vee Q\big) \rightarrow \big(\bigwedge_{U \in \mathcal{E}} \big(ES^{\star}_{P}(U) \leftrightarrow ES^{\star}_{Q}(U)\big)\big)$ *is a tautology.*

Proof. By Theorem 5, P and Q are uniformly equivalent iff, for every interpretation Y such that $Y \models P$ or $Y \models Q$, P and Q possess the same elementarily unfounded sets with respect to Y. Clearly, any elementarily unfounded set of P or Q belongs to the set $\mathcal{E}$ of all elementary sets of P and Q, which is a subset of the set $\mathcal{L}$ of all loops of P and Q, and every element of $\mathcal{L}$ is a subset of $\mathcal{U}$. Furthermore, by Lemma 4, a set $\emptyset \subset U \subseteq \mathcal{U}$ is elementarily unfounded for P (resp., Q) with respect to Y iff $Y \not\models ES^{\star}_{P}(U)$ (resp., $Y \not\models ES^{\star}_{Q}(U)$). Finally, we have $ES^{\star}_{P}(\emptyset) \equiv ES^{\star}_{Q}(\emptyset) \equiv \bot$, so that $Y \models \big(ES^{\star}_{P}(\emptyset) \leftrightarrow ES^{\star}_{Q}(\emptyset)\big)$ for any interpretation Y. From this, the statement of Theorem 8 follows. □

Again, we exploit the fact that, for tight programs, all loops and elementary sets are singletons. It is thus sufficient to consider only the external support formulas of singletons. To the best of our knowledge, this provides a novel technique to decide uniform equivalence between tight programs. Indeed, the following result is an immediate consequence of (c), or likewise (d), in Theorem 8.

Corollary 3. *Let P and Q be tight programs. Then, P and Q are uniformly equivalent iff $\big(P \vee Q\big) \rightarrow \big(\bigwedge_{a \in \mathcal{U}} \big(ES_{P}(\{a\}) \leftrightarrow ES_{Q}(\{a\})\big)\big)$ is a tautology.*

Indeed, for singletons $\{a\}$, $\neg\big(\bigwedge_{\emptyset \subset V \subset \{a\}} ES_{P}(V)\big)$ (resp., $\neg\big(\bigwedge_{\emptyset \subset V \subset \{a\}} ES_{Q}(V)\big)$) can be dropped from $ES^{\star}_{P}(\{a\})$ (resp., $ES^{\star}_{Q}(\{a\})$) because it is equivalent to $\bot$.

Except for ordinary and uniform equivalence between tight programs, all of the above encodings are of exponential size. As with the known encodings for answer sets, reproduced in Proposition 3, we do not suggest to *a priori* reduce the problem of deciding program equivalence to propositional logic. Rather, our encodings provide an alternative view on the conditions underlying program equivalence; similar characterizations have already been successfully exploited in answer-set solving [6, 13].

For strong equivalence, however, we can resolve the exponential number of conjuncts in Theorem 7 as follows. We use a copy $\mathcal{U}' = \{p' \mid p \in \mathcal{U}\}$ of the universe $\mathcal{U}$, where all p' are mutually distinct new atoms, and introduce a module representing $ES_{P}(U)$, as given in (2), but without explicitly referring to certain sets U; rather, a particular U is determined by the true atoms from the copy $\mathcal{U}'$ of $\mathcal{U}$. We define:

$$ES_{P} = \bigvee_{r \in P}\big(\bigvee_{p \in H(r)} p' \wedge \bigwedge_{p \in H(r)} (p' \vee \neg p) \wedge \bigwedge_{p \in B^{+}(r)} (p \wedge \neg p') \wedge \bigwedge_{p \in B^{-}(r)} \neg p\big).$$

Given a program P, for an interpretation Y (over $\mathcal{U}$) and $U \subseteq \mathcal{U}$, U is unfounded for P with respect to Y iff $(Y \cup \{p' \mid p \in U\}) \not\models ES_{P}$. This yields the following result:

Theorem 9. *Let P and Q be programs. Then, P and Q are strongly equivalent iff $(P \vee Q) \rightarrow (ES_{P} \leftrightarrow ES_{Q})$ is a tautology.*

5 Discussion

We provided novel characterizations for program equivalence in terms of unfounded sets, along with the related notions of loops and elementary sets. This allowed us to identify close relationships between these important concepts. While answer sets, and thus ordinary equivalence, rely on the absence of (nonempty) unfounded sets, we have shown that potential extensions of programs, captured by SE- and UE-models, can also be characterized directly by appeal to unfounded sets, thereby avoiding any reference to reducts of programs. We have seen that uniform equivalence is located in between ordinary and strong equivalence, in the sense that it considers all models, similar to strong equivalence, but only minimal (nonempty) unfounded sets, which are sufficient to decide whether a model is an answer set. This allowed us to develop particularly simple characterizations for uniform equivalence between tight programs.

References

1. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
2. Turner, H.: Strong equivalence made easy: Nested expressions and weight constraints. Theory and Practice of Logic Programming **3**(4-5) (2003) 602–622
3. Eiter, T., Fink, M.: Uniform equivalence of logic programs under the stable model semantics. In Palamidessi, C., ed.: Proceedings of the 19th International Conference on Logic Programming (ICLP'03), Springer-Verlag (2003) 224–238
4. Van Gelder, A., Ross, K., Schlipf, J.: The well-founded semantics for general logic programs. Journal of the ACM **38**(3) (1991) 620–650
5. Gebser, M., Lee, J., Lierler, Y.: Elementary sets for logic programs. In: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI'06), AAAI Press (2006)
6. Lin, F., Zhao, Y.: ASSAT: Computing answer sets of a logic program by SAT solvers. Artificial Intelligence **157**(1-2) (2004) 115–137
7. Lee, J.: A model-theoretic counterpart of loop formulas. In Kaelbling, L., Saffiotti, A., eds.: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05), Professional Book Center (2005) 503–508
8. Fages, F.: Consistency of Clark's completion and the existence of stable models. Journal of Methods of Logic in Computer Science **1** (1994) 51–60
9. Erdem, E., Lifschitz, V.: Tight logic programs. Theory and Practice of Logic Programming **3**(4-5) (2003) 499–518
10. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Transactions on Computational Logic **2**(4) (2001) 526–541
11. Leone, N., Rullo, P., Scarcello, F.: Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation. Information and Computation **135**(2) (1997) 69–112
12. Clark, K.: Negation as failure. In Gallaire, H., Minker, J., eds.: Logic and Data Bases. Plenum Press (1978) 293–322
13. Giunchiglia, E., Lierler, Y., Maratea, M.: Answer set programming based on propositional satisfiability. Journal of Automated Reasoning **36**(4) (2006) 345–377

Relativised Equivalence in Equilibrium Logic and its Applications to Prediction and Explanation: Preliminary Report[⋆]

David Pearce[1], Hans Tompits[2], and Stefan Woltran[2]

[1] Universidad Rey Juan Carlos, Computing Science and Artificial Intelligence,
Calle Tulipan s/n, E-28933 Madrid, Spain
Email: `davidandrew.pearce@urjc.es`
[2] Technische Universität Wien, Institut für Informationssysteme 184/3,
Favoritenstrasse 9-11, A-1040 Vienna, Austria
Email: {`tompits,stefan`}`@kr.tuwien.ac.at`

Abstract. For a given semantics, two nonmonotonic theories Π_1 and Π_2 can be said to be equivalent if they have the same intended models and strongly (resp., uniformly) equivalent if for any Σ, $\Pi_1 \cup \Sigma$ and $\Pi_2 \cup \Sigma$ are equivalent, where Σ is a set of sentences (resp., literals). In the general case, no restrictions are placed on the language (signature) of Σ. Relativised notions of strong and uniform equivalence are obtained by requiring that Σ belongs to a specified sublanguage $\mathcal{L}$ of the theories Π_1 and Π_2. For normal and disjunctive logic programs under stable-model semantics, relativised strong and uniform equivalence have been defined and characterised in previous work by Woltran. Here, we extend these concepts to nonmonotonic theories in equilibrium logic and discuss applications in the context of prediction and explanation.

1 Introduction

Equilibrium logic [12] is a general purpose formalism for nonmonotonic reasoning extending the stable-model and answer-set semantics for all the usual classes of logic programs, adhering to the general *answer-set programming* (ASP) paradigm. It is a form of minimal-model reasoning in the non-classical logic of *here-and-there*, which is basically intuitionistic logic restricted to two worlds, “here” and “there”, and subsumes all important syntactic extensions considered in ASP, including the addition of strong negation, rules with negation-by-default in their heads, and nested programs, as well as those constructs like cardinality and weight constraints and aggregates that have equivalent representations in the more general syntax of equilibrium logic [4, 5].

Recent research in ASP focuses on advanced notions of program equivalence relevant for program optimisation and modular programming [11, 1, 14]. A traditional concept of equivalence, where two nonmonotonic theories, under a given semantics, are viewed as being equivalent if they have the same intended models, is not adequate for

[⋆] This work was partially supported by the Spanish Ministry of Education and Science (MEC) under projects TIC-2003-9001-C02 and TIN2006-15455-CO3, and by the Austrian Science Fund (FWF) under grant P18019.

these purposes because such a notion does not satisfy a replacement property like in classical logic. Better candidates, however, are strong and uniform equivalence. While the former meets a replacement principle by definition, the latter is suitable for hierarchically ordered modules. In formal terms, two nonmonotonic theories, Π_1 and Π_2, are strongly (resp., uniformly) equivalent if for any Σ, $\Pi_1 \cup \Sigma$ and $\Pi_2 \cup \Sigma$ are equivalent, where Σ is a set of sentences (resp., literals). In the general case, no restrictions are placed on the language (signature) of Σ. *Relativised notions* of strong and uniform equivalence are obtained by requiring that Σ belongs to a specified sublanguage $\mathcal{L}$ of the theories Π_1 and Π_2. For normal and disjunctive logic programs under stable-model semantics, relativised strong and uniform equivalence have been defined and characterised in previous work by Woltran [19], together with a discussion about complexity issues and implementation strategies. Furthermore, relativised strong and uniform equivalence are special cases of *update equivalence* introduced by Inoue and Sakama [7].

In this paper, we extend the work of Woltran [19] and Pearce and Valverde [14] by characterising relative notions of equivalence for arbitrary (propositional) theories in equilibrium logic. Furthermore, we discuss how relativised equivalences can be applied to certain problems from the areas of diagnosis and abduction, with respect to the problem of deciding whether two logical descriptions have the same explanatory power, and provide a semantical characterisation of this problem. The formal model of an abductive explanation our discussion is based is an extension of a corresponding concept used by Inoue and Sakama [8] for disjunctive logic programs with default negation in their heads. Finally, we address the computational complexity of relative equivalence in equilibrium logic, showing that it remains on the same level as for logic programs.

2 Equilibrium Logic

We work in the nonclassical logic of here-and-there with strong negation $\mathbf{N}_5$ and its nonmonotonic extension, equilibrium logic [12], which generalises the answer-set semantics for logic programs to arbitrary propositional theories [11]. For more details, the reader is referred to [12, 13] and the logic texts cited below.

Formulas of $\mathbf{N}_5$ are built-up in the usual way using the logical constants $\wedge$, $\vee$, $\rightarrow$, $\neg$, $\sim$, standing respectively for conjunction, disjunction, implication, weak (or intuitionistic) negation, and strong negation. The axioms and rules of inference for $\mathbf{N}_5$ include those of intuitionistic logic (see, e.g., [16]) and the strong negation axioms from the calculus of Vorob'ev [17, 18]; for details, see [13].

