Universidad Rey Juan Carlos

# Characterising Equilibrium Logic and Nested Logic Programs: Reductions and Complexity

D. Pearce, H. Tompits, and S. Woltran

GIA Technical Report 2007-01-12
January 2007

**Universidad Rey Juan Carlos**
Tulipan s/n
28933, Móstoles, Madrid
ESPAÑA
http://www.ia.escet.urjc.es/

GIA Technical Report

GIA Technical Report 2007-01-12, January 2007

# Characterising Equilibrium Logic and Nested Logic Programs: Reductions and Complexity

DAVID PEARCE
Universidad Rey Juan Carlos, Computing Science and Artificial Intelligence,
Calle Tulipan s/n, E-28933 Madrid, Spain
davidandrew.pearce@urjc.es
HANS TOMPITS, STEFAN WOLTRAN
Technische Universität Wien, Institut für Informationssysteme 184/3,
Favoritenstrasse 9-11, A-1040 Vienna, Austria
[stefan,tompits]@kr.tuwien.ac.at

**Abstract.** Equilibrium logic is an approach to nonmonotonic reasoning that extends the stable model and answer-set semantics for logic programs. In particular, it includes the general case of *nested logic programs*, where arbitrary Boolean combinations are permitted in heads and bodies of rules, as special kinds of theories. In this paper, we present efficient reductions of the main reasoning tasks associated with equilibrium logic and nested logic programs into *quantified propositional logic*, an extension of classical propositional logic where quantifications over atomic formulas are permitted. Thus, quantified propositional logic is a fragment of second-order logic, and its formulas are usually referred to as *quantified Boolean formulas* (QBFs). We provide reductions not only for decision problems, but also for the central semantical objects of equilibrium logic and nested logic programs. In particular, our encodings map a given reasoning task into some QBF such that the latter is valid precisely in case the former holds. The reasoning tasks we deal with here are the consistency problem, brave reasoning, and skeptical reasoning. Additionally, we also provide encodings for testing equivalence of theories or programs under different notions of equivalence, viz. ordinary, strong, and uniform equivalence. For all considered reasoning tasks, we analyse their computational complexity and give strict complexity bounds. Hereby, our encodings yield upper bounds in a direct manner. Besides this convenient feature, our approach has the following benefits: Firstly, our encodings yield a *uniform axiomatisation* of differing problems in a common language. Secondly, extant solvers for QBFs can be used as back-end inference engines to realise implementations of the encoded task in a rapid prototyping manner. Thirdly, our axiomatisations also allow us to straightforwardly relate equilibrium logic with circumscription, showing that they generalise an early result by Lin given for disjunctive logic programs.

[1]Part of the material in this paper has appeared, in preliminary form, in the Proceedings of the 10th Portuguese Conference on Artificial Intelligence (EPIA 2001), pp. 306-320, LNCS vol. 2258, Springer, 2001.

## 1 Introduction

*Equilibrium logic*, introduced by Pearce (1997), is a general purpose propositional formalism for nonmonotonic reasoning. 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". One of the main features of equilibrium logic is that, under all the usual classes of logic programs, it is equivalent to reasoning under answer set semantics and therefore amounts to a conservative extension of answer set inference to the full propositional language. It even includes the general case of *nested logic programs* (Lifschitz et al. 1999), where arbitrary Boolean expressions are permitted in heads and bodies of program rules. With the emergence of answer-set solvers such as `DLV` (Leone et al. 2006), `Smodels` (Simons et al. 2002), or `ASSat` (Lin and Zhao 2002), *answer-set programming* (ASP) now provides a practical and viable environment for knowledge representation and declarative problem solving. (For an overview of equilibrium logic, see Pearce (2006); an excellent treatise on ASP is the compehensive textbook by Baral (2003); a survey of applications is compiled by Woltran (ed.) (2005).)

Our main contribution in this work is to provide a uniform axiomatisation of various problems associated with equilibrium logic and nested logic programs. In fact, we reformulate these problems within a common language such that a sentence in that language is valid iff the encoded problem holds. Moreover, the resulting sentences are not only useful for decision problems, but also characterise via models certain semantical objects in the original setting of equilibrium logic and stable semantics. Finally, our axiomatisations allow us to draw in a direct manner upper complexity bounds for the problems in question. We strengthen these results by providing matching lower bounds, thus giving *strict* complexity results for the encoded problems. At the same time, these complexity results show that our axiomatisations are *adequate* in the sense of Besnard et al. (2005), i.e., roughly speaking, they reflect the inherent complexity of the original problems.

The target language for our endeavour is *quantified propositional logic*, an extension of classical propositional logic where quantifications over atomic formulas are permitted. Thus, quantified propositional logic is a fragment of second-order logic, and its sentences are usually referred to as *quantified Boolean formulas* (QBFs). We describe polynomial-time constructible encodings providing characterisations of the problems under consideration by means of QBFs. Moreover, since logic programming under the answer set semantics is just a special form of equilibrium logic, our encodings for the latter yield also encodings for the former. While the general encodings are quadratic in size of the input problem, we improve them for logic programs and obtain linear encodings.

The general approach of using reductions to QBFs as adopted here has already been applied to several diverse reasoning problems from the area of artificial intelligence. Examples include the following areas:

- nonmonotonic formalisms like logic-based abduction, default logic, and modal nonmonotonic logics (Egly et al. 2000; Eiter et al. 2002);
- consistency-based belief revision (Delgrande et al. 2004);

- paraconsistent reasoning (Arieli and Denecker 2003; Besnard et al. 2005); and
- planning (Rintanen 1999).

Concerning the specific encodings discussed in this paper, for both equilibrium logic and logic programs with nested expressions we provide encodings for the basic reasoning tasks associated with these formalisms. To wit, we consider the *consistency problem*, *brave reasoning*, and *skeptical reasoning*. In the context of equilibrium logic, these tasks are the following:

- decide whether a given theory possesses some equilibrium model;
- decide whether a given formula is contained in at least one equilibrium model of a given theory; and
- decide whether a given formula is contained in all equilibrium models of a given theory.

The corresponding tasks for logic programs are defined analogously. Additionally, we also provide encodings for testing the equivalence of theories or programs under different notions of equivalence. Besides ordinary equivalence, which, in case of equilibrium logic, tests whether two theories have the same equilibrium models, we also consider more refined ones, viz. *strong equivalence* (Lifschitz et al. 2001) and *uniform equivalence* (Eiter and Fink 2003; Pearce and Valverde 2004c). In detail, two theories are

- strongly equivalent iff, for any addition of formulas, the two augmented theories are ordinarily equivalent, and
- uniformly equivalent iff, for any addition of *atoms*, the two augmented theories are ordinarily equivalent.

(Again, above concepts are defined with respect to theories of equilibrium logic; for logic programs, the corresponding notions are defined *mutatis mutandis.*) Intuitively, while strong equivalence realises some form of substitution principle, uniform equivalence, first studied in the context of datalog programs (Maher 1988; Sagiv 1988), is useful for dealing with hierarchically structured program components. Importantly, these notions serve as formal underpinnings for program simplification and modular programming (Eiter et al. 2004b; Pearce 2004), issues which are gaining increasing attention in the literature on answer-set programming.

As equilibrium logic is based on the logic of here-and-there, we also address the latter in our work, not only because it is underlying equilibrium logic, but also since it is closely related to strong and uniform equivalence. Indeed, a QBF module representing reasoning in here-and-there plays a central role in all axiomatisations we provide. This module, together with other simple modules, will serve as sort of a tool box for arranging the encodings of the different problems considered. Hence, our reductions to quantified propositional logic provide an elegant axiomatisation of the considered reasoning tasks. In this way, a whole variety of *prima facie* distinct problems becomes comparable in a uniform setting.

Regarding the concrete complexity results shown in this paper, we obtain that the consistency problem and brave reasoning are $\Sigma_2^P$-complete, while skeptical reasoning is $\Pi_2^P$-complete. Furthermore, checking ordinary or uniform equivalence is

$\Pi_2^P$-complete, while checking strong equivalence is co-NP-complete. All these results hold for both equilibrium logic as well as for nested logic programs. This shows that, with respect to the considered reasoning tasks, equilibrium logic and nested logic programs behave complexitywise precisely as disjunctive logic programs.

Let us now have some words about the advantages of our reduction approach to quantified propositional logic. First of all, as already pointed out above, reductions to QBFs allow us to axiomatise diverse problems in a uniform language. As well, they enable us to derive upper complexity bounds with ease. Secondly, an increasing number of practicably efficient solvers for quantified propositional logic emerged in recent years (see, for instance, Le Berre et al. (2005) for an overview). These can be used as back-end inference engines for our encoded tasks, allowing us to build implementations for the considered reasoning tasks in a rapid-prototyping manner. Thirdly, our axiomatisations allow us also to relate equilibrium logic to circumscription (McCarthy 1980). For the case of disjunctive logic programs, an early result on the relation between answer sets and circumscription was given by Lin (1991). Our encodings generalise Lin's result not only to logic programs with nested expressions but also to full equilibrium logic.

The remainder of the paper is laid out as follows. First, we review the nonclassical logic of here-and-there and its nonmonotonic extension, equilibrium logic. Then, we introduce logic programs as a special case of this logic and discuss the basic elements of quantified propositional logic. We then turn in Section 3 to the main issue of the paper, the characterisation of equilibrium logic in terms of QBFs. As an underlying task, we show how to re-express satisfiability in here-and-there in the setting of classical logic. We then address logic programs with nested expressions, providing reductions which are optimisations of the ones for equilibrium logic. We continue with relating our approach to circumscription, and the final part of Section 3 deals with the encodings for ordinary, uniform, and strong equivalence. In Section 4, we turn to the complexity of the considered reasoning tasks, and Section 5 briefly discusses issues which concern the addition of a second negation, which is usually termed *strong negation* or (with a bit of a misnomer) *classical negation.* Section 6 addresses related work and Section 7 concludes the paper with a brief summary and outlook. A more involving proof is relegated to an appendix.

## 2 Background

Throughout this paper, we use a propositional language whose alphabet consists of (i) the primitive logical connectives $\neg$, $\vee$, $\wedge$, and $\supset$, (ii) the logical constants $\top$ and $\bot$, (iii) the punctuation symbols '(' and ')', and (iv) a class of (propositional) variables. The class of (propositional) formulas is constructed in the usual inductive fashion. Propositional variables are also referred to as *atomic formulas*, or simply *atoms.* We use $p, q, r, \dots$ to denote propositional variables and Greek lower-case letters to denote arbitrary formulas; in either case, the letters may also contain subscripts. In order to improve the readability of formulas, we also allow the use of other kinds of punction symbols besides '(' and ')'. A propositional formula which does not contain the sentential connective $\supset$ is called an *expression.* As usual, a *literal* is either a propositional variable (a *positive literal*) or a propositional variable preceded by the negation symbol $\neg$ (a *negative literal*). Furthermore, $\varphi \equiv \psi$ is an abbreviation of the formula $(\varphi \supset \psi) \wedge (\psi \supset \varphi)$.

The *logical complexity*, $lc(\varphi)$, of a formula $\varphi$ is the number of occurrences of the logical symbols $\neg$, $\vee$, $\wedge$, and $\supset$ in $\varphi$.

By $var(\varphi)$ we understand the set of all variables occurring in a formula $\varphi$. Likewise, for a set $T$ of formulas, $var(T)$ refers to the set of all variables occurring in the formulas in $T$, i.e., $var(T) = \bigcup_{\varphi \in T} var(\varphi)$.

A finite set of formulas is called a *theory*. Usually, a theory $T$ will be identified with the conjunction $\bigwedge_{\varphi \in T} \varphi$ of its elements. For $T = \emptyset$, we define $\bigwedge_{\varphi \in T} \varphi = \top$.

By a (*classical*) *interpretation*, $I$, we understand a set of variables. Informally, a variable $p$ is true under $I$ iff $p \in I$. The *truth value* of a formula $\varphi$, in the sense of classical propositional logic, is denoted by $\nu_I(\varphi)$ and is defined in the usual way. We write $\models \varphi$ to indicate that $\varphi$ is valid in classical propositional logic, and accordingly, we write $T \models \phi$ to denote that $\phi$ is a logical consequence of $T$.

We employ the following notational convention: Given a formula $\varphi$, by $\varphi'$ we understand the result of replacing each variable $p$ occurring in $\varphi$ by a globally new variable $p'$. This is applied analogously for sets of formulas. In particular, if $I$ is an interpretation, then $I'$ is the interpretation $\{p' \mid p \in I\}$ which is disjoint to $I$. Whenever needed, we apply this concept iteratively, i.e., we also use new variables $p''$, $p'''$, etc.

## *2.1 Equilibrium Logic*

We recall the basic concepts of equilibrium logic, an approach to nonmonotonic reasoning developed by Pearce (1997) as a generalisation of the answer-set semantics for logic programs. We give only the relevant aspects here; for more details the reader is referred to (Pearce 1997; Pearce 1999; Lifschitz et al. 2001; Pearce et al. 2000b; Pearce et al. 2000a; Pearce 2006) and the logic texts cited below. Besides the propositional version discussed here, a first-order variant of equilibrium logic has been introduced as well (Pearce and Valverde 2004b).

Equilibrium logic is based on the non-classical logic of *here-and-there*, which we denote by HT. The language of HT is given by the class of propositional formulas as described above, and the axioms and rules of inference of HT are those of intuitionistic logic (cf., e.g., van Dalen (1986)) together with the axiom schema

$$(\neg\varphi \supset \psi) \supset (((\psi \supset \varphi) \supset \psi) \supset \psi)$$

which characterises the three-valued here-and-there logic of Heyting (1930) and Gödel (1932) (hence, HT is sometimes known as *Gödel's three-valued logic*). The standard version of equilibrium logic has two kinds of negation, *intuitionistic negation*, $\neg$, and *strong negation*, $\sim$. For simplicity, we deal first with the restricted version containing just the first negation. Later, in Section 5, we show how strong negation can be added.

The model theory of HT is based on the usual Kripke semantics for intuitionistic logic, which is given in terms of Kripke frames of form $\langle W, \leq \rangle$, where $W$ is a set of *points*, or *worlds*, and $\leq$ is a partial-ordering on $W$ (cf., e.g., van Dalen (1986)), except that Kripke frames for HT are restricted to just one containing exactly two worlds, say $H$ ("here") and $T$ ("there"), with $H \leq T$. As in ordinary Kripke semantics for intuitionistic logic, we can imagine that in each world a set of atoms is verified and that, once verified "here", an atom remains verified "there".

In view of the restricted nature of Kripke frames for HT, it is convenient to define the semantics of HT in terms of *HT-interpretations*, which are ordered pairs of form $\langle I_H, I_T \rangle$, where $I_H$ and $I_T$ are sets of variables such that $I_H \subseteq I_T$. For an HT-interpretation $\mathcal{I} = \langle I_H, I_T \rangle$, a world $w \in \{H, T\}$, and a formula $\varphi$, the *truth value*, $\nu_{\mathcal{I}}(w, \varphi)$, *of* $\varphi$ *in* $w$ *under* $\mathcal{I}$ is given as follows:

1. if $\varphi = \top$, then $\nu_{\mathcal{I}}(w, \varphi) = 1$;
2. if $\varphi = \bot$, then $\nu_{\mathcal{I}}(w, \varphi) = 0$;
3. if $\varphi = v$, for some variable $v$, then $\nu_{\mathcal{I}}(w, \varphi) = 1$ if $v \in I_w$, and $\nu_{\mathcal{I}}(w, \varphi) = 0$ otherwise;
4. if $\varphi = \neg\psi$, then $\nu_{\mathcal{I}}(w, \varphi) = 1$ if, for every world $u$ such that $w \leq u$, $\nu_{\mathcal{I}}(u, \psi) = 0$, and $\nu_{\mathcal{I}}(w, \varphi) = 0$ otherwise;
5. if $\varphi = (\varphi_1 \wedge \varphi_2)$, then $\nu_{\mathcal{I}}(w, \varphi) = \mathit{min}(\{\nu_{\mathcal{I}}(w, \varphi_1), \nu_{\mathcal{I}}(w, \varphi_2)\})$;
6. if $\varphi = (\varphi_1 \vee \varphi_2)$, then $\nu_{\mathcal{I}}(w, \varphi) = \mathit{max}(\{\nu_{\mathcal{I}}(w, \varphi_1), \nu_{\mathcal{I}}(w, \varphi_2)\})$;
7. if $\varphi = (\varphi_1 \supset \varphi_2)$, then $\nu_{\mathcal{I}}(w, \varphi) = 1$ if, for every world $u$ such that $w \leq u$, $\nu_{\mathcal{I}}(u, \varphi_1) \leq \nu_{\mathcal{I}}(u, \varphi_2)$, and $\nu_{\mathcal{I}}(w, \varphi) = 0$ otherwise.