The model theory of $\mathbf{N}_5$ is based on the usual Kripke semantics for Nelson's constructive logic $\mathbf{N}$ (see, e.g., [6, 16]), but $\mathbf{N}_5$ is complete for Kripke frames $\mathcal{F} = \langle W, \leq \rangle$ (where as usual W is the set of *points* or *worlds* and $\leq$ is a partial-ordering on W) having exactly two worlds, say h ("here") and t ("there") with $h \leq t$. As usual, a *model* is a frame together with an assignment i that associates to each element of W a set of *literals*[1] such that if $w \leq w'$ then $i(w) \subseteq i(w')$. An assignment is then extended inductively to all formulas via the usual rules for conjunction, disjunction, implication and (weak) negation in intuitionistic logic together with the following rules governing

[1] We use the term "literal" to denote an atom, or an atom prefixed by strong negation.

strongly negated formulas:

$$\begin{array}{l}
{\sim}(\varphi \wedge \psi) \in i(w) \text{ iff } {\sim}\varphi \in i(w) \text{ or } {\sim}\psi \in i(w);\\
{\sim}(\varphi \vee \psi) \in i(w) \text{ iff } {\sim}\varphi \in i(w) \text{ and } {\sim}\psi \in i(w);\\
{\sim}(\varphi \rightarrow \psi) \in i(w) \text{ iff } \varphi \in i(w); \text{ and } {\sim}\psi \in i(w);\\
{\sim}\neg\varphi \in i(w) \text{ iff } {\sim}{\sim}\varphi \in i(w) \text{ iff } \varphi \in i(w).
\end{array}$$

It is convenient to represent an $\mathbf{N}_5$-model as an ordered pair $\langle H, T\rangle$ of sets of literals, where $H = i(h)$ and $T = i(t)$ under a suitable assignment i. By $h \leq t$ it follows that $H \subseteq T$. Again, by extending i inductively we know what it means for an arbitrary formula φ to be true in a model $\mathcal{M} = \langle H, T\rangle$. We write $\mathcal{M}, w \models \varphi$ to express that φ is true at world w in model $\mathcal{M}$.

A formula φ is true in a here-and-there model $\mathcal{M} = \langle H, T\rangle$, in symbols $\mathcal{M} \models \varphi$, if it is true at each world in $\mathcal{M}$. A formula φ is said to be *valid* in $\mathbf{N}_5$, in symbols $\models \varphi$, if it is true in all here-and-there models. Logical consequence for $\mathbf{N}_5$ is understood as follows: φ is said to be an $\mathbf{N}_5$-*consequence* of a set Π of formulas, written $\Pi \models \varphi$, iff for all models $\mathcal{M}$ and any world $w \in \mathcal{M}$, $\mathcal{M}, w \models \Pi$ implies $\mathcal{M}, w \models \varphi$. Equivalently, this can be expressed by saying that φ is true in all models of Π. Further properties of $\mathbf{N}_5$ are studied in [10].

Equilibrium models are special kinds of minimal $\mathbf{N}_5$ Kripke models. We first define a partial ordering $\trianglelefteq$ on $\mathbf{N}_5$ models that will be used both to characterise the equilibrium property as well as the property of uniform equivalence.

Definition 1. *Given any two models $\langle H, T\rangle$, $\langle H', T'\rangle$, we set $\langle H, T\rangle \trianglelefteq \langle H', T'\rangle$ if $T = T'$ and $H \subseteq H'$.*

Definition 2. *Let Π be a set of $\mathbf{N}_5$ formulas and $\langle H, T\rangle$ a model of Π.*

1. *$\langle H, T\rangle$ is said to be* total *if $H = T$ (otherwise, if $H \subset T$, it is* non-total).
2. *$\langle H, T\rangle$ is said to be an* equilibrium *model if it is total and minimal under $\trianglelefteq$ among models of Π.*

In other words, a model $\langle H, T\rangle$ of Π is in equilibrium if it is total and there is no model $\langle H', T\rangle$ of Π with $H' \subset H$. Equilibrium logic is the logic determined by the equilibrium models of a theory. It generalises answer-set semantics in the following sense: For all the usual classes of logic programs, including normal, extended, disjunctive and nested programs, equilibrium models correspond to answer sets [12, 11]. The "translation" from the syntax of programs to $\mathbf{N}_5$ propositional formulas is the trivial one, viz., a ground rule of an (extended) disjunctive program of the form

$$K_1 \vee \ldots \vee K_k \leftarrow L_1, \ldots L_m, notL_{m+1}, \ldots, notL_n,$$

where the L_i and K_j are literals, corresponds to the $\mathbf{N}_5$ sentence

$$L_1 \wedge \ldots \wedge L_m \wedge \neg L_{m+1} \wedge \ldots \wedge \neg L_n \rightarrow K_1 \vee \ldots \vee K_k.$$

A set of $\mathbf{N}_5$ sentences is called a *theory*. Two theories are *equivalent* if they have the same equilibrium models.

3 Relativised Equivalence Concepts

We consider theories Π_1, Π_2, etc., and languages $\mathcal{L}$, $\mathcal{L}'$, etc. It will be convenient notationally viewing a language as a set of literals. A theory is said to be *in* the language $\mathcal{L}$ if all its atomic formulas belong to $\mathcal{L}$.

Definition 3. *Let Π_1 and Π_2 be theories.*

(*i*) *Π_1 and Π_2 are* strongly equivalent relative to *$\mathcal{L}$ iff for any* (*empty or non-empty*) *set Σ of $\mathcal{L}$ formulas, $\Pi_1 \cup \Sigma$ and $\Pi_2 \cup \Sigma$ are equivalent, i.e., have the same equilibrium models.*
(*ii*) *Π_1 and Π_2 are* uniformly equivalent relative to *$\mathcal{L}$ iff for any* (*empty or non-empty*) *set X of $\mathcal{L}$ literals, $\Pi_1 \cup X$ and $\Pi_2 \cup X$ are equivalent, i.e., have the same equilibrium models.*

Note that if the theories are logic programs, this means they have the same answer sets.

We explain some terminology and notation. A model $\langle H, T\rangle$ of a theory Π is said to be *maximally non-total* (or just *maximal*) if it is non-total and is maximal among models of Π under the ordering $\trianglelefteq$. In other words, a model $\langle H, T\rangle$ of Π is maximal if for any model $\langle H', T\rangle$ of Π, if $H \subset H'$ then $H' = T$. It is clear that if a theory Π is finite and has a non-total model $\langle H, T\rangle$, then it has a maximally non-total model $\langle H', T\rangle$ such that $H \subseteq H'$. However, maximal models need not exist in case that Π is an infinite theory. In what follows, we shall assume that all theories are finite.

Let $\mathcal{L}$ be a sublanguage of $\mathcal{L}'$. If $\mathcal{M} = \langle H, T\rangle$ is an $\mathcal{L}'$ model, its *$\mathcal{L}$-1-reduct* is defined by

$$\langle H \cap \mathcal{L}, T\rangle$$

and denoted by $\mathcal{M}|\mathcal{L}$. The term “1-reduct” stems from the fact that it refers to the first component of the model.

4 Characterising Relative Equivalence

For logic programs, the above relativised notions of equivalence are characterised by Woltran [19] in terms of what are called *relativised strong* (resp., *uniform*) *equivalence models*, or *RSE* (resp., *RUE*) *models* for short. We start by re-expressing these concepts in terms of ordinary models in the logic $\mathbf{N}_5$.

Definition 4. *Let Π be a theory in $\mathcal{L}'$ and $\mathcal{L}$ a sublanguage of $\mathcal{L}'$. A model $\mathcal{M} = \langle H, T\rangle$ is an* RSE$_\mathcal{L}$-model *of Π if it meets the following criteria:*

4.1 $\mathcal{M}$ is a total model of Π or
4.2 $\mathcal{M}$ is the $\mathcal{L}$-1-reduct of a non-total model $\langle H', T\rangle$ of Π, and
4.3 for any non-total model $\langle J, T\rangle$ of Π, $T \setminus J \cap \mathcal{L} \neq \emptyset$.

In other words, 4.3 holds together with one of 4.1 or 4.2. It is easy to see that for disjunctive logic programs, the above concept coincides with that of an RSE-model as defined by Woltran [19]. Indeed, we must check a preliminary condition and Conditions (i)-(iii) of Definition 6 by Woltran [19]. Clearly, both 4.1 and 4.2 above imply that T is a classical model of Π as required by (i). Condition 4.3 above re-expresses Clause (ii), while Condition 4.2 re-expresses Clause (iii). Finally, we check the preliminary condition of Woltran [19]. By 4.2, if $H \neq T$ then $\langle H, T\rangle$ is the reduct of a non-total model $\langle H', T\rangle$ of Π, so $H' \subset T$. Therefore, $H' \cap \mathcal{L} \subseteq T \cap \mathcal{L}$. But by 4.3, $H' \cap \mathcal{L} \neq T \cap \mathcal{L}$. Since $H = H' \cap \mathcal{L}$, it follows that $H \subset T \cap \mathcal{L}$ as required by the original definition of an RSE-model.

Now the following lemma is straightforward but useful. It says that two models with the same $\mathcal{L}$-1-reduct satisfy the same $\mathcal{L}$-sentences.

Lemma 1. *Let $\mathcal{M}$ and $\mathcal{M}'$ be $\mathbf{N}_5$ models and φ a formula all of whose atoms belong to the language $\mathcal{L}$. If $\mathcal{M}|\mathcal{L} = \mathcal{M}'|\mathcal{L}$, then $\mathcal{M} \models \varphi$ iff $\mathcal{M}' \models \varphi$.*

4.1 Relativised Strong Equivalence

Relativised strong equivalence (RSE) is defined as Woltran [19] does but for arbitrary theories. We can now show that sameness of RSE-models is a sufficient condition to ensure RSE.

Theorem 1. *Let Π_1 and Π_2 be theories having the same RSE$_{\mathcal{L}}$-models. Then, Π_1 and Π_2 are strongly equivalent relative to $\mathcal{L}$.*

Proof. Assume the hypothesis of the theorem and consider the theory $\Pi_1 \cup \Sigma$ where Σ is any set of sentences in $\mathcal{L}$. Consider any equilibrium model $\mathcal{M} = \langle T, T\rangle$ of $\Pi_1 \cup \Sigma$. We shall show that $\mathcal{M}$ is also an equilibrium model of $\Pi_2 \cup \Sigma$. By the symmetry of the situation, the same argument will show that any equilibrium model of $\Pi_2 \cup \Sigma$ must be an equilibrium model of $\Pi_1 \cup \Sigma$.

We first show that $\mathcal{M}$ is an RSE$_{\mathcal{L}}$-model of Π_1. Evidently, it is a total model of Π_1, so Condition 4.1 holds. Suppose that Condition 4.3 fails, so that there is a model $\langle J, T\rangle$ of Π_1 with $J \subset T$ such that $T \cap \mathcal{L} = J \cap \mathcal{L}$. Since $\langle T, T\rangle \models \Sigma$, by Lemma 1, $\langle J, T\rangle \models \Sigma$, but this contradicts the assumption that $\langle T, T\rangle$ is an equilibrium model of $\Pi_1 \cup \Sigma$. So Condition 4.3 applies and $\mathcal{M}$ is an RSE$_{\mathcal{L}}$-model of Π_1 and hence by assumption of Π_2. Therefore

$$\langle T, T\rangle \models \Pi_2 \cup \Sigma.$$

We need to show that it is in equilibrium. Note that since $\langle T, T\rangle$ is an RSE$_{\mathcal{L}}$-model of Π_2, by Condition 4.3 there is no model $\langle J, T\rangle$ of Π_2 with $J \subset T$ such that $T \cap \mathcal{L} = J \cap \mathcal{L}$. Suppose that $\mathcal{M}$ is not an equilibrium model of $\Pi_2 \cup \Sigma$. Then $\Pi_2 \cup \Sigma$ has a model $\langle H, T\rangle$ with $H \subset T$, so in particular $\langle H, T\rangle \models \Pi_2$ and by 4.3, $T \backslash H \cap \mathcal{L} \neq \emptyset$. So, $H \cap \mathcal{L} \subset T \cap \mathcal{L} \subseteq T$. It follows that $\langle H \cap \mathcal{L}, T\rangle$ is the $\mathcal{L}$-1-reduct of a model $\langle H, T\rangle \models \Pi_2$, with $H \subset T$. By Condition 4.2, $\langle H \cap \mathcal{L}, T\rangle$ is therefore an RSE$_{\mathcal{L}}$-model of Π_2, hence of Π_1. So, again by 4.2, it is the $\mathcal{L}$-1-reduct of some model $\langle H', T\rangle$ of Π_1 with $H' \subset T$ such that $H' \cap \mathcal{L} = H \cap \mathcal{L}$. By Lemma 1, since $\langle H, T\rangle \models \Sigma$ also $\langle H', T\rangle \models \Sigma$ and hence $\langle H', T\rangle \models \Pi_1 \cup \Sigma$. But this contradicts the assumption that $\langle T, T\rangle$ is an equilibrium model of $\Pi_1 \cup \Sigma$. Therefore, $\langle T, T\rangle$ is an equilibrium model of $\Pi_2 \cup \Sigma$. ◻

We now tackle the converse of Theorem 1.