We say that $\varphi$ is *true under* $\mathcal{I}$ *in* $w$ if $\nu_{\mathcal{I}}(w, \varphi) = 1$, otherwise $\varphi$ is *false under* $\mathcal{I}$ *in* $w$. An HT-interpretation $\mathcal{I} = \langle I_H, I_T \rangle$ *satisfies* $\varphi$, or $\mathcal{I}$ is an *HT-model* of $\varphi$, iff $\nu_{\mathcal{I}}(H, \varphi) = 1$. If $\varphi$ possesses some HT-interpretation satisfying it, then $\varphi$ is said to be *HT-satisfiable*, and if every HT-interpretation satisfies $\varphi$, then $\varphi$ is *HT-valid*. An HT-interpretation is an HT-model of a set $T$ of formulas iff it is an HT-model of all elements of $T$. Finally, an HT-interpretation $\langle I_H, I_T \rangle$ is said to be *total* if $I_H = I_T$.

It is easily seen that any HT-valid formula is valid in classical logic, but the converse does not always hold. For instance, $p \vee \neg p$ and $\neg\neg p \supset p$ are valid in classical logic but not in the logic of here-and-there, because $\mathcal{I} = \langle \emptyset, \{p\} \rangle$ is not an HT-model for either of these formulas.

We say that two theories are *equivalent in the logic of here-and-there*, or *HT-equivalent*, iff they possess the same HT-models. Two formulas, $\varphi$ and $\psi$, are HT-equivalent iff the theories $\{\varphi\}$ and $\{\psi\}$ are HT-equivalent.

Equilibrium logic is characterised in terms of a particular minimal-model construction in HT. Formally, an *equilibrium model* of a theory $T$ is a total HT-interpretation $\langle I, I \rangle$ such that (i) $\langle I, I \rangle$ is an HT-model of $T$, and (ii) for every proper subset $J$ of $I$, $\langle J, I \rangle$ is not an HT-model of $T$. $\langle I, I \rangle$ is an equilibrium model of a formula $\varphi$ iff $\langle I, I \rangle$ is an equilibrium model of $\{\varphi\}$.

A formula $\varphi$ is a *brave consequence* of a theory $T$, symbolically $T \vdash_b \varphi$, iff some equilibrium model of $T$ satisfies $\varphi$. Dually, $\varphi$ is a *skeptical consequence* of $T$, symbolically $T \vdash_s \varphi$, iff all equilibrium models of $T$ satisfy $\varphi$.

The basic reasoning tasks in the context of equilibrium logic are the following decision problems:

- Decide whether a given theory $T$ possesses some equilibrium model.

- Given a theory $T$ and a formula $\varphi$, decide whether $T \vdash_b \varphi$ holds.
- Given a theory $T$ and a formula $\varphi$, decide whether $T \vdash_s \varphi$ holds.

The first task is called the *consistency problem*; the second and third tasks are respectively called *brave reasoning* and *skeptical reasoning.*

The following two propositions are straightforward and will be useful later on:

*Proposition 1*
For any HT-interpretation $\mathcal{I} = \langle I_H, I_T \rangle$ and any propositional formula $\varphi$, the following relations hold:

1. $\nu_{\mathcal{I}}(T, \varphi) = 1$ iff $\nu_{I_T}(\varphi) = 1$;
2. $\nu_{\mathcal{I}}(H, \varphi) = 1$ implies $\nu_{\mathcal{I}}(T, \varphi) = 1$; and
3. $\nu_{\mathcal{I}}(H, \varphi) = 1$ iff $\nu_{I_H}(\varphi) = 1$, providing $\varphi$ is an expression without a negation.

Notice that the first part of this proposition states that $\varphi$ is true under $\mathcal{I} = \langle I_H, I_T \rangle$ in the world $T$ iff $\varphi$ is true under $I_T$ in classical logic. The second part is a direct consequence of the notion of an HT-interpretation, viz. in view of the proviso $I_H \subseteq I_T$, which holds for each HT-interpretation $\langle I_H, I_T \rangle$. The third part states that formulas without negations and implications can be evaluated by pure classical means, i.e., in both worlds seperately.

*Proposition 2*
A total HT-interpretation $\langle I, I \rangle$ is an HT-model of $\varphi$ iff $I$ is a model of $\varphi$ in classical logic.

## *2.2 Logic Programs*

Next, we review logic programs with nested expressions under the stable-model semantics, following Lifschitz et al. (1999).[1] Programs of this kind are characterised by allowing bodies and heads of rules to be comprised of arbitrary expressions, i.e., formulas composed of $\wedge$, $\vee$, and $\neg$ only.

Formally, by a rule, $r$, we understand an ordered pair of the form

$$H(r) \leftarrow B(r),$$

where $B(r)$ and $H(r)$ are expressions. We call $B(r)$ the *body* of $r$ and $H(r)$ the *head* of $r$. A *nested logic program*, or simply a *program*, $\Pi$, is a finite set of rules. A *fact* is a rule of form $p \leftarrow$ where $p$ is an atom.

We employ for rules and programs the same notational convention concerning priming as we did for formulas, i.e., $r'$ shall be the result of replacing each atom $p$ in $r$ by $p'$ and, similarly, $\Pi'$ is given by $\{r' \mid r \in \Pi\}$.

Note that programs properly generalise *disjunctive logic programs* (Gelfond and Lifschitz 1991), which are characterised by the condition that bodies of rules are conjunctions of literals and heads are disjunctions of atoms.

[1] Analogously as for equilibrium logic, here we consider programs with only one kind of negation first, however, corresponding to default negation; the case of strong negation will be discussed later.

In what follows, we associate to each rule $r$ a corresponding propositional formula

$$\hat{r} = B(r) \supset H(r)$$

and, accordingly, to each program $\Pi$ a corresponding set of formulas

$$\hat{\Pi} = \{\hat{r} \mid r \in \Pi\}.$$

Furthermore, we define $var(\Pi)$ as the set of all variables occurring in $\Pi$, i.e., $var(\Pi) = var(\hat{\Pi})$.

We call expressions, rules, and programs *basic* iff they do not contain the operator $\neg$. An interpretation $I$ is a *model* of a basic program $\Pi$ if it is a model of the associated set $\hat{\Pi}$ of formulas.

Given an interpretation $I$ and an (arbitrary) program $\Pi$, the *reduct*, $\Pi^I$, of $\Pi$ with respect to $I$ is the basic program obtained from $\Pi$ by replacing every occurrence of an expression $\neg\psi$ in $\Pi$ which is not in the scope of any other negation by $\top$ if $\neg\psi$ is true under $I$ (i.e., if $\nu_I(\neg\psi) = 1$), and by $\bot$ otherwise. $I$ is a *stable model* of $\Pi$ iff it is a minimal model (with respect to set inclusion) of the reduct $\Pi^I$.

*Example 1*
Consider a logic program $\Pi$ consisting of the single rule

$$p \leftarrow (q \wedge r) \vee (\neg q \wedge \neg s). \tag{1}$$

Let us check whether $I = \{p\}$ is a stable model of $\Pi$. Since both $\neg q$ and $\neg s$ are true under $I$, we obtain

$$\Pi^I = \{p \leftarrow (q \wedge r) \vee (\top \wedge \top)\}.$$

Since $(q \wedge r) \vee (\top \wedge \top)$ is classically equivalent to $\top$, the only minimal model of $\Pi^I$ is $\{p\}$. So, $I$ is a stable model of $\Pi$. In fact, there is no other stable model of $\Pi$.

A formula $\varphi$ is said to be a *brave consequence* of a logic program $\Pi$, symbolically $\Pi \vdash_b \varphi$, iff there is a stable model $I$ of $\Pi$ such that $\varphi$ is true under $I$, and $\varphi$ is a *skeptical consequence* of $\Pi$, symbolically $\Pi \vdash_s \varphi$, iff $\varphi$ is true under all stable models $I$ of $\Pi$.

The basic reasoning tasks in the context of logic programs are defined, *mutatis mutandis*, as for equilibrium logic and are likewise referred to as the *consistency problem*, *brave reasoning*, and *skeptical reasoning*, respectively.

It is well known that disjunctive logic programs satisfy the so-called *anti-chain property*, which expresses that, for all stable models $I$ and $J$ of a program $\Pi$, if $I \subseteq J$, then $I = J$. The next example shows that programs with nested expressions do not meet the anti-chain property in general.

*Example 2*
Let $\Pi = \{p \leftarrow \neg\neg p\}$. We show that $\Pi$ possesses two stable models, viz. $I_1 = \emptyset$ and $I_2 = \{p\}$. Concerning $I_1$, we have that $\neg p$ is true under $I_1$, and therefore $\Pi^{I_1} = \{p \leftarrow \bot\}$, which has $\emptyset$ as its minimal model. As for $I_2$, here it holds that $\neg p$ is false under $I_2$, and so $\Pi^{I_2} = \{p \leftarrow \top\}$, which has $\{p\}$ as its minimal model. Consequently, we have stable models $I_1$ and $I_2$ of $\Pi$ such that $I_1 \subseteq I_2$ but $I_1 \neq I_2$. Hence, the anti-chain property is violated.

The following result, originally established by Pearce (1997) and later generalised to nested programs by Lifschitz et al. (2001), reveals the close connection between equilibrium models and stable models, showing that stable models are actually a special case of equilibrium models.

*Proposition 3*
For any program $\Pi$, $I$ is a stable model of $\Pi$ iff $\langle I, I\rangle$ is an equilibrium model of $\hat{\Pi}$.

## *2.3 Quantified Propositional Logic*

We now introduce *quantified propositional logic*, an extension of classical propositional logic in which formulas are permitted to contain quantifications over propositional variables. More formally, in addition to the symbols used to construct propositional formulas, the language of quantified propositional logic is assumed to contain the two symbols $\forall$ and $\exists$. The formation rules for constructing formulas of quantified propositional logic are similar to the usual formation rules for propositional formulas, together with the condition that if $\Phi$ is a formula and $p$ is a propositional variable, then $(\forall p\, \Phi)$ and $(\exists p\, \Phi)$ are also formulas. We call $\forall p$ a *universal quantifier* and $\exists p$ an *existential quantifier*, for every variable $p$. Informally, $(\forall p\, \Phi)$ expresses that $\Phi$ is true for *all* truth assignments of $p$, and $(\exists p\, \Phi)$ means that $\Phi$ is true for *some* truth assignment of $p$. We allow the usual convention of omitting parantheses if no ambiguity arises. Formulas of quantified propositional logic are usually referred to as *quantified Boolean formulas* (QBFs) and are denoted by Greek upper-case letters (recall that we use Greek lower-case letters to denote standard propositional formulas). We define the *logical complexity* of a QBF $\Phi$ in accordance to propositional formulas, i.e., $lc(\Phi)$ is given as the number of occurrences of the logical symbols $\forall$, $\exists$, $\neg$, $\vee$, $\wedge$, and $\supset$ in $\Phi$.

The semantics of quantified propositional logic is defined as follows. First, we require some ancillary notation. An occurrence of a variable $p$ in a QBF $\Phi$ is *free* iff it does not appear in the scope of a quantifier $\mathsf{Q}p$ ($\mathsf{Q} \in \{\forall, \exists\}$), otherwise the occurrence of $p$ is *bound.* If $\Phi$ contains no free variables, then $\Phi$ is *closed*, otherwise $\Phi$ is *open.* Furthermore, $\Phi[p_1/\psi_1, \dots, p_n/\psi_n]$ denotes the result of uniformly substituting the free occurrences of variables $p_i$ in $\Phi$ by $\psi_i$ ($1 \leq i \leq n$). This notation is extended to sets of variables and formulas in the obvious way. That is to say, for pairwise disjoint sets $V_i = \{p^i_1, \dots, p^i_{j_i}\}$ of variables and sets $S_i = \{\psi^i_1, \dots, \psi^i_{j_i}\}$ of formulas, for $1 \leq i \leq n$, $\Phi[V_1/S_1, \dots, V_n/S_n]$ stands for

$$\Phi[p^1_1/\psi^1_1, \dots, p^1_{i_1}/\psi^1_{i_1}, \dots, p^n_1/\psi^n_1, \dots, p^n_{i_n}/\psi^n_{i_n}].$$

Given a (classical) interpretation $I$, the *truth value* $\nu_I(\Phi)$ *under* $I$ of a QBF is recursively defined as follows:

1. if $\Phi = \top$, then $\nu_I(\Phi) = 1$;
2. if $\Phi = \bot$, then $\nu_I(\Phi) = 0$;
3. if $\Phi = p$, for some variable $p$, then $\nu_I(\Phi) = 1$ if $p \in I$, and $\nu_I(\Phi) = 0$ otherwise;
4. if $\Phi = \neg\Psi$, then $\nu_I(\Phi) = 1 - \nu_I(\Psi)$;
5. if $\Phi = (\Phi_1 \wedge \Phi_2)$, then $\nu_I(\Phi) = min(\{\nu_I(\Phi_1), \nu_I(\Phi_2)\})$;

6. if $\Phi = (\Phi_1 \vee \Phi_2)$, then $\nu_I(\Phi) = max(\{\nu_I(\Phi_1), \nu_I(\Phi_2)\})$;
7. if $\Phi = (\Phi_1 \supset \Phi_2)$, then $\nu_I(\Phi) = 1$ iff $\nu_I(\Phi_1) \leq \nu_I(\Phi_2)$;
8. if $\Phi = \forall p\, \Psi$, then $\nu_I(\Phi) = min(\nu_I(\Psi[p/\top]), \nu_I(\Psi[p/\bot]))$;
9. if $\Phi = \exists p\, \Psi$, then $\nu_I(\Phi) = max(\nu_I(\Psi[p/\top]), \nu_I(\Psi[p/\bot]))$.

Observe that it obviously holds that

$$\nu_I(\forall p\, \Psi) = \nu_I(\Psi[p/\top] \wedge \Psi[p/\bot]) \quad \text{and} \quad \nu_I(\exists p\, \Psi) = \nu_I(\Psi[p/\top] \vee \Psi[p/\bot]).$$

We say that $\Phi$ is *true under* $I$ if $\nu_I(\Phi) = 1$, otherwise $\Phi$ is *false under* $I$. If $\nu_I(\Phi) = 1$, then $I$ is a *model* of $\Phi$. Likewise, for a set $S$ of QBFs, if $\nu_I(\Phi) = 1$ for all $\Phi \in S$, then $I$ is a model of $S$. If $\Phi$ has some model, then $\Phi$ is said to be *satisfiable*. If $\Phi$ is true under any interpretation, then $\Phi$ is *valid*, symbolically $\models \Phi$. Observe that a closed QBF is either valid or unsatisfiable, because closed QBFs are either true under each interpretation or false under each interpretation. Hence, for closed QBFs, there is no need to refer to particular interpretations.

In the sequel, we employ the following abbreviations in the context of QBFs: Let $S = \{\varphi_1, \ldots, \varphi_n\}$ and $T = \{\psi_1, \ldots, \psi_n\}$ be sets of indexed formulas. Then, $S \leq T$ abbreviates $\bigwedge_{i=1}^n (\varphi_i \supset \psi_i)$, and $S < T$ is a shorthand for $(S \leq T) \wedge \neg(T \leq S)$. Furthermore, for a set $P = \{p_1, \ldots, p_n\}$ of propositional variables and a quantifier $\mathsf{Q} \in \{\forall, \exists\}$, we let $\mathsf{Q}P\, \Phi$ stand for the formula $\mathsf{Q}p_1 \mathsf{Q}p_2 \cdots \mathsf{Q}p_n\, \Phi$.

The operators $\leq$ and $<$ are fundamental tools for expressing certain tests on sets of atoms. In particular, the following properties hold: Let $P = \{p_1, \ldots, p_n\}$ be a set of indexed atoms, and let $I_1 \subseteq P$ and $I_2 \subseteq P$ be two interpretations. Then,

(i) $I_1 \cup I_2'$ is a model of $P \leq P'$ iff $I_1 \subseteq I_2$, and
(ii) $I_1 \cup I_2'$ is a model of $P < P'$ iff $I_1 \subset I_2$.

We also note the following obvious yet central property, which will be relevant later on: Let $\Phi$ be a QBF whose free variables are given by $P$, and let $Q \subseteq P$. Then, an interpretation $I \subseteq P \setminus Q$ is a model of $\exists Q \Phi$ iff there is some $J \subseteq Q$ such that $J \cup I$ is a model of $\Phi$.