Theorem 2. *Let Π_1 and Π_2 be theories such that Π_1 and Π_2 are strongly equivalent relative to $\mathcal{L}$. Then, they have the same RSE$_{\mathcal{L}}$-models.*

Proof. Suppose that Π_1 and Π_2 have different RSE$_{\mathcal{L}}$-models. We shall define a set of $\mathcal{L}$-sentences Σ such that $\Pi_1 \cup \Sigma$ and $\Pi_2 \cup \Sigma$ have different equilibrium models. Without loss of generalisation, assume there is an $\mathcal{M}$ which is an RSE$_{\mathcal{L}}$-model of Π_1 but not of Π_2. We consider several cases and subcases.

CASE 1. $\mathcal{M} = \langle T, T\rangle$ is a total RSE$_{\mathcal{L}}$-model of Π_1 that is not an RSE$_{\mathcal{L}}$-model of Π_2. Set $\Sigma = T \cap \mathcal{L}$. Then clearly $\mathcal{M} \models \Pi_1 \cup \Sigma$. Moreover, $\mathcal{M}$ is an equilibrium model of $\Pi_1 \cup \Sigma$. For, if not, there is a model $\langle H, T\rangle$ of $\Pi_1 \cup \Sigma$ with $H \subset T$. Since $\Sigma = T \cap \mathcal{L}$, we must have $T \cap \mathcal{L} \subseteq H$. But then $T \cap \mathcal{L} = H \cap \mathcal{L}$, which contradicts Condition 4.3 for $\mathcal{M}$ being an RSE$_{\mathcal{L}}$-model of Π_1. There are two reasons why $\mathcal{M}$ is not an RSE$_{\mathcal{L}}$-model of Π_2.

SUBCASE 1.1. $\mathcal{M} \not\models \Pi_2$. In this case, since $\mathcal{M} \not\models \Pi_2$, it cannot be an equilibrium model of $\Pi_2 \cup \Sigma$.

SUBCASE 1.2. $\mathcal{M} \models \Pi_2$, but Condition 4.3 fails for Π_2. So, there is a model $\langle J, T\rangle$ of Π_2 with $J \subset T$ such that $T \cap \mathcal{L} = J \cap \mathcal{L}$. Applying Lemma 1, we conclude that $\langle J, T\rangle \models \Sigma$ since $\langle T, T\rangle \models \Sigma$. Therefore, $\langle J, T\rangle \models \Pi_2 \cup \Sigma$, so $\mathcal{M}$ is not an equilibrium model of $\Pi_2 \cup \Sigma$.

CASE 2. $\mathcal{M} = \langle H, T\rangle$ is a non-total RSE$_{\mathcal{L}}$-model of Π_1 that is not an RSE$_{\mathcal{L}}$-model of Π_2. Observe that $\langle T, T\rangle$ is a total RSE$_{\mathcal{L}}$-model of Π_1. Hence, in case $\langle T, T\rangle$ is not an RSE$_{\mathcal{L}}$-model of Π_2, we can apply the same argument of Case 1 to conclude that $\langle T, T\rangle$ is an equilibrium model of $\Pi_1 \cup \Sigma$ and, again, cannot be an equilibrium model of $\Pi_2 \cup \Sigma$.

So suppose $\langle T, T\rangle$ is an RSE$_{\mathcal{L}}$-model of Π_2 and Condition 4.2 fails for $\mathcal{M} = \langle H, T\rangle$, i.e., there is no non-total model of Π_2 whose $\mathcal{L}$-1-reduct equals $\mathcal{M}$. Let $\Gamma = \{A \rightarrow B \mid A, B \in (T \backslash H) \cap \mathcal{L}\}$. By Condition 4.3, Γ is non-empty. Set $\Sigma = H \cup \Gamma$. Now, evidently $\langle T, T\rangle$ is a model of both H, since $H \subseteq T$, and of Γ, so $\langle T, T\rangle \models \Pi_2 \cup \Sigma$. We claim it is an equilibrium model of $\Pi_2 \cup \Sigma$, For, if not, there is a model $\langle J, T\rangle$ of $\Pi_2 \cup \Sigma$ with $J \subset T$. Clearly, $H \subseteq J$, but $H \neq J \cap \mathcal{L}$, otherwise $\langle J \cap \mathcal{L}, T\rangle = \mathcal{M}$ would be an RSE$_{\mathcal{L}}$-model of Π_2. So, $H \subset J \cap \mathcal{L}$. Thus, $(J \cap \mathcal{L}) \backslash H$ is non-empty, and by Condition 4.3, $(T \backslash J) \cap \mathcal{L}$ is also non-empty. Choose an A from $(J \cap \mathcal{L}) \backslash H$ and B from $(T \backslash J) \cap \mathcal{L}$. Then, $A \rightarrow B \in \Gamma$, but $\langle J, T\rangle \not\models A \rightarrow B$, since $\langle J, T\rangle, h \models A$ but $\langle J, T\rangle, h \not\models B$. It follows that $\langle J, T\rangle \not\models \Sigma$ and so $\langle T, T\rangle$ is an equilibrium model of $\Pi_2 \cup \Sigma$. On the other hand, it is not an equilibrium model of $\Pi_1 \cup \Sigma$. In particular, we know that $\langle H', T\rangle \models \Pi_1 \cup H$, since there is a non-total model $\langle H', T\rangle$ of Π_1 whose $\mathcal{L}$-1-reduct equals $\mathcal{M}$. Moreover, $\langle H', T\rangle \models \Gamma$ since $\langle H', T\rangle, h \not\models A$ for each $A \rightarrow B \in \Gamma$ and $\langle H', T\rangle, t \models B$ for each $A \rightarrow B \in \Gamma$. □

4.2 Relativised Uniform Equivalence

We now turn to the characterisation of relativised uniform equivalence via the concept of a relativised uniform equivalence model. First, we mention the following lemma that will be useful later.

Lemma 2. *Suppose Π_1 and Π_2 are theories which are uniformly equivalent relative to $\mathcal{L}$. Then, they have same total RSE$_{\mathcal{L}}$-models.*

Proof. Assume the hypothesis. Suppose Π_1 has a total RSE$_{\mathcal{L}}$-model $\langle T, T\rangle$ that is not a total RSE$_{\mathcal{L}}$-model of Π_2. Evidently, $\langle T, T\rangle \models \Pi_1 \cup (T \cap \mathcal{L})$. Moreover, by Condition 4.3, $\langle T, T\rangle$ must be an equilibrium model of $\Pi_1 \cup (T \cap \mathcal{L})$ since there is no model $\langle J, T\rangle$ of Π_1 with $J \subset T$ such that $T \cap \mathcal{L} \subseteq J \cap \mathcal{L}$. Clearly, if $\langle T, T\rangle \not\models \Pi_2$, it cannot be an equilibrium model of $\Pi_2 \cup (T \cap \mathcal{L})$. On the other hand, if $\langle T, T\rangle \models \Pi_2$ and it is not an RSE$_{\mathcal{L}}$-model of Π_2, then Condition 4.3 fails for Π_2. So, there is a model $\langle J, T\rangle$ of Π_2 with $J \subset T$ such that $T \cap \mathcal{L} = J \cap \mathcal{L}$, whence clearly $\langle T, T\rangle$ is not in equilibrium for $\Pi_2 \cup (T \cap \mathcal{L})$. This contradicts the assumption of relativised uniform equivalence. □

From now on we assume that all theories are finite. As mentioned previously, this means that, under the $\trianglelefteq$-ordering among their models, maximal elements are guaranteed to exist. So, the following notion is well-defined.

Definition 5. *Let Π be a theory in $\mathcal{L}'$ and $\mathcal{L}$ a sublanguage of $\mathcal{L}'$. An RSE$_{\mathcal{L}}$-model of Π is an* RUE$_{\mathcal{L}}$-model *of Π if it is either total or maximal under $\trianglelefteq$ among all non-total RSE$_{\mathcal{L}}$-models of Π.*

Theorem 3. *Let Π_1 and Π_2 be theories which are uniformly equivalent relative to $\mathcal{L}$. Then, they have the same RUE$_{\mathcal{L}}$-models.*

Proof. Assume the hypothesis. By Lemma 2, the two theories have the same total RSE$_{\mathcal{L}}$-models, hence total RUE$_{\mathcal{L}}$-models. Suppose that they differ on non-total RUE$_{\mathcal{L}}$-models, say that Π_1 has a non-total RUE$_{\mathcal{L}}$-model $\langle H, T\rangle$ that is not an RUE$_{\mathcal{L}}$-model of Π_2.

CASE 1. Suppose there is a non-total RSE$_{\mathcal{L}}$-model $\langle J, T\rangle$ of Π_2 with $H \subset J$. So, Π_2 has a non-total model $\langle H', T\rangle$ with $H' \cap \mathcal{L} = J$. Choose an element A from $J \backslash H$ and set $X = H \cup \{A\}$. Clearly, $\langle T, T\rangle \models \Pi_1 \cup X$ and by maximality, $\langle T, T\rangle$ is an equilibrium model of $\Pi_1 \cup X$. On the other hand, by inspection, $\langle H', T\rangle$ is a non-total model of $\Pi_2 \cup X$, so $\langle T, T\rangle$ is not an equilibrium model of $\Pi_2 \cup X$.

CASE 2. Suppose there is no non-total RSE$_{\mathcal{L}}$-model $\langle J, T\rangle$ of Π_2 with $H \subset J$. Since $\langle H, T\rangle$ is not an RUE$_{\mathcal{L}}$-model of Π_2, it cannot be an RSE$_{\mathcal{L}}$-model of Π_2 as well. Consider the model $\langle T, T\rangle$. Since Condition 4.3 holds for Π_1, clearly $\langle T, T\rangle$ is an RSE$_{\mathcal{L}}$-model of Π_1, and hence by Lemma 2 an RSE$_{\mathcal{L}}$-model of Π_2. So, $\langle T, T\rangle \models \Pi_2 \cup H$. Since there is no $H_2 \supseteq H$ such that $H_2 \subset T$ and $\langle H_2, T\rangle \models \Pi_2$, $\langle T, T\rangle$ is an equilibrium model of $\Pi_2 \cup H$. On the other hand, $\langle T, T\rangle$ is not an equilibrium model of $\Pi_2 \cup H$ since $\langle H', T\rangle \models \Pi_1 \cup H$, for some $H' \cap \mathcal{L} = H$. □

Theorem 4. *Suppose that Π_1 and Π_2 are theories with the same RUE$_{\mathcal{L}}$-models. Then, they are uniformly equivalent relative to $\mathcal{L}$.*

Proof. Assume the hypothesis and suppose that for some set X of $\mathcal{L}$ atoms, $\Pi_1 \cup X$ has an equilibrium model $\langle T, T\rangle$ that is not an equilibrium model of $\Pi_2 \cup X$. Clearly, $\langle T, T\rangle$ is a total RUE$_{\mathcal{L}}$-model of Π_1 and so, by assumption, also of Π_2. Therefore, $\langle T, T\rangle \models$

Π_2. Since it is not an equilibrium model of $\Pi_2 \cup X$, there is a model $\langle H, T\rangle \models \Pi_2 \cup X$ with $H \subset T$ and clearly $X \subseteq H$. Then, $\langle H \cap \mathcal{L}, T\rangle$ is an RSE$_\mathcal{L}$-model of Π_2. Keeping T fixed, we extend this to a maximal non-total RSE$_\mathcal{L}$-model $\langle H_2, T\rangle$ of Π_2, where $H \subseteq H_2$. Then, there is a model $\langle H', T\rangle$ of Π_2 such that

$$H' \cap \mathcal{L} = H_2 \supseteq H \cap \mathcal{L}.$$

Evidently, $\langle H_2, T\rangle$ is an RUE$_\mathcal{L}$-model of Π_2. However, it is not even an RSE$_\mathcal{L}$-model of Π_1. If it were, there would be a model $\langle H_1, T\rangle$ of Π_1 with $H_1 \cap \mathcal{L} = H_2$. Since $X \subseteq H \cap \mathcal{L} \subseteq H_1$, $\langle H_1, T\rangle$ would be a non-total model of $\Pi_1 \cup X$, which is impossible by the initial assumption that $\langle T, T\rangle$ is an equilibrium model of $\Pi_1 \cup X$. $\square$

As we have seen in Lemma 2, total RUE$_\mathcal{L}$-models and total RSE$_\mathcal{L}$-models coincide. For non-total RUE$_\mathcal{L}$-models, we obtain an alternative characterisation as follows:

Lemma 3. *Let Π be a theory in $\mathcal{L}'$ and $\mathcal{L}$ a sublanguage of $\mathcal{L}'$. A pair $\langle H, T\rangle$ is a non-total RUE$_\mathcal{L}$-model of Π iff $\langle T, T\rangle \models \Pi$ and it is the $\mathcal{L}$-projection of an* (*unrelativised*) *UE-model $\langle H', T\rangle$ of Π with $H' \cap \mathcal{L} \subset T \cap \mathcal{L}$.*

5 An Application to Prediction and Explanation

In this section, we illustrate how the concept of relativised uniform equivalence can be applied in contexts such as prediction and abductive inference and explanation. Different types of scenarios are possible. For instance, in predicting the behaviour of physical systems we might have a general theory Π comprising strict laws as well as nonmonotonic rules, e.g., describing inertia axioms, default conditions etc., together with initial conditions represented by atomic formulas in a suitable subset of the language. Another type of scenario is represented by an *abductive logic program*, $\langle \Pi, \mathcal{A}\rangle$, where Π is a logic program (of any general type, e.g., disjunctive, nested, etc.) and $\mathcal{A}$ is a set of literals called *abducibles* in a suitable sublanguage of Π. In each case, we are interested in the question: When are two such "theories" equivalent in terms of predictive power, explanatory capacity, and so on? The structure of inference is similar in the two cases mentioned. In each case, the theory Π conjoined with a set $\{A_1, \ldots, A_n\}$ of literals representing initial conditions, abducibles, etc., entails a sentence, say φ, representing, e.g., the prediction of a physical state, the effects of an action, or an explanandum in an abductive system. In the context of equilibrium logic and ASP, entailment is of course nonmonotonic.[2]

To fix notation and terminology, let us consider the general case of *abductive theories*, which are given as pairs of form $\langle \Pi, \mathcal{A}\rangle$, where Π is a theory and $\mathcal{A}$ is a set of literals, and the matter of equivalence with respect to abductive explanations. This leads to the following definition.

[2] The main difference between a prediction in the former sense and an abductive explanation in the latter sense is *methodological*: in the first case, the literals $\{A_1, \ldots, A_n\}$ are specified in advance as part of the initial conditions of the system, while in the second case, it is φ that is supplied in advance as an explanandum, and the abducibles $\{A_1, \ldots, A_n\}$ are to be discovered.

Definition 6. *An* abductive explanation *of a sentence φ by an abductive theory $\mathcal{P} = \langle \Pi, \mathcal{A} \rangle$ is a set $\{A_1, \ldots, A_n\}$ satisfying*

$$\Pi \cup \{A_1, \ldots, A_n\} \mathrel{|\!\sim} \varphi \tag{1}$$

as well as the following two conditions:

6.1 $\{A_1, \ldots, A_n\} \subseteq \mathcal{A}$ and
6.2 $\Pi \cup \{A_1, \ldots, A_n\}$ is consistent,

where $\mathrel{|\!\sim}$ is nonmonotonic entailment.[3]

Note that Condition 6.2 merely ensures that the explanation of φ is non-trivial. For present purposes we do not, however, insist that $\{A_1, \ldots, A_n\}$ be a minimal set of abducibles explaining φ, nor even that it is non-empty. We note further that Definition 6 is equivalent to the definition of an abductive explanation as given by Inoue and Sakama [8] for the case of disjunctive logic programs with default negations in their heads.

If $\{A_1, \ldots, A_n\}$ is an abductive explanation of φ from $\mathcal{P}$, then we also say that $\{A_1, \ldots, A_n\}$ *explains φ in $\mathcal{P}$*. $\mathcal{P}$ is said to have *explanatory power* if there exist some φ and $\{A_1, \ldots, A_n\}$ satisfying (1) as well as Conditions 6.1 and 6.2. Evidently, two abductive theories can have the same explanatory power in weaker or stronger senses. They may capture the same explananda by means of possibly differing explanans (abducibles), and therefore differing explanations, or they may support essentially the same explanations. In this latter sense, we can say therefore that two abductive theories, $\mathcal{P}_1$ and $\mathcal{P}_2$, based on the same abducible set $\mathcal{A}$, have the *same explanatory power in the strong sense* if, for any φ and any $\{A_1, \ldots, A_n\} \subseteq \mathcal{A}$, $\{A_1, \ldots, A_n\}$ explains φ in $\mathcal{P}_1$ iff $\{A_1, \ldots, A_n\}$ explains φ in $\mathcal{P}_2$. We consider here only abductive theories with (non-vacuous) explanatory power.

We can easily relate this notion of explanatory equivalence to relativised uniform equivalence. The following is straightforward.

Proposition 1. *Let $\mathcal{P}_1 = \langle \Pi_1, \mathcal{A} \rangle$ and $\mathcal{P}_2 = \langle \Pi_2, \mathcal{A} \rangle$ be abductive theories based on the same abducibles. If Π_1 and Π_2 are uniformly equivalent relative to $\mathcal{A}$, then $\mathcal{P}_1$ and $\mathcal{P}_2$ have the same explanatory power* (*in the strong sense*).

If Π_1 and Π_2 are uniformly equivalent relative to $\mathcal{A}$, then for any $\{A_1, \ldots, A_n\} \subseteq \mathcal{A}$, $\Pi_1 \cup \{A_1, \ldots, A_n\}$ and $\Pi_2 \cup \{A_1, \ldots, A_n\}$ have the same equilibrium models, so the explanatory power of $\mathcal{P}_1$ and $\mathcal{P}_2$ is the same whether we interpret entailment $\mathrel{|\!\sim}$ in the cautious or brave sense.

To establish a converse of Proposition 1, we need to pin down the type of inference defined by $\mathrel{|\!\sim}$. Evidently, brave reasoning has a greater chance of succeeding, since *prima facie* it seems possible that theories might have the same consequences in the cautious sense, even under the addition of new atoms, yet have different equilibrium models and therefore not be relativised uniformly equivalent.

So let us suppose that $\mathrel{|\!\sim}$ is entailment with respect to to some equilibrium model; in other words, $\Pi \mathrel{|\!\sim} \varphi$ iff φ is true in some equilibrium model of Π. Then we have:

[3] We leave open for the moment whether entailment is to be understood in the cautious or brave sense.

Proposition 2. *If $\mathcal{P}_1$ and $\mathcal{P}_2$ have the same (non-vacuous) explanatory power (in the strong sense), then Π_1 and Π_2 are uniformly equivalent relative to $\mathcal{A}$.*

Proof. Assume the hypothesis of the proposition and suppose that they are not uniformly equivalent relative to $\mathcal{A}$. Then, there exists a subset $\{A_1, \ldots, A_n\} \subseteq \mathcal{A}$ such that $\Pi_1 \cup \{A_1, \ldots, A_n\}$ and $\Pi_2 \cup \{A_1, \ldots, A_n\}$ have different equilibrium models. Say, $\Pi_1 \cup \{A_1, \ldots, A_n\}$ has an equilibrium model $\mathcal{M}$ that is not an equilibrium model of $\Pi_2 \cup \{A_1, \ldots, A_n\}$. We can establish that $\mathcal{P}_1$ and $\mathcal{P}_2$ have different explanatory powers if we can find a sentence φ that is true in $\mathcal{M}$, so that

$$\Pi_1 \cup \{A_1, \ldots, A_n\} \mid\!\sim \varphi \tag{2}$$

but

$$\Pi_2 \cup \{A_1, \ldots, A_n\} \not\mid\!\sim \varphi. \tag{3}$$

This means that φ has to be chosen so there is no other equilibrium model of $\Pi_2 \cup \{A_1, \ldots, A_n\}$ in which φ is true. Moreover, Conditions 6.1 and 6.2 above should also hold for (2). By assumption, no equilibrium model of $\Pi_2 \cup \{A_1, \ldots, A_n\}$ can be equivalent to $\mathcal{M}$ in that it satisfies exactly the same sentences; otherwise it would make exactly the same literals true and false and so be exactly $\mathcal{M}$. So, for each equilibrium model $\mathcal{M}_i$ of $\Pi_2 \cup \{A_1, \ldots, A_n\}$, there must be some sentence α_i true in $\mathcal{M}$ that is not true in $\mathcal{M}_i$. Since we are assuming that the theories are finite, there are at most finitely many equilibrium models $\mathcal{M}_i$ of $\Pi_2 \cup \{A_1, \ldots, A_n\}$ and therefore finitely many such α_i. Evidently, the sentence $\bigwedge_i \alpha_i$ is true in $\mathcal{M}$ but not true in any equilibrium model of $\Pi_2 \cup \{A_1, \ldots, A_n\}$. So, we have

$$\Pi_1 \cup \{A_1, \ldots, A_n\} \mid\!\sim \bigwedge_i \alpha_i \text{ and} \tag{4}$$

$$\Pi_2 \cup \{A_1, \ldots, A_n\} \not\mid\!\sim \bigwedge_i \alpha_i. \tag{5}$$

Furthermore, we have that 6.1 is satisfied and 6.2 holds since $\Pi_1 \cup \{A_1, \ldots, A_n\}$ has a model. This contradicts the initial assumption that $\mathcal{P}_1$ and $\mathcal{P}_2$ have the same explanatory power. □

Combining Propositions 1 and 2 with Theorems 3 and 4 yields the following semantic characterisation of explanatory equivalence.