Similar to classical first-order logic, there are several results concerning the shifting and renaming of quantifiers. We list some fundamental relations below and refer the interested reader to Egly et al. (2004) and Woltran (2003) for a fuller discussion.

*Proposition 4*
Let $p$ and $q$ be atoms, and $\mathsf{Q} \in \{\forall, \exists\}$. Furthermore, let $\Phi$ and $\Psi$ be QBFs such that $\Psi$ does not contain free occurrences of $p$. Then,

1. $\models (\neg\exists p\, \Phi) \equiv \forall p(\neg\Phi)$,
2. $\models (\neg\forall p\, \Phi) \equiv \exists p(\neg\Phi)$,
3. $\models (\Psi \circ \mathsf{Q}p\, \Phi) \equiv \mathsf{Q}p(\Psi \circ \Phi)$, for $\circ \in \{\wedge, \vee, \supset\}$, and
4. $\models (\mathsf{Q}q\, \Psi) \equiv (\mathsf{Q}p\, \Psi[q/p])$.

A QBF $\Phi$ is in *prenex normal form* iff it is of the form

$$\mathsf{Q}_1 V_1 \mathsf{Q}_2 V_2 \ldots \mathsf{Q}_n V_n\, \phi,$$

where $\phi$ is a propositional formula, $\mathsf{Q}_i \in \{\exists, \forall\}$ such that $\mathsf{Q}_i \neq \mathsf{Q}_{i+1}$ for $1 \leq i \leq$

$n-1$, and $V_1, \ldots, V_n$ are pairwise disjoint sets of propositional variables. Without going into details, we mention that any QBF is easily transformed into an equivalent QBF in prenex normal form (by applying, among other reduction steps, the equivalences depicted in Proposition 4). In fact, this transformation can be carried out in polynomial time.

Historically, among the first logical analyses of systems dealing with quantifiers over propositional variables are the investigations by Russell (1906) ("theory of implication") and by Łukasiewicz and Tarski (1930) ("erweiterter Aussagenkalkül"), not to mention the monumental *Principia Mathematica* (Whitehead and Russell 1913). The particular idea of quantifying propositional variables was extended by Leśniewski (1929) in his *protothetic logic* where variables whose values are *truth functions* are allowed and quantification is defined over these variables (cf. also Srzednicki and Stachniak (1998) for more details about Leśniewski's system).[2] In the beginning of the seventies of the last century, propositional quantification came into the spotlight of computer science—in particular in the new and developing field of complexity theory (Garey and Johnson 1979)—when evaluation problems for QBFs were recognised as the prototypical problems for the *polynomial hierarchy* (Stockmeyer 1976) as well as for the prominent complexity class PSPACE (Meyer and Stockmeyer 1973). Details on this issue are given Section 4.

## 3 Characterisations

We now show how equilibrium logic and nested logic programs can be mapped in polynomial time into QBFs. We first deal with the case of equilibrium logic, providing an encoding which is constructible in *quadratic* time and space. From this, we obtain, as an optimisation, an encoding for logic programs which is constructible in *linear* time and space. Next, we shed some light on the relation between our encodings and circumscription (McCarthy 1980), thereby discussing an extension of a well-known result by Lin (1991). We conclude this section by expressing different notions of equivalence in terms of ordinary and quantified propositional logic, respectively.

### *3.1 Encodings for Equilibrium Logic*

Our first goal is to express satisfiability in the logic of here-and-there in terms of satisfiability in classical logic. We begin with the following translation:

*Definition 1*
Let $\varphi$ be a formula. Then, $\tau[\varphi]$ is recursively defined as follows:

1. if $\varphi$ is an atomic formula, or one of $\top$ or $\bot$, then $\tau[\varphi] = \varphi$,
2. if $\varphi = (\varphi_1 \circ \varphi_2)$, for $\circ \in \{\wedge, \vee\}$, then $\tau[\varphi] = \tau[\varphi_1] \circ \tau[\varphi_2]$,
3. if $\varphi = \neg\psi$, then $\tau[\varphi] = \neg\psi'$, and

[2] A more elaborate overview on these early historical aspects of propositional quantification can be found in §28 of Church (1956).

4. if $\varphi = (\varphi_1 \supset \varphi_2)$, then $\tau[\varphi] = (\tau[\varphi_1] \supset \tau[\varphi_2]) \wedge (\varphi'_1 \supset \varphi'_2)$.

Recall that, as per our convention, primed formulas refer to the result of replacing each atom $p$ occurring in the corresponding unprimed formula by a globally new atom $p'$.

Let us note first that the size of $\tau[\varphi]$ is quadratic in the size of the input formula $\varphi$. More precisely, we have the following property:

*Lemma 1*
Let $\tau[\cdot]$ be the transformation defined above. Then,

1. $lc(\tau[\varphi]) \leq lc(\varphi)(lc(\varphi) + 2)$, for any formula $\varphi$, and
2. $lc(\tau[\varphi]) = lc(\varphi)$, for any expression $\varphi$.

*Proof*
Part 1 is shown by induction on $lc(\varphi)$. Part 2 follows by definition. □

Intuitively, the primed formulas in $\tau[\varphi]$ correspond to formulas evaluated in the world "there", whilst unprimed formulas correspond to formulas evaluated in "here". In order to fully express the semantics of the logic of here-and-there, the only additional requirement needed is to ensure that all formulas true in "here" are also true in "there". This can be conveniently expressed in terms of the condition $V \leq V'$, where $V$ is the set of atoms occurring in $\varphi$. Indeed, we have the following relation:

*Lemma 2*
Let $\varphi$ be a formula and $V = var(\varphi)$. Furthermore, let $I_H, I_T \subseteq V$ be interpretations. Then, $\langle I_H, I_T \rangle$ is an HT-model of $\varphi$ iff $I_H \cup I'_T$ is a model of

$$\mathcal{T}_{HT}[\varphi] = (V \leq V') \wedge \tau[\varphi].$$

*Proof*
See Appendix A. □

Note that, in virtue of Lemma 1, $lc(\mathcal{T}_{HT}[\varphi])$ is quadratic in $lc(\varphi)$, for every formula $\varphi$.

*Example 3*
To illustrate the mechanism of transformation $\mathcal{T}_{HT}[\cdot]$, consider the formula $\varphi = \neg\neg p \supset p$. We already demonstrated in Section 2.1 that $\varphi$ is not valid in the logic of here-and-there, although it is clearly a tautology of classical propositional logic.

Let us first construct $\tau[\varphi]$, which is given by

$$(\tau[\neg\neg p] \supset \tau[p]) \wedge (\neg\neg p' \supset p').$$

Evaluating $\tau[\neg\neg p]$ and $\tau[p]$, we get

$$(\neg\neg p' \supset p) \wedge (\neg\neg p' \supset p'). \tag{2}$$

The first conjunct of (2) is equivalent to $(p' \supset p)$ in classical propositional logic and the second conjunct is a tautology of classical propositional logic. Hence, the entire translation $\mathcal{T}_{HT}[\varphi] = (p \supset p') \wedge \tau[\varphi]$ is equivalent to

$$(p \supset p') \wedge (p' \supset p),$$

which has two models, viz. $I_1 = \emptyset$ and $I_2 = \{p, p'\}$. Therefore, by Lemma 2, the HT-models of $(\neg\neg p \supset p)$ are given by $\langle\emptyset, \emptyset\rangle$ and $\langle\{p\}, \{p\}\rangle$. Observe that $\langle\emptyset, \{p\}\rangle$ is therefore not an HT-model of $\varphi$, which is in accordance with our discussion above.

We proceed with expressing equilibrium models. To retain polynomiality as in the above translation, we have to switch the target language from propositional logic to its quantified pendant.

*Theorem 1*
Let $\varphi$ be a formula, $V = var(\varphi)$, and $I \subseteq V$. Then, $\langle I, I\rangle$ is an equilibrium model of $\varphi$ iff $I'$ is a model of

$$\mathcal{T}_E[\varphi] = \varphi' \wedge \neg\exists V\big((V < V') \wedge \tau[\varphi]\big).$$

*Proof*
We first note that $\mathcal{T}_E[\varphi]$ is obviously equivalent to

$$\varphi' \wedge \neg\exists V\big((V < V') \wedge \mathcal{T}_{HT}[\varphi]\big), \tag{3}$$

because $(V < V')$ is logically equivalent to $(V < V') \wedge (V \leq V')$.