Corollary 1. *Two abductive theories $\mathcal{P}_1 = \langle \Pi_1, \mathcal{A} \rangle$ and $\mathcal{P}_2 = \langle \Pi_2, \mathcal{A} \rangle$ have the same explanatory power (in the strong sense) iff Π_1 and Π_2 have the same RUE$_{\mathcal{A}}$-models.*

We note that Inoue and Sakama [8, 9] provided for the case of abductive logic programs with default negations in the heads a characterisation similar to our Propositions 1 and 2. However, they derived that two abductive programs $\langle \Pi_1, \mathcal{A} \rangle$ and $\langle \Pi_1, \mathcal{A} \rangle$ have the same explanatory power iff Π and Π_2 are strongly equivalent relative to $\mathcal{A}$. In view of our results, it seems that relativised strong equivalence should in their characterisation be replaced by relative uniform equivalence. Because otherwise we would obtain that, for any $\mathcal{A}$, strong equivalence relative to $\mathcal{A}$ would coincide with uniform equivalence relative to $\mathcal{A}$, which is obviously violated (consider, e.g., the programs $\{a \vee b \leftarrow\}$ and $\{a \leftarrow \mathit{not}\, b;\; b \leftarrow \mathit{not}\, a\}$ which are uniformly equivalent relative to $\{a, b\}$ but not strongly equivalent relative to $\{a, b\}$). Let us also note that they do not apply any semantic characterisations of equivalence analogous to Corollary 1 above. On the other hand, they also consider equivalence in the context of an extended abduction concept [9].

6 Complexity

The complexity of relativised equivalence between disjunctive logic programs has been established by Woltran [19] and has been further studied by Eiter, Fink, and Woltran [2]. Both notions, i.e., RSE and RUE, yield Π_2^P-complete decision problems. Thus, Π_2^P-hardness for these problems is immediate for equilibrium logic. To show that RSE and RUE remain in class Π_2^P for the general setting studied here, first observe that the central subtask of checking whether a given pair $\langle T, T\rangle$ is an equilibrium model of some theory Π is in coNP. Moreover, to decide the complementary problem of RUE between Π_1 and Π_2, one can guess sets T, F of literals and check whether $\langle T, T\rangle$ is an equilibrium model of exactly one of $\Pi_1 \cup F$ and $\Pi_2 \cup F$. This algorithm runs in non-deterministic time with access to an NP-oracle, and thus in Σ_2^P. Π_2^P-membership for RUE follows immediately. The same argumentation holds for RSE in view of the proof of Theorem 2, where it is shown that only very simple theories (which are polynomial in the size to the compared programs) are sufficient to decide RSE.

7 Conclusions and Future Work

In this paper, we extended results for relativised notions of equivalence from logic programs under the answer-set semantics to arbitrary (propositional) theories in equilibrium logic. To this end, we introduced the concept of an $\mathcal{L}$-1-reduct which restricts the language of one world in the two-world Kripke-model for equilibrium logic. These partially bound models can be shown to characterise relativised strong and uniform equivalence between theories in the same manner as relativised SE- and UE-models are used for logic programs [19]. Furthermore, we discussed a possible application of relativised equivalences in the area of abduction and we briefly studied the complexity of the introduced equivalence notions.

An interesting topic for further work is to extend our notions to include the removal of auxiliary letters—important for considering submodules of theories having dedicated output atoms—tantamount to considering *projected equilibrium models*, where only a subset of the atoms are of interest. This would be an extension of the framework introduced by Eiter, Fink, and Woltran [3] for disjunctive logic programs.

References

1. T. Eiter and M. Fink. Uniform equivalence of logic programs under the stable model semantics. In *Proc. ICLP 2003*, vol. 2916 in LNCS, pg. 224–238. Springer, 2003.

2. T. Eiter, M. Fink, and S. Woltran. Semantical characterizations and complexity of equivalences in stable logic programming. *ACM Transactions on Computational Logic*, 2007. To appear.
3. T. Eiter, H. Tompits, and S. Woltran: On solution correspondences in answer-set programming. In *Proc. IJCAI 2005*, pg. 97–102, 2005.
4. P. Ferraris and V. Lifschitz. Weight constraints as nested expressions. *Theory and Practice of Logic Programming* 5:45–74, 2005.
5. P. Ferraris. Answer sets for propositional theories. in *Proc. LPNMR 2005*, vol. 3662 in LNCS, pg. 119–131. Springer, 2005.
6. Y. Gurevich. Intuitionistic logic with strong negation. *Studia Logica*, 36(1–2):49–59, 1977.
7. K. Inoue and C. Sakama. Equivalence of logic programs under updates. In *Proc. JELIA 2004*, vol. 3229 of LNCS, pg. 174–186. Springer, 2004.
8. K. Inoue and C. Sakama. Equivalence in abductive logic. In *Proc. IJCAI 2005*, pg. 472–477, 2005.
9. K. Inoue and C. Sakama. On abductive equivalence. In L. Magnani (ed), *Model-Based Reasoning in Science and Engineering. Cognitive Science, Epistemology, Logic*, Studies in Logic, pg. 333–352, College Publications, London, 2006.
10. M. Kracht. On extensions of intermediate logics by strong negation. *Journal of Philosophical Logic*, 27(1):49–73, 1998.
11. V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.
12. D. Pearce. A new logical characterisation of stable models and answer sets. In *Non-Monotonic Extensions of Logic Programming* (*NMELP'96*), vol. 1216 of LNCS, pg. 57–70. Springer, 1997.
13. D. Pearce. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence*, 47(1-2):3–41, 2006.
14. D. Pearce and A. Valverde. Uniform equivalence for equilibrium logic and logic programs. In *Proc. LPNMR 2004*, vol. 2923 of LNCS, pg. 194–206. Springer, 2004.
15. H. Turner. Strong equivalence for logic programs and default theories (made easy). In *Proc. LPNMR 2001*, vol. 2173 of LNCS, pg. 81–92. Springer, 2001.
16. D. van Dalen. Intuitionistic logic. In *Handbook of Philosophical Logic, Volume III: Alternatives in Classical Logic*. D. Reidel Publishing, 1986.
17. N. N. Vorob'ev. A constructive propositional calculus with strong negation (in Russian). *Doklady Akademii Nauk SSR*, 85:465–468, 1952.
18. N. N. Vorob'ev. The problem of deducibility in constructive propositional calculus with strong negation (in Russian). *Doklady Akademii Nauk SSR*, 85:689–692, 1952.
19. S. Woltran. Characterizations for relativized notions of equivalence in answer set programming. in *Proc. JELIA 2004*, vol. 3229 of LNCS, pg. 161–173. Springer, 2004.

Interpretability and Equivalence in Quantified Equilibrium Logic

David Pearce[1]* and Agustín Valverde[2]**

[1] Computing Science and Artificial Intelligence,
Univ. Rey Juan Carlos, (Móstoles, Madrid), Spain.
davidandrew.pearce@urjc.es

[2] Dept. of Applied Mathematics, Univ. of Málaga, Spain.
a_valverde@ctima.uma.es

Abstract. The study of synonymy among propositional theories in equilibrium logic, begun in [36], is extended to the first-order case.

1 Introduction

Quantified equilibrium logic (QEL) has been developed in [37–39] as a logical foundation for answer set programs with variables. In particular, the version of QEL presented in [39] and [23] can be considered adequate for the general, first-order version of stable model semantics as given in [16]. This version of QEL is based on the logic $\mathbf{QHT}^s_=$, called quantified here-and-there logic with static domains and decidable equality. Logic programs or general theories are strongly equivalent with respect to QEL (or stable model semantics) if and only if they are logically equivalent in $\mathbf{QHT}^s_=$, [23].

In answer set programming (ASP) strong equivalence (and other forms of equivalence between programs) has been recognised as providing an important conceptual and practical tool for program simplification, transformation and optimisation. Following its initial study in [22], the concept of strong equivalence for logic programs in ASP has given rise to a substantial body of further work looking at different characterisations [15, 43], new variations and applications of the idea [8, 35, 44], as well as developing systems to test for strong equivalence [35, 9]. Recently, some of this work on program transformation [10, 45] has been extended to the first-order case.

In basic areas of mathematics, like algebra and geometry, one is familiar with the idea that theories may be presented in different ways with different primitive concepts. Similarly, if one consideres taxonomies, classification schemes, ontologies and in general any knowledge-based system, there are often many different ways to represent apparently the same information. This motivates the search for a concept of equivalence or synonymy that applies to logic programs or nonmonotonic theories that are formulated in different vocabularies. This idea was pursued in [36] which proposed a formal concept of synonymy applying to logic programs and propositional theories in equilibrium logic and answer set semantics. The aim of the present paper is to extend this

* Partially supported by CICyT projects TIC-2003-9001-C02 and TIN2006-15455-CO3.

** Partially supported by CICyT project TIC-2003-9001-C01, TIN2006-15455-CO1 and Junta de Andalucia project TIC-115 work to theories formulated in first-order logic by using quantified equilibrium logic. We start following [36] by considering formal and informal desiderata that a concept of synonymy should fulfil. We then introduce QEL as a logical foundation for ASP and extensions, and present the main characterisation of strong equivalence from [23]. In §4 we propose a strong concept of equivalence or synonymy for theories in quantified equilibrium logic, give different characterisations of it, and show that it fulfils the adequacy conditions discussed in 2. The main characteristics of this concept are as follows. Theories Π_1 and Π_2 in distinct languages are said to be synonymous if each is bijectively interpretable in the other. In particular, this means that there is faithful interpretation of each theory in the other and a one-one correspondence between the models of the two theories. This correspondence preserves the property of being an equilibrium model or answer set. In addition, Π_1 has a definitional extensions that is strongly equivalent to a definitional extension of Π_2. Moreover, in a suitable sense, Π_1 and Π_2 remain equivalent or synonymous when extended by the addition of new formulas.

2 Synonymous Theories

What does it mean to say that two programs or theories, Π_1 and Π_2, in different languages, $\mathcal{L}_1$ and $\mathcal{L}_2$, are synonymous? We consider six desiderata D1-D6 that we believe should be satisfied by any basic concept of synonymy. D1-D3 and D5-D6 are quite general and seem to be applicable to any theories describing or modelling some knowledge domain; D4 takes account of the special nature of a nonmonotonic or logic programming system.

D1. Translatability. The language $\mathcal{L}_1$ of Π_1 should be translatable, via a mapping, say τ, into the language $\mathcal{L}_2$ of Π_2. The translation τ should be uniform, so we require it to be recursive.

D2. Semantic correspondence. There should be a corresponding correlation between the structures of $\mathcal{L}_1$ and $\mathcal{L}_2$, in particular a mapping F from $\mathcal{L}_2$-structures to $\mathcal{L}_1$-structures that respects the translation τ in the sense that for any $\mathcal{L}_2$-structure $\mathcal{I}$ and $\mathcal{L}_1$-formula φ,

$$F(\mathcal{I}) \models \varphi \Leftrightarrow \mathcal{I} \models \tau(\varphi).$$

D3. Equivalence. Under translation, Π_1 and Π_2 should be in an obvious sense equivalent.

D4. Intended models. The semantic correlation should respect the intended models of the two theories. In the present case this means preserving the property of being an equilibrium model or answer set: $\mathcal{M}$ is an answer set Π_2 iff $F(\mathcal{M})$ is an answer set of Π_1.

D5. Idempotence. If Π_1 is synonymous with Π_2 under the previous mappings, then under corresponding mappings, say τ' and F', Π_2 should be synonymous with Π_1.

D6. Robustness. Π_1 and Π_2 should remain synonymous under the addition of new formulas, ie. for any Σ, $\Pi_1 \cup \Sigma$ should be synonymous with $\Pi_2 \cup \tau(\Sigma)$, similarly $\Pi_2 \cup \Pi$ with $\Pi_1 \cup \tau'(\Pi)$.

The first two conditions provide the cornerstone of any formal approach to intertheory relations. Different kinds of relations between theories are obtained by specifying additional conditions that the mappings should satisfy (see eg [30, 34, 41]). In this case we require (D3, D5) that theories are in an obvious sense equivalent once the translation maps are made available. Since we are dealing here with logic programs and their generalisations in the ASP framework, we can understand this either in the weaker sense of having the same answer sets, or in the sense of strong equivalence explained earlier. The problem is that if we choose the weaker variant then we have virtually no hope to fulfil condition D6 which requires that the theories remain equivalent when embedded in any richer context. On the other hand, if we interpret D3 to mean that under suitable translation manuals, Π_1 and Π_2 are strongly equivalent, then we may expect that Π_1 and Π_2 remain synonymous when extended with new rules.

Perhaps somewhat surprisingly we shall approach the problem of synonymy via the classical theory of interpretations. Briefly we shall say that theories are synonymous if each is faithfully interpreted in the other in such a way that the interpretations are idempotent (see below); this is basically the standard approach followed in classical predicate logic, see eg. [4, 40]. We adapt it here to the case of a nonmonotonic system based on a non-classical logic.

3 Quantified Equilibrium Logic (QEL)

Equilibrium logic for propositional theories and logic programs was presented in [31] as a foundation for answer set semantics and extended to the first-order case in [37, 38] and in slightly more general, modified form in [39]. For a survey of the main properties of equilibrium logic, see [32]. Usually in quantified equilibrium logic we consider a full first-order language allowing function symbols and we include a second, strong negation operator as occurs in several ASP dialects. For the present purpose we consider the function-free language with a single negation symbol, '$\neg$'. So, in particular, we shall work with a quantified version of the logic HT of *here-and-there*. In other respects we follow the treatment of [39].

3.1 General Structures for Quantified Here-and-There Logic

A *function-free first-order language* $\mathcal{L} = \langle C, P \rangle$ consists of a sets of constants C and predicate symbols P; each predicate symbol $p \in P$ has an assigned arity. Moreover, we assume a fixed countably infinite set of variables, the symbols, '$\rightarrow$', '$\vee$', '$\wedge$', '$\neg$', '$\exists$', '$\forall$' and auxiliary parentheses '(','\)'. Variables and constant are generically called *terms*. *Atoms* and *formulas* are constructed as usual; *closed* formulas, or *sentences*, are those where each variable is bound by some quantifier. A *theory* Π is a set of sentences.

If D is a non-empty set, we denote by $\mathrm{At}_D(C, P)$ the set of atomic sentences of $\mathcal{L} = \langle C, P \rangle$ with additional constant symbols for each element of D. A *here-and-there* $\mathcal{L}$-structure with static domains is a tuple $\mathcal{I} = \langle (D, I), I^h, I^t \rangle$ where

– D is a non-empty set, called the *domain* of $\mathcal{I}$.
– $I\colon C \cup D \rightarrow D$ is called the *assignment* and verifies $I(d) = d$ for all $d \in D$.
– $I^h \subseteq I^t \subset \mathrm{At}_D(C, P)$.

We can think of $\mathcal{I}$ as a structure similar to a first-order classical structure, but having two parts or components h and t that correspond to two different points or "worlds", 'here' and 'there' in the sense of Kripke semantics for intuitionistic logic [7], where the worlds are ordered by $h \leq t$. At each world $w \in \{h, t\}$ one verifies a set of atoms I^w in the expanded language for the domain D. We call the model static, since, in contrast to say intuitionistic logic, the same domain serves each of the worlds.[1] Since $h \leq t$, whatever is verified at h remains true at t. The satisfaction relation for $\mathcal{I}$ is defined so as to reflect the two different components, so we write $\mathcal{I}, w \models \varphi$ to denote that φ is true in $\mathcal{I}$ with respect to the w component. Evidently we should require that an atomic sentence is true at w just in case it belongs to I^w. Formally, if $p(t_1, \ldots, t_n) \in \mathrm{At}_D$ then

$$\mathcal{I}, w \models p(t_1, \ldots, t_n) \quad \text{iff} \quad p(I(t_1), \ldots, I(t_n)) \in I^w.$$

Then $\models$ is extended recursively as follows[2]:

- $\mathcal{I}, w \models \varphi \wedge \psi$ iff $\mathcal{I}, w \models \varphi$ and $\mathcal{I}, w \models \psi$.
- $\mathcal{I}, w \models \varphi \vee \psi$ iff $\mathcal{I}, w \models \varphi$ or $\mathcal{I}, w \models \psi$.
- $\mathcal{I}, t \models \varphi \rightarrow \psi$ iff $\mathcal{I}, t \not\models \varphi$ or $\mathcal{I}, t \models \psi$.
- $\mathcal{I}, h \models \varphi \rightarrow \psi$ iff $\mathcal{I}, t \models \varphi \rightarrow \psi$ and $\mathcal{I}, h \not\models \varphi$ or $\mathcal{I}, h \models \psi$.
- $\mathcal{I}, w \models \neg\varphi$ iff $\mathcal{I}, t \not\models \varphi$.
- $\mathcal{I}, t \models \forall x \varphi(x)$ iff $\mathcal{I}, t \models \varphi(d)$ for all $d \in D$.
- $\mathcal{I}, h \models \forall x \varphi(x)$ iff $\mathcal{I}, t \models \forall x \varphi(x)$ and $\mathcal{I}, h \models \varphi(d)$ for all $d \in D$.
- $\mathcal{I}, w \models \exists x \varphi(x)$ iff $\mathcal{I}, w \models \varphi(d)$ for some $d \in D$.

Truth of a sentence in a structure is defined as follows: $\mathcal{I} \models \varphi$ iff $\mathcal{I}, w \models \varphi$ for each $w \in \{h, t\}$; in this case, $\mathcal{I}$ is said to be a *model* of φ. An structure $\mathcal{I}$ is a model of a theory Π if it is a model of every $\varphi \in \Pi$, denoted by $\mathcal{I} \models \Pi$. A sentence φ is *valid* if it is true in all structures, denoted by $\models \varphi$. A sentence φ is a *consequence* of a theory Π if every model of Π is a model of φ, in symbols $\Pi \models \varphi$. The resulting logic is called *Quantified Here-and-There Logic with static domains* denoted by $\mathbf{QHT}^s(\mathcal{L})$. In terms of satisfiability and validity this logic is equivalent to the logic introduced before in [38].

The logic $\mathbf{QHT}^s(\mathcal{L})$ can be axiomatised as follows. We start with the usual axioms and rules of intuitionistic propositional logic and add the axiom of Hosoi

$$\alpha \vee (\neg\beta \vee (\alpha \rightarrow \beta))$$

which determines 2-element, here-and-there models. This system is extended to first-order logic (see [38, 39]) by adding the following axiom to obtain the usual non-static version of first-order here-and-there logic:

$$\forall x \neg\neg\alpha(x) \rightarrow \exists x(\alpha(x) \rightarrow \forall x \alpha(x))$$

[1] Alternatively it is quite common to speak of a logic with *constant* domains. However this is ambiguous since it might suggest that the domain is composed only of constants, which is not intended here.

[2] The reader may easily check that the following correspond exactly to the usual Kripke semantics for intuitionistic logic given our assumptions about the two worlds h and t and the single domain D, see eg [6]

Finally, we add the following axiom for static domains, to obtain $\mathbf{QHT}^s(\mathcal{L})$:

$$\neg\neg\exists x\alpha(x) \rightarrow \exists x\neg\neg\alpha(x)$$

Ono proved in [28] that the system obtained by extending the propositional calculus with the axiom $\forall x(\alpha(x) \vee \beta) \rightarrow (\forall x\alpha(x) \vee \beta)$ is complete for $\mathbf{QHT}^s(\mathcal{L})$. In [23], another complete calculus is obtained by extending the propositional calculus with the axiom

$$\exists x(\alpha(x) \rightarrow \forall x\alpha(x))$$

In this paper we also consider the equality predicate, $\doteq \notin P$, interpreted by the following condition for every $w \in \{h, t\}$

– $\mathcal{M}, w \models a \doteq b$ iff $I(a) = I(b)$ for all constants a, b.

To obtain a complete axiomatisation, we then need to add the axiom of "decidible equality"

$$\forall x\forall y(x \doteq y \vee x \not\doteq y).$$

We denote the resulting logic by $\mathbf{QHT}^s_{=}(\mathcal{L})$ (see [23] for details).

As usual in first order logic, satisfiability and validity are independent from the language. If $\mathcal{I} = \langle(D, I), I^h, I^t\rangle$ is an $\mathcal{L}'$-structure and $\mathcal{L}' \supset \mathcal{L}$, we denote by $\mathcal{I}|_{\mathcal{L}}$ the restriction of $\mathcal{I}$ to the sublanguage $\mathcal{L}$:

$$\mathcal{I}|_{\mathcal{L}} = \langle(D, I|_{\mathcal{L}}), I^h|_{\mathcal{L}}, I^t|_{\mathcal{L}}\rangle$$

Proposition 1. *Suppose that $\mathcal{L}' \supset \mathcal{L}$, Π is a theory in $\mathcal{L}$ and $\mathcal{M}$ is an $\mathcal{L}'$-model of Π. Then $\mathcal{M}|_{\mathcal{L}}$ is a $\mathcal{L}'$-model of Π.*

Proposition 2. *Suppose that $\mathcal{L}' \supset \mathcal{L}$ and $\varphi \in \mathcal{L}$. Then φ is valid (resp. satisfiable) in $\mathbf{QHT}^s_{=}(\mathcal{L})$ if and only if is valid (resp. satisfiable) in $\mathbf{QHT}^s_{=}(\mathcal{L}')$.*

This proposition allows us to omit reference to the language in the logic so it can be denoted simply by $\mathbf{QHT}^s_{=}$.

3.2 Equilibrium Models

As in the propositional case, quantified equilibrium logic is based on a suitable notion of minimal model.

Definition 1. *Among quantified here-and-there structures we define the order $\trianglelefteq$ as follows: $\langle(D, I), I^h, I^t\rangle \trianglelefteq \langle(D', J), J^h, J^t\rangle$ if $D = D'$, $I = J$, $I^t = J^t$ and $I^h \subseteq J^h$. If the subset relation holds strictly, we write '$\triangleleft$'.*

Definition 2. *Let Π be a theory and $\mathcal{I} = \langle(D, I), I^h, I^t\rangle$ a model of Π.*

1. *$\mathcal{I}$ is said to be* total *if $I^h = I^t$.*
2. *$\mathcal{I}$ is said to be an* equilibrium *model of Π (or short, we say: "$\mathcal{I}$ is in equilibrium") if it is minimal under $\trianglelefteq$ among models of Π, and it is total. It is denoted by $\mathcal{I} \models_e \Pi$.*

Notice that a total here-and-there model of a theory Π is equivalent to a classical first order model of Π.

The logic defined by the equilibrium models is called *Quantified Equilibrium Logic* and it is also independent of the language, as seen by the following result.

Proposition 3. *Let Π be a theory in $\mathcal{L}$ and $\mathcal{M}$ an equilibrium model of Π in $\mathbf{QHT}^s_=(\mathcal{L}')$ with $\mathcal{L}' \supset \mathcal{L}$. Then $\mathcal{M}|_\mathcal{L}$ is an equilibrium model of Π in $\mathbf{QHT}^s_=(\mathcal{L})$.*

3.3 Strong equivalence for theories

We say that two sets Γ, Δ of first-order sentences are *strongly equivalent* if for every set Σ of first-order sentences, possibly of a larger signature, the sets $\Gamma \cup \Sigma$, $\Delta \cup \Sigma$ have the same equilibrium models.

Theorem 1 (Strong Equivalence of theories, [23]). *For any sets Γ, Δ of first-order sentences, the following conditions are equivalent:*

(i) the sets Γ and Δ are satisfied by the same here-and-there structures;
(ii) for every set Σ of first-order sentences, possibly of a larger signature, the sets $\Gamma \cup \Sigma$ and $\Delta \cup \Sigma$ have the same equilibrium models, ie Γ and Δ are strongly equivalent.

Note that the above notion of equilibrium model coincides with the concept of stable model for logic programs with variables presented in [16]. The concept of strong equivalence and its characterisation can be found in [23]. By strong completeness, condition (i) of Theorem 1 means that Γ and Δ are logically equivalent in $\mathbf{QHT}^s_=$.