Let $I \subseteq V$ be some interpretation. Recall that $\langle I, I\rangle$ is an equilibrium model of $\varphi$ iff (i) $\langle I, I\rangle$ is an HT-model of $\varphi$, and (ii) for every $J \subset I$, $\langle J, I\rangle$ is not an HT-model of $\varphi$. We show that (i) and (ii) hold iff $I'$ is a model of (3).

First of all, by Proposition 2 and a simple renaming, it follows that Condition (i) holds iff $\varphi'$ is true under $I'$. Now consider the QBF $\Psi = \exists V((V < V') \wedge \mathcal{T}_{HT}[\varphi])$. By the properties of $<$ and the semantics of the existential quantifier, we have that $\Psi$ is true under $I'$ iff there is some $J \subset I$ such that $\mathcal{T}_{HT}[\varphi]$ is true under $J \cup I'$. Hence, invoking Lemma 2, we get that $\Psi$ is true under $I'$ iff Condition (ii) does not hold. Consequently, (ii) holds iff $\neg\Psi$ is true under $I'$. Therefore, (i) and (ii) are jointly satisfied iff $I'$ is a model of $\varphi' \wedge \neg\Psi$. $\square$

Observe that, since $lc(\mathcal{T}_{HT}[\varphi])$ is quadratic in $lc(\varphi)$, the same holds for $\mathcal{T}_E[\varphi]$ as well, for every formula $\varphi$.

*Example 4*
Let us compute the equilibrium models of $\neg\neg p \supset p$ by means of Theorem 1. From Example 3, we already know that $\tau[\varphi]$ is equivalent to $p' \supset p$. Moreover, $V < V'$ stands in the present case for the formula $(p \supset p') \wedge \neg(p' \supset p)$. Hence, $\mathcal{T}_E[\varphi]$ is equivalent to

$$(\neg\neg p' \supset p') \wedge \neg\exists p\Big((p \supset p') \wedge \neg(p' \supset p) \wedge (p' \supset p)\Big). \tag{4}$$

Note that $\neg(p' \supset p) \wedge (p' \supset p)$ makes the whole formula in the scope of $\exists p$ unsatisfiable, so (4) is equivalent to $(\neg\neg p' \supset p')$, which is a tautology of classical logic. Thus, every subset of $V' = \{p'\}$ is a model of $\mathcal{T}_E[\varphi]$, and Theorem 1 implies that the equilibrium models of $(\neg\neg p \supset p)$ are given by $\langle\emptyset, \emptyset\rangle$ and $\langle\{p\}, \{p\}\rangle$.

Having now the encoding $\mathcal{T}_E[\cdot]$ at hand, we easily obtain corresponding encodings for the basic reasoning tasks associated with equilibrium logic. Recall that we identify theories with the conjunction of its (finitely many) elements.

*Corollary 1*
Let $T$ be a theory, $\varphi$ a formula, $V = var(T)$, and $W = var(T \cup \{\varphi\})$. Then,

1. $T$ has an equilibrium model iff $\models \exists V' \, \mathcal{T}_E[T]$,
2. $T \vdash_b \varphi$ iff $\models \exists W' \left(\mathcal{T}_E[T] \wedge \varphi'\right)$, and
3. $T \vdash_s \varphi$ iff $\models \forall W' \left(\mathcal{T}_E[T] \supset \varphi'\right)$.

## *3.2 Encodings for Logic Programs*

In view of Proposition 3, our encodings introduced so far, based on the transformation $\mathcal{T}_E[\cdot]$, yield corresponding encodings for logic programs in a straightforward way. For instance, by Corollary 1, we have that $\Pi$ has a stable model iff $\exists V' \, \mathcal{T}_E[\hat{\Pi}]$ is valid, for $V = var(\Pi)$. These encodings, then, have logical complexities which are quadratic in the size of the input programs. It turns out, however, that we can actually *refine* the transformation $\mathcal{T}_E[\cdot]$, obtaining a transformation for computing the stable models of logic programs which is *linear* in the size of the input programs.

*Theorem 2*
Let $\Pi$ be a logic program, $\hat{\Pi} = \{B(r) \supset H(r) \mid r \in \Pi\}$ the set of formulas associated with $\Pi$, $V = var(\Pi)$, and $I \subseteq V$. Then, $I$ is a stable model of $\Pi$ iff $I'$ is a model of

$$\mathcal{T}_S[\Pi] = \hat{\Pi}' \wedge \neg\exists V\Big((V < V') \wedge \bigwedge_{r \in \Pi} \big(\tau[B(r)] \supset \tau[H(r)]\big)\Big).$$

*Proof*
We show that $\mathcal{T}_S[\Pi]$ is equivalent to

$$\mathcal{T}_E[\hat{\Pi}] = \hat{\Pi}' \wedge \neg\exists V\big((V < V') \wedge \tau[\hat{\Pi}]\big).$$

Then, the result immediately holds by Theorem 1 and Proposition 3.

First of all, by definition of $\tau[\cdot]$,

$$\tau[\hat{\Pi}] = \bigwedge_{r \in \Pi} \Big(\big(\tau[B(r)] \supset \tau[H(r)]\big) \wedge \big(B(r') \supset H(r')\big)\Big).$$

By associativity of $\wedge$, and by identifying $\bigwedge_{r \in \Pi} \big(B(r') \supset H(r')\big)$ with $\hat{\Pi}'$, we have that $\tau[\hat{\Pi}]$ can be written as

$$\bigwedge_{r \in \Pi} \big(\tau[B(r)] \supset \tau[H(r)]\big) \wedge \hat{\Pi}',$$

yielding that $\mathcal{T}_E[\hat{\Pi}]$ is equivalent to

$$\hat{\Pi}' \wedge \neg\exists V\Big[(V < V') \wedge \bigwedge_{r \in \Pi} \big(\tau[B(r)] \supset \tau[H(r)]\big) \wedge \hat{\Pi}'\Big].$$

Since no atom from $V$ occurs in $\hat{\Pi}'$, by applying Proposition 4 repeatedly, the occurrence of $\hat{\Pi}'$ in the scope of $\exists V$ can be moved outside the quantifier, obtaining

$$\hat{\Pi}' \wedge \neg\Big[\exists V\Big((V < V') \wedge \bigwedge_{r\in\Pi} \big(\tau[B(r)] \supset \tau[H(r)]\big)\Big) \wedge \hat{\Pi}'\Big]. \tag{5}$$

Using De Morgan's law, (5) can then be rewritten into

$$\hat{\Pi}' \wedge \Big[\neg\exists V\Big((V < V') \wedge \bigwedge_{r\in\Pi} \big(\tau[B(r)] \supset \tau[H(r)]\big)\Big) \vee \neg\hat{\Pi}'\Big]. \tag{6}$$

By the distributivity law, $\neg\hat{\Pi}'$ finally gets absorbed by the first conjunct of (6). The result of this manipulation is $\mathcal{T}_S[\Pi]$. $\square$

Observe that, for any program $\Pi$, $lc(\mathcal{T}_S[\Pi])$ is linear in $lc(\hat{\Pi})$. This is a direct consequence of Part 2 of Lemma 1, since $B(r)$ and $H(r)$ are expressions, for each $r \in \Pi$.

*Example 5*
For illustration, let us analyse the functioning of $\mathcal{T}_S[\cdot]$ applied to the program $\Pi = \{p \leftarrow (q \wedge r) \vee (\neg q \wedge \neg s)\}$ from Example 1. Then, $\mathcal{T}_S[\Pi]$ is given by

$$\hat{\Pi}' \wedge \neg\exists V\Big[(V < V') \wedge \Big(\big((q \wedge r) \vee (\neg q' \wedge \neg s')\big) \supset p\Big)\Big],$$

where $\hat{\Pi} = \big((q \wedge r) \vee (\neg q \wedge \neg s)\big) \supset p$, $V = \{p, q, r, s\}$, and

$$\begin{aligned}(V < V') = {} & (p \supset p') \wedge (q \supset q') \wedge (r \supset r') \wedge (s \supset s') \wedge \\ & \neg\big((p' \supset p) \wedge (q' \supset q) \wedge (r' \supset r) \wedge (s' \supset s)\big).\end{aligned}$$

First, let us verify that the interpretation $I' = \{p'\}$—which corresponds to the only stable model $\{p\}$ of $\Pi$—is a model of $\mathcal{T}_S[\Pi]$. Clearly, $\hat{\Pi}'$ is true under $I'$, so it remains to check whether

$$\exists V\Big[(V < V') \wedge \Big(\big((q \wedge r) \vee (\neg q' \wedge \neg s')\big) \supset p\Big)\Big] \tag{7}$$

is false under $I'$. By the properties of $<$ and the semantics of the existential quantifier, we have that (7) is true iff there is some $J \subset I$ such that

$$\big((q \wedge r) \vee (\neg q' \wedge \neg s')\big) \supset p \tag{8}$$

is true under $J \cup I'$. Since $I = \{p\}$, the only proper subset of $I$ is the empty set, but (8) is false under $I'$, because the antecedent of (8) is true under $I'$ whereas $p$ is false under $I'$. Hence, (7) is false under $I'$, and consequently $I'$ is a model of $\mathcal{T}_S[\Pi]$.