4 Interpretability and Synonymy

We use the following notation and terminology. Boldface $\mathbf{x}$ stands for a tuple of variables, $\mathbf{x} = (x_1, \ldots, x_n)$, while $\varphi(\mathbf{x}) = \varphi(x_1, \ldots, x_n)$ is a formula whose free variables are x_1,..., x_n, and $\forall\mathbf{x} = \forall x_1 \ldots \forall x_n$. If t_i are terms, then $\mathbf{t} = (t_1, \ldots, t_n)$ denotes a *vector* of terms. Let $\mathcal{L} = \langle C, P\rangle$ be a first-order language, $p \notin P$ a new predicate symbol and $\mathcal{L}' = \langle C, P \cup \{p\}\rangle$. Let Π be a theory in $\mathcal{L}'$. Explicit and implicit definability are understood as follows

(i) p is said to be *explicitly definable* in Π, if there is an $\mathcal{L}$-formula $\delta^\tau_p(\mathbf{x})$ such that

$$\Pi \models \forall\mathbf{x}(p(\mathbf{x}) \leftrightarrow \delta^\tau_p(\mathbf{x})).$$

δ^τ_p is called the *definition* of p.

(ii) p is said to be *implicitly definable* in Π if for any models $\mathcal{M}_1$ and $\mathcal{M}_2$ of Π such that $\mathcal{M}_1|_\mathcal{L} = \mathcal{M}_2|_\mathcal{L}$ we have $\mathcal{M}_1 = \mathcal{M}_2$.
By the strong completeness theorem for $\mathbf{QHT}^s_=$ proved in [23], this definition is equivalent to the following one.

(ii') p is *implicitly definable* in Π if

$$\Pi \cup \Pi[p/q] \models \forall\mathbf{x}(p(\mathbf{x}) \leftrightarrow q(\mathbf{x}))$$

where $q \notin P$ is a new predicate symbol with the same arity as p and $\Pi[p/q]$ is the theory obtained by replacing every occurrence of p by q.

In other words, p is implicitly definable if whenever the interpretation of the $\mathcal{L}$ predicates in models of Π is fixed, the interpretation of p becomes fixed also. The above definitions are readily extended to the case where several new predicates are definable in a theory.

4.1 Interpolation and Beth properties in superintuitionistic logics

When the conditions (i) and (ii') of explicit and implicit definability are always equivalent, the logic in question is said to have the *Beth property*, [18]. Closely related to Beth is the property of *interpolation*. A logic is said to have the interpolation property if whenever

$$\vdash \varphi \to \psi$$

there exists a sentence ξ (the *interpolant*) such that

$$\vdash \varphi \to \xi \quad \text{and} \quad \vdash \xi \to \psi$$

where all predicate and constant symbols of ξ are contained in both φ and ψ.

It can be shown that the interpolation property implies the Beth property in all superintuionistic predicate logics [18]. Moreover, Ono [28] showed that interpolation holds in the logic $\mathbf{QHT}^s$ of quantified here-and-there with constant domains.[3] Consequently, $\mathbf{QHT}^s$ also has the Beth property. Lastly, Maksimova showed in [24, 25] that adding pure equality axioms, eg decidible equality axiom, to any superintuitionistic logic preserves the interpolation and Beth properties (see also [18]). We conclude therefore

Proposition 4. *The logic $\mathbf{QHT}^s_=$ possesses the Beth property.*

Let $\mathcal{L}_1 = \langle C_1, P_1\rangle$ and $\mathcal{L}_2 = \langle C_2, P_2\rangle$ be disjoint languages.[4] By an *interpretation* of $\mathcal{L}_1$ in $\mathcal{L}_2$ we mean

1. For each predicate $p \in P_1$, an $\mathcal{L}_2$-formula δ^τ_p explicitly defining p by the formula $\forall \mathbf{x}(p(\mathbf{x}) \leftrightarrow \delta^\tau_p(\mathbf{x}))$; we denote by $\overline{\tau}$ the set of all definitions.
2. An induced mapping, also denoted by τ, from $\mathcal{L}_1$-formulas (resp. $\mathcal{L}_1$-terms) to $\mathcal{L}_2$-formulas (resp. $\mathcal{L}_2$-terms) such that
   (a) $\tau(x) = x$ and for every $a \in C_1$, $\tau(a) \in C_2$; if $\mathbf{t} = (t_1, \ldots, t_n)$ is a vector of terms, $\tau(\mathbf{t})$ denotes $(\tau(t_1), \ldots, \tau(t_n))$;
   (b) if $\mathbf{t}$ is a vector of terms, then $\tau(p(\mathbf{t})) = \delta^\tau_p(\tau(\mathbf{t}))$; $\tau(t_1 \doteq t_2) = \tau(t_1) \doteq \tau(t_2)$;
   (c) τ is extended recursively by $\tau(\varphi \wedge \psi) = \tau(\varphi) \wedge \tau(\psi)$, $\tau(\varphi \vee \psi) = \tau(\varphi) \vee \tau(\psi)$, $\tau(\varphi \to \psi) = \tau(\varphi) \to \tau(\psi)$, $\tau(\neg\varphi) = \neg\tau(\varphi)$, $\tau(\forall x\varphi) = \forall x\tau(\varphi)$ and $\tau(\exists x\varphi) = \exists x\tau(\varphi)$.

Any interpretation τ of $\mathcal{L}_1$ in $\mathcal{L}_2$ induces a mapping F_τ from $\mathcal{L}_2$-structures to $\mathcal{L}_1$-structures: if $\mathcal{I} = \langle (D, I), I^h, I^t\rangle$, then $F_\tau(\mathcal{I}) = \langle (D, J), J^h, J^t\rangle$ is defined as follows:

[3] Ono's axiomatisation of $\mathbf{QHT}^s$ uses the constant domains axiom $\forall x(\alpha(x) \vee \beta) \to (\forall x\alpha(x) \vee \beta)$, as well as alternative axioms for propositional here-and-there, viz. $p \vee (p \to (q \vee \neg q))$ and $(p \to q) \vee (q \to p) \vee (p \leftrightarrow \neg q)$. However, the axioms given here are equivalent to Ono's.

[4] Any languages can be made disjoint by renaming. Alternatively we can allow that $\mathcal{L}_1$ and $\mathcal{L}_2$ have a common sublanguage which any translations simply leave untouched, ie the sublanguage is always translated by the identity map.

- For every $a \in C_1$, $J(a) = I(\tau(a))$
- $p(\mathbf{t}) \in J^w$ iff $\mathcal{I}, w \models \delta_p^\tau(\tau(\mathbf{t}))$

It is easy to check that for any $\mathcal{L}_1$-sentence φ and any $w \in \{h, t\}$:

$$F_\tau(\mathcal{I}), w \models \varphi \quad \Leftrightarrow \quad \mathcal{I}, w \models \tau(\varphi) \tag{1}$$

and therefore

$$F_\tau(\mathcal{I}) \models \varphi \quad \Leftrightarrow \quad \mathcal{I} \models \tau(\varphi) \tag{2}$$

Let Π_1 and Π_2 be theories in $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively and let τ be an interpretation of $\mathcal{L}_1$ in $\mathcal{L}_2$. Then τ is said to be an *interpretation of Π_1 in Π_2* if for all $\mathcal{L}_1$-sentence φ,

$$\Pi_1 \models \varphi \quad \Rightarrow \quad \Pi_2 \models \tau(\varphi). \tag{3}$$

In this case it is evident that

$$\mathcal{I} \models \Pi_2 \quad \Rightarrow \quad F_\tau(\mathcal{I}) \models \Pi_1. \tag{4}$$

Generally speaking the map F_τ associated with an interpretation τ of $\mathcal{L}_1$ in $\mathcal{L}_2$ does not preserve the ordering $\trianglelefteq$ between $\mathcal{L}_2$-structures. However the following properties are easy to check and will be useful later:

Lemma 1. *Let τ be an interpretation of $\mathcal{L}_1$ in $\mathcal{L}_2$, and let $\mathcal{I}$ be a total $\mathcal{L}_2$-structure. Then (i) $F_\tau(\mathcal{I})$ is a total $\mathcal{L}_1$-structure; and (ii) if $\mathcal{I}' \trianglelefteq \mathcal{I}$, then $F_\tau(\mathcal{I}') \trianglelefteq F_\tau(\mathcal{I})$.*

An interpretation of Π_1 in Π_2 is said to be *faithful* if the converse of (3) also holds, ie we have $\Pi_1 \models \varphi$ iff $\Pi_2 \models \tau(\varphi)$. As in classical interpretability theory, further special cases of interpretation can be obtained by imposing additional conditions on the syntactic and semantic translations.

Proposition 5. *Let τ be an interpretation of Π_1 in Π_2. Then the following are equivalent.*
(i) For every $\mathcal{L}_2$-formula $\psi(\mathbf{x})$ there is an $\mathcal{L}_1$-formula $\varphi(\mathbf{x})$ such that $\Pi_2 \models \forall\mathbf{x}(\psi(\mathbf{x}) \leftrightarrow \tau(\varphi(\mathbf{x})))$; ie τ is surjective.
(ii) There is an interpretation σ of $\mathcal{L}_2$ in $\mathcal{L}_1$ such that for every $\mathcal{L}_2$-formula ψ, $\Pi_2 \models \forall\mathbf{x}(\psi(\mathbf{x}) \leftrightarrow \tau(\sigma(\psi(\mathbf{x}))))$.
(iii) The mapping F_τ from models of Π_2 into models of Π_1 is an injection.

An interpretation satisfying any of (i)-(iii) of Proposition 5 is said to be *surjective*. Such interpretation preserve the property of being an equilibrium model, in the following sense.

Proposition 6. *Let τ be a surjective interpretation of Π_1 in Π_2. For any model $\mathcal{M}$ of Π_2, if $F_\tau(\mathcal{M})$ is an equilibrium model of Π_1 then $\mathcal{M}$ is an equilibrium model of Π_2.*

If τ is a surjective and a faithful interpretation, then it is said to be a *bijective interpretation* of Π_1 in Π_2. It is easy to verify that if τ is a bijective interpretation of Π_1 in Π_2, then the interpretation σ of Π_2 in Π_1, defined by condition (ii) in Prop. 5, is also bijective. The interpretation σ is called the *inverse* of τ and we say that the two programs or theories are *synonymous* with respect to τ and σ.

Proposition 7. *If τ is a bijective interpretation of Π_1 in Π_2 then the mapping F_τ is a one-one correspondence between models of Π_1 and models of Π_2.*

Given an inverse interpretation σ, we can map $\mathcal{L}_1$-structures $\mathcal{I}$ to $\mathcal{L}_2$-structures $F_\sigma(\mathcal{I})$ in the same way as before. It is readily seen that $F_\sigma(F_\tau(\mathcal{M})) = \mathcal{M}$ if $\mathcal{M}$ is a model of Π_2; however the equality need not hold for other structures (even in the classical case).

4.2 Verifying the adequacy conditions

Let us now consider synonymy in light of the adequacy conditions D1-D6. First we consider the sense in which two synonymous theories can be considered equivalent.