To verify that no other $I \subseteq V$ is a model of $\mathcal{T}_S[\Pi]$, we first check that no $I'$ with $\{p'\} \subset I' \subseteq \{p', q', r', s'\}$ is a model of $\mathcal{T}_S[\Pi]$. Obviously, such an $I'$ satisfies $\hat{\Pi}'$, but it also satisfies (7), since there is some $J \subset I$, viz. $J = \{p\}$, such that $J \cup I'$ is a model of both $(V < V')$ and (8). So, $I'$ is then not a model of $\mathcal{T}_S[\Pi]$. Finally, observe that the remaining interpretations, $\emptyset$, $\{q'\}$, $\{r'\}$, and $\{s'\}$, are not models of $\mathcal{T}_S[\Pi]$ because they are not models of $\hat{\Pi}'$.

Similar to the case of equilibrium logic, from $\mathcal{T}_S[\cdot]$ we straightforwardly obtain encodings for the basic reasoning tasks in the context of logic programs. It moreover holds that the logical complexities of these encodings are linear in the size of the input programs.

*Corollary 2*
Let $\Pi$ be a logic program, $\varphi$ a formula, $V = \mathit{var}(\Pi)$, and $W = \mathit{var}(\Pi) \cup \mathit{var}(\varphi)$. Then,

1. $\Pi$ has a stable model iff $\models \exists V' \mathcal{T}_S[\Pi]$,
2. $\Pi \vdash_b \varphi$ iff $\models \exists W' \big(\mathcal{T}_S[\Pi] \wedge \varphi'\big)$, and
3. $\Pi \vdash_s \varphi$ iff $\models \forall W' \big(\mathcal{T}_S[\Pi] \supset \varphi'\big)$.

In concluding, let us mention that Theorem 2 generalises a similar QBF encoding put forth by Egly et al. (2000) for the case of disjunctive logic programs. There, the following QBF was used to express the stable models of a given disjunctive logic program $\Pi$:

$$\mathcal{T}_{DLP}[\Pi] = \hat{\Pi} \wedge \neg\exists V'\Big[(V' < V) \wedge \bigwedge_{r\in\Pi} \Big(\big(B^+(r') \wedge B^-(r)\big) \supset H(r')\Big)\Big].$$

Here, $B^+(r)$ denotes the conjunction of all positive literals in $B(r)$, and $B^-(r)$ denotes the conjunction of all negative literals in $B(r)$. It holds that, for any $I \subseteq V$, $I$ is a stable model of $\Pi$ iff $I$ is a model of $\mathcal{T}_{DLP}[\Pi]$.

It is easily seen that, given a disjunctive logic program $\Pi$, the transformation $\mathcal{T}_S[\Pi]$ coincides with $\mathcal{T}_{DLP}[\Pi]$, providing the priming of formulas is interchanged.

## *3.3 Relation to Circumscription*

By using the language of quantified propositional logic, we were able to conveniently express the inherent minimality postulates of both equilibrium models and stable models in terms of the operator $<$ and existential quantification. In this section, we shed some further light into the minimisation conditions of equilibrium models and stable models by comparing them to *circumscription* (McCarthy 1980), a well known technique to realise minimal-model reasoning.

For the case of disjunctive logic programs, an early result about the relation between stable models and circumscription was given by Lin (1991). Our subsequent discussion reveals that Lin's result holds for nested programs and equilibrium logic as well.

Originally, as put forth by McCarthy (1980), circumscription is defined as a special formula schema of second-order logic. In the propositional case, which is the relevant setting for our purposes here, circumscription is actually a formula of quantified propositional logic and can be defined in the following way (cf. also Lifschitz (1994)).

Let $T$ be a theory and $(P, Q, Z)$ a partition of $\mathit{var}(T)$. For two (classical) models, $I$ and $J$, of $T$, we define $I \leq_{P;Z} J$ iff

1. $(I \cap Q) = (J \cap Q)$, and
2. $(I \cap P) \subseteq (J \cap Q)$.

A model $I$ of $T$ is $(P; Z)$-*minimal* iff no model $J$ of $T$ with $J \neq I$ satisfies $J \leq_{P;Z} I$.

Informally, the partition $(P, Q, Z)$ can be interpreted as follows: The set $P$ contains the variables to be minimised, $Z$ are those variables that can vary in minimising $P$, and the remaining variables $Q$ are fixed in minimising $P$.

Let $T$ be a theory and $(P, Q, Z)$ a partition of $var(T)$, where $P = \{p_1, \ldots, p_n\}$ and $Z = \{z_1, \ldots, z_m\}$. The set of $(P; Z)$-minimal models of $T$ is given by the models of the QBF

$$\mathrm{CIRC}(T; P; Z) = T \wedge \neg\exists\tilde{P}\,\exists\tilde{Z}\Big((\tilde{P} < P) \wedge T[P/\tilde{P}, Z/\tilde{Z}]\Big),$$

where $\tilde{P} = \{\tilde{p}_1, \ldots, \tilde{p}_n\}$ and $\tilde{Z} = \{\tilde{z}_1, \ldots, \tilde{z}_m\}$ are sets of new variables corresponding to $P$ and $Z$, respectively. In what follows, we write $\mathrm{CIRC}(T; P)$ for $\mathrm{CIRC}(T; P; \emptyset)$.

The next result paraphrases the characterisation of Lin (1991).

*Proposition 5*

Let $\Pi$ be a disjunctive logic program, $V = var(\Pi)$, and $I \subseteq V$. Then, $I$ is a stable model of $\Pi$ iff $I \cup I'$ is a model of

$$\bigwedge_{p \in V} (p \equiv p') \wedge \mathrm{CIRC}(\bigwedge_{r \in \Pi} \big((B^+(r) \wedge B^-(r')) \supset H(r)\big); V), \tag{9}$$

where $B^+(r)$ is the conjunction of all positive literals in $B(r)$ and $B^-(r)$ is the conjunction of all negative literals in $B(r)$.

Using our terminology, it is quite obvious that $\bigwedge_{r\in\Pi} \big((B^+(r) \wedge B^-(r')) \supset H(r)\big)$ is given by $\bigwedge_{r\in\Pi} \big(\tau[B(r)] \supset \tau[H(r)]\big)$. Furthermore, in analogy to our operator $\leq$, let us abbreviate, for all sets $S = \{\varphi_1, \ldots, \varphi_n\}$ and $T = \{\psi_1, \ldots, \psi_n\}$ of formulas, $\bigwedge_{i=1}^{n} (\varphi_i \equiv \psi_i)$ by $S = T$. Then, we can rewrite (9) into the following QBF:

$$(V = V') \wedge \mathrm{CIRC}(\bigwedge_{r \in \Pi} \big(\tau[B(r)] \supset \tau[H(r)]\big); V).$$

As the next result demonstrates, this formula schema can actually be used to characterise the stable models of *arbitrary* nested logic programs, and not only disjunctive ones, thus generalising Proposition 5.

*Theorem 3*

Let $\Pi$ be a logic program, $V = var(\Pi)$, and $I \subseteq V$. Then, $I$ is a stable model of $\Pi$ iff $I \cup I'$ is a model of

$$(V = V') \wedge \mathrm{CIRC}(\bigwedge_{r \in \Pi} \big(\tau[B(r)] \supset \tau[H(r)]\big); V). \tag{10}$$

*Proof*

In view of Theorem 2, we know that $I$ is a stable model of $\Pi$ iff $I'$ is a model of

$$\mathcal{T}_S[\Pi] = \hat{\Pi}' \wedge \neg\exists V\Big((V < V') \wedge \bigwedge_{r \in \Pi} \big(\tau[B(r)] \supset \tau[H(r)]\big)\Big).$$

Now, clearly, $I'$ is a model of $\mathcal{T}_S[\Pi]$ iff $I \cup I'$ is a model of $(V = V') \wedge \mathcal{T}_S[\Pi]$. It thus suffices to show that the latter formula is equivalent to (10). This can be seen as follows.

By the definition of circumscription, formula (10) is the following QBF:

$$(V = V') \wedge \tau^\star[\Pi] \wedge \neg\exists \widetilde{V}\Big((\widetilde{V} < V) \wedge (\tau^\star[\Pi])[V/\widetilde{V}]\Big), \tag{11}$$

where $\tau^\star[\Pi] = \bigwedge_{r\in\Pi}\big(\tau[B(r)] \supset \tau[H(r)]\big)$. Observe that the atoms from $V'$ are *not* among the atoms renamed by $(\tau^\star[\Pi])[V/\widetilde{V}]$. Since furthermore the formula $(V = V')$ enforces that the same truth value is assigned to $p$ and $p'$, for each $p \in V$, QBF (11) is obviously equivalent to

$$(V = V') \wedge \tau^\star[\Pi] \wedge \neg\exists \widetilde{V}\Big((\widetilde{V} < V') \wedge (\tau^\star[\Pi])[V/\widetilde{V}]\Big), \tag{12}$$

by using $V'$ instead of $V$ in the second argument of $<$. Now, in the presence of $(V = V')$, $\tau^\star[\Pi]$ is equivalent to $\hat{\Pi}'$. Furthermore, no atom from $V$ occurs in

$$\neg\exists \widetilde{V}\Big((\widetilde{V} < V') \wedge (\tau^\star[\Pi])[V/\widetilde{V}]\Big).$$

Therefore, we can replace the existentially quantified variables $\widetilde{V}$ by $V$, from which we obtain that (12) is equivalent to

$$(V = V') \wedge \hat{\Pi}' \wedge \neg\exists V\Big((V < V') \wedge \tau^\star[\Pi]\Big),$$

which is $(V = V') \wedge \mathcal{T}_S[\Pi]$. We thus showed that the latter formula is indeed equivalent to (10). $\square$

We can extend Theorem 3, in turn, to full equilibrium logic in the following fashion:

*Theorem 4*
Let $\varphi$ be a formula, $V = var(\varphi)$, and $I \subseteq V$. Then, $\langle I, I\rangle$ is an equilibrium model of $\varphi$ iff $I \cup I'$ is a model of

$$(V = V') \wedge \mathrm{CIRC}(\tau[\varphi]; V). \tag{13}$$

*Proof*
The proof proceeds along the same line of reasoning as the proof of Theorem 3, except by resorting to Theorem 1 and by showing that $(V = V') \wedge \mathcal{T}_E[\Pi]$ is equivalent to (13). $\square$

Let us mention that, prior to Lin (1991), the relation between default negation and circumscription has already been investigated by Gelfond et al. (1989; 1990). In particular, they show that the stable models of programs with a restricted use of negation (viz. with *stratified negation*) can be characterised via the minimal models of a more involved method of circumscription, called *iterated circumscription.* Incidentally, a similar result, albeit in terms of *prioritised circumscription*, is given by Baral (2003). Thus, loosely speaking, in order to relate stable models with circumscription, the presence of default negation calls either for a suitable renaming of variables or an adaption of the original circumscription technique.

However, in the case of disjunctive programs without negation, it is well known (Baral 2003) that the stable models of a program coincide with its minimal models. We extend this observation to nested logic programs as follows: Let us call a program *positive* if the expressions in the bodies and heads of its rules do not contain any negation. Now, for any expression $\phi$ without negation, it holds that $\tau[\phi] = \phi$. Hence, for any positive program $\Pi$ with $var(\Pi) = V$, the encoding (10) from Theorem 3 reduces to

$$(V = V') \wedge \mathrm{CIRC}(\bigwedge_{r \in \Pi} (B(r) \supset H(r)); V),$$

which is just

$$(V = V') \wedge \mathrm{CIRC}(\hat{\Pi}; V).$$

Since no primed variables occur in $\mathrm{CIRC}(\hat{\Pi}; V)$ anymore, we can safely drop the first conjunct $(V = V')$ and get as result a purely circumscriptive theory, with all atoms being minimised. We can thus state:

*Theorem 5*
For any positive program $\Pi$ with $var(\Pi) = V$, the stable models of $\Pi$ are given by $\mathrm{CIRC}(\hat{\Pi}; V)$.

In summarising, the discussion in this section thus shows that our encodings conveniently allow us to extend well-known results for disjunctive programs to the general nested case (and even to full equilibrium logic).

### *3.4 Characterising Different Notions of Equivalence*

We now turn our attention to different notions of equivalence which have been studied recently in the context of equilibrium logic and nested logic programs. In the same fashion as before, we first deal with the general case—that is to say, with theories in equilibrium logic—and afterwards, as a special case, with nested logic programs.

In classical logic, the well known *replacement property* holds, according to which any formula $\varphi$ occurring as a specified part in a formula $C_\varphi$ can be replaced by any logically equivalent formula $\psi$ yielding a formula $C_\psi$ which is still logically equivalent to $C_\varphi$.[3] In a nonmonotonic setting, however, when interpreting logical equivalence in the traditional way as the relation which holds in case two theories have the same "intended models"(like, two theories in equilibrium logic have the same equilibrium models or two logic programs have the same stable models), such a property fails in general. Thus, in order to ensure the validity of a replacement property in nonmonotonic formalisms, more robust notions of equivalence have to be considered. In fact, a suitable notion to that effect is *strong equivalence*, first introduced and studied by Lifschitz et al. (2001) for nested logic programs and equilibrium logic. A weaker variant of strong equivalence is *uniform equivalence*,

[3] The replacement property is also known as substitution *salva veritate* in view of Leibniz's principle "eadem sunt, quorum unum potest substitui alteri salva veritate".

which was discussed by Eiter and Fink (2003) for disjunctive logic programs and by Pearce and Valverde (2004c) for nested programs and equilibrium logic.

In what follows, we characterise strong and uniform equivalence, along with the "traditional" concept of equivalence, in terms of quantified propositional logic. In fact, strong equivalence will be captured by means of ordinary classical logic, i.e., without requiring any quantifier.

We start with formally defining the equivalence relations under consideration. Let $T_1$ and $T_2$ be theories. Then,

1. $T_1$ and $T_2$ are (*ordinarily*) *equivalent*, in symbols $T_1 \equiv_o T_2$, iff they possess the same equilibrium models,
2. $T_1$ and $T_2$ are *uniformly equivalent*, in symbols $T_1 \equiv_u T_2$, iff, for every set $R$ of atoms, $T_1 \cup R \equiv_o T_2 \cup R$, and
3. $T_1$ and $T_2$ are *strongly equivalent*, in symbols $T_1 \equiv_s T_2$, iff, for every theory $S$, $T_1 \cup S \equiv_o T_2 \cup S$.

In case of programs, we get analogous relations by setting, for every program $\Pi_1$ and $\Pi_2$, $\Pi_1 \equiv_e \Pi_2$ iff $\hat{\Pi}_1 \equiv_e \hat{\Pi}_2$, for $e \in \{o, s, u\}$. Hence, we have that $\Pi_1 \equiv_u \Pi_2$ iff, for every program $\Pi_3$ containing facts only, $\Pi_1 \cup \Pi_3 \equiv_o \Pi_2 \cup \Pi_3$. As well, $\Pi_1 \equiv_s \Pi_2$ iff, for every program $\Pi_3$, $\Pi_1 \cup \Pi_3 \equiv_o \Pi_2 \cup \Pi_3$.

Clearly, strong equivalence implies uniform equivalence, which in turn implies ordinary equivalence. However, it is a straightforward matter to verify that the implications in the other direction fail and therefore all three notions are distinct. This already holds for the special case of logic programs, as shown by the following example.

*Example 6*
Consider the programs $\{p \leftarrow\}$ and $\{p \leftarrow \neg q\}$, which are easily verified to be ordinarily equivalent but not uniformly so (just add the fact $q \leftarrow$ to each program). Similarly, strong and uniform equivalence are distinct in view of the programs

$$\Pi