Proposition 8. *Let Π_1 and Π_2 be synonymous wrt τ and σ. Then $\Pi_2 \cup \overline{\tau}$ is strongly equivalent with $\Pi_1 \cup \overline{\sigma}$. Thus Π_1 and Π_2 have a common definitional extension, ie there is a theory Π in $\mathcal{L}_2 \cup \mathcal{L}_1$, such that $\Pi_2 \cup \overline{\tau} \equiv \Pi_1 \cup \overline{\sigma} \equiv \Pi$.*

In fact Proposition 8 can be strengthened to an equivalence: two theories are bijectively interpretable if and only if they have a common definitional extension. This expresses one way in which the two theories are in an obvious sense equivalent once enriched with suitable translation manuals. Notice too that there is a close relationship between Π_2 and the translation $\tau(\Pi_1)$ of Π_1 (similarly between Π_1 and the translation $\sigma(\Pi_2)$ of Π_2). It is already clear that $\Pi_2 \models \tau(\Pi_1)$. Although it is not generally true, even in the classical case, that $\Pi_2 \equiv \tau(\Pi_1)$, we do however have:

Corollary 1. *Let Π_1 and Π_2 be synonymous wrt τ and σ. For any $\mathcal{L}_2$-formula φ, $\Pi_2 \models \varphi \leftrightarrow \tau\sigma(\varphi)$, and $\Pi_2 \models \varphi \Rightarrow \tau(\Pi_1) \models \tau\sigma(\varphi)$.*

Next we turn to condition D4.

Proposition 9. *Let Π_1 and Π_2 be theories in $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively, synonymous wrt τ and σ. Then the bijective mapping F_τ from models of Π_2 to models of Π_1 preserves the equilibrium property, ie. $\mathcal{M} \models_e \Pi_2$ iff $F_\tau(\mathcal{M}) \models_e \Pi_1$.*

Clearly, condition D5 is satisfied and the presence of an inverse interpretation provides the sense in which the correspondence between Π_1 and Π_2 is idempotent. Lastly we consider D6.

Proposition 10. *Let Π_1 and Π_2 be theories in $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively synonymous wrt τ and σ. Let Π a set of $\mathcal{L}_1$-formulas. Then $\Pi_1 \cup \Pi$ is synonymous with $\Pi_2 \cup \tau(\Pi)$ wrt τ and σ.*

5 Literature and Related Work

In classical logic there is a large and well-developed body of work on interpretability dating from the 1950s. The first systematic treatments of synonymous theories in this context can be found in [3, 4], a more algebraic approach can be found in [20]. The classical version of Proposition 6 is essentially contained in [3], though a more detailed statement and proof can be found in [40]. Outside the field of mathematics, the classical theory of interpretability and definitional equivalence was extended and applied to empirical forms of knowledge in [29, 34, 30]; see also [41] for a more recent account of translatability issues in such contexts. The theory of interpretations and equivalence in nonclassical logics is less developed, however especially in the case of superintuitionistic logics much is known about key properties, such as interpolation and Beth, on which interpretability theory depends, see eg. [24–26]. In the context of nonmonotonic logic programming the study of different kinds of equivalence between programs is relatively new (see references in section 1). Until now the case of programs in different languages has only been considered in [36]. There has been some discussion of the role and properties of definitions in ASP in [17, 12],.

6 Concluding Remarks

We have argued that formal approaches to intertheory relations developed for mathematical and scientific knowledge can be applied to systems of logic programming and nonmonotonic reasoning used for practical problem solving and knowledge representation in AI. In particular, we have described how the theory of interpretability and definitional equivalence can be applied in the context of first-order logic programs under answer set semantics and nonmonotonic theoreis in the system of quantified equilibrium logic. In this setting we regard theories as synonymous if each is bijectively interpretable in the other, and we have characterised this relation in different ways. We also showed that this reconstruction satisfies a number of intuitive, informal adequacy conditions. The applicability of what is essentially a classical logical approach in a nonclassical context relies on two essential features: first, our underlying logic has several properties such as *Beth* that help to relate the syntax to the semantics of definitions and translations; secondly, in ASP and equilibrium logic the strong concept of equivalence between theories is fully captured in the underlying monotonic logic (*quantified here-and-there*). This allows us to define a robust or modular concept of equivalence across different languages.

Several avenues are left open for future exploration. For example, one might want to study other kinds of interpretability relations, eg where the formula δ_p^τ defining a predicate p may contain additional parameters, or where the semantic mapping F_τ may relate models with different domains. Secondly, one might search for simple structural properties on the models of two programs or theories that are equivalent to or sufficient for synonymy. Thirdly, based on these or other properties of the theories concerned, it would be useful to develop systems for checking synonymy, thereby extending current methods for checking strong equivalence in the case of programs in the same language [9, 35].

References

1. C. Baral *Knowlewdge Representation, Reasoning and Declarative Problem Solving* Cambridge University Press, 2003.

2. M. Balduccini, M. Gelfond, R. Watson and M. Noguiera. The USA-Advisor: A case study in answer set planning. In *Logic Programming and Nonmonotonic Reasoning*, LPNMR 2001, Springer LNAI 2173, 2001.
3. K. de Bouvère. Logical Synonymity. *Indagationes Mathematicae* 27 (1965), 622-629.
4. K. de Bouvère. Synonymous Theories. In J. Addison, L. Henkin and A.Tarski (eds), *Symposium of the Theory of Models*, North-Holland, Amsterdam, 1965, 402-406.
5. F. Calimeri, S. Galizia, M. Ruffolo, P. Rullo Enhancing Disjunctive Logic Programming for Ontology Specification. Proceedings AGP 2003
6. D. van Dalen. *Logic and Structure.* Springer, 1983..
7. D. van Dalen. Intuitionistic logic. In *Handbook of Philosophical Logic, Volume III: Alternatives in Classical Logic*, Kluwer, Dordrecht, 1986.
8. T. Eiter and M. Fink. Uniform equivalence of logic programs under the stable model semantics. In *Int. Conf. of Logic Programming, ICLP'03*, Mumbay, India. Springer, 2003.
9. T. Eiter, M. Fink, H. Tompits and S. Woltran Simplifying Logic Programs under Uniform and Strong Equivalence In V. Lifschitz and I. Niemelä (eds), *Logic Programming and Nonmonotonic Reasoning*, LPNMR 2004, Springer LNAI 2923, 2004.
10. T. Eiter, P. Trazler & S. Woltran. An Implementation for Recognizing Rule Replacements in Non-Ground Answer Set Programs. in M. Fischer *at al* (eds), *Jelia 2006*, LNAI 4160, Springer, 2006, 477-480.
11. H. Enderton. *A Mathematical Introduction to Logic* Academic Press, 1972.
12. S. Erdogem and V. Lifschitz. Definitions in Answer Set Programming in V. Lifschitz and I. Niemelä (eds), *Proceedings LPNMR 2004*, Springer LNAI 2923, 2004.
13. K. Gödel. Zum intuitionistischen aussagenkalkül. *Anzeiger der Akademie der Wissenschaften Wien, mathematisch, naturwissenschaftliche Klasse*, 69:65–66, 1932.
14. A. Heyting. Die formalen regeln der intuitionistischen logik. *Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-mathematische Klasse*, pages 42–56, 1930.
15. D. De Jongh and L. Hendriks. Characterization of strongly equivalent logic programs in intermediate logics. *Theory and Practice of Logic Programming*, 3(3):259–270, 2003.
16. Paolo Ferraris, Joohyung Lee, and Vladimir Lifschitz. A new perspective on stable models. IJCAI 2007.
17. P. Ferraris and V. Lifschitz. Weight Constraints as Nested Expressions *Theory and Practice of Logic Programming*, to appear, 2004.
18. D. Gabbay and L. Maksimova. *Interpolation and Definability: Modal and Intuitionistic Logics.* Ser. Oxford Logic Guides:46, Oxford University Press, Oxford, 2005
19. T. Janhumen, I. Niemelä, D. Seipel, P. Simons, and J.-H. You. Unfolding partiality and disjunctions in stable model semantics. CoRR: cs.AI/0303009, March, 2003.
20. S. Kanger. Equivalent Theories. *Theoria*, 38 (1972), 1-6.
21. N. Leone, G. Pfeifer, W. Faber, T. Eiter, G, Gottlob, S. Perri and F. Scarcello. The DLV System for Knowledge Representation and Reasoning. CoRR: cs.AI/0211004, September, 2003.
22. V. Lifschitz, D. Pearce, and A. Valverde. Strongly equivalent logic programs. *ACM Transactions on Computational Logic*, 2(4):526–541, 2001.
23. V. Lifschitz, D. Pearce, and A. Valverde. A Characterization of Strong Equivalence for Logic Programs with Variables *Proceedings LPNMR 07*, Springer, 2007, to appear.
24. L. Maksimova. Interpolation in superintuitionistic predicate logics with equality. *Algebra and Logic*, 36:543-561, 1997.
25. L.Maksimova. Interpolation in superintuitionistic and modal predicate logics with equality. In: M.Kracht, M. de Rijke, H.Wansing and M.Zakharyaschev, editors. *Advances in Modal Logic, Volume I*, pages 133-141. CSLI Publications, Stanford, 1998.
26. L. Maksimova. Intuitionistic logic and implicit definability. *Annals of Pure and Applied Logic* 105(1-3): 83-102, 2000

27. E. Oikarinen and T. Janhunen. Verifying the Equivalence Logic Programs in the Disjunctive Case. In V. Lifschitz and I. Niemelä (eds), *Logic Programming and Nonmonotonic Reasoning*, LPNMR 2004, Springer LNAI 2923, 2004.
28. H. Ono. Model extension theorem and Craig's interpolation theorem for intermediate predicate logics. *Reports on Mathematical Logic* 15 (1983), 41-58.
29. D. Pearce. Some Relations between Empirical Systems. *Epistemologia* 4 (1981), 363-380.
30. D. Pearce. *Roads to Commensurability* Kluwer (Synthese Library Vol. 187),1987.
31. D. Pearce. A new logical characterization of stable models and answer sets. In *Non-Monotonic Extensions of Logic Programming, NMELP 96*, Springer LNCS 1216, 1997, pages 57–70.
32. D. Pearce. Equilibrium Logic. *Annals of Mathematics and AI*, 2006.
33. D. Pearce, I. P. de Guzmán, and A. Valverde. A tableau calculus for equilibrium entailment. In *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2000*, LNAI 1847, pages 352–367. Springer, 2000.
34. D. Pearce and V. Rantala. New Foundations for Metascience. *Synthese*, 1985.
35. D. Pearce and A. Valverde. Uniform Equivalence for Equilibrium Logic and Logic Programs. In V. Lifschitz and I. Niemelä (eds) *Logic Programming and Nonmonotonic Reasoning*, LPNMR 2004, Springer LNAI 2923, 2004.
36. D. Pearce and A. Valverde. Synonymous Theories in Answer Set Programming and Equilibrium Logic. In Proceedings ECAI 2004, 2004.
37. David Pearce and Agustin Valverde. Towards a first order equilibrium logic for nonmonotonic reasoning. In *Proceedings of European Conference on Logics in Artificial Intelligence (JELIA)*, pages 147–160, 2004.
38. David Pearce and Agustin Valverde. A first order nonmonotonic extension of constructive logic. *Studia Logica*, 80:323–348, 2005.
39. David Pearce and Agustin Valverde. Quantified Equilibrium Logic and the First Order Logic of Here-and-There. Technical Report, Univ. Rey Juan Carlos, 2006,
40. C. Pinter Properties Preserved under Definitional Equivalence and Interpretations. In *Zeitschr. f. math. Logik und Grundlagen d. Math.*, 24: 481–488, 1978.
41. V. Rantala. *Explanatory Translation.* Kluwer (Synthese Library Vol 312), 2002.
42. P. Simons, I. Niemelä, and T. Soininen. Extending and implementing the stable model semantics. *Artificial Intellingence*, 138(1–2):181–234, 2002.
43. H. Turner. Strong equivalence for logic programs and default theories (made easy). In *Proc. of the Logic Programming and Nonmonotonic Reasoning, LPNMR'01*, Springer LNAI 2173, pages 81–92, 2001.
44. H. Turner Strong Equivalence for Causal Theories In V. Lifschitz and I. Niemelä (eds) *Logic Programming and Nonmonotonic Reasoning*, LPNMR 2004, Springer LNAI 2923, 2004.
45. Stefan Woltran. *LPNMR 2007*, to appear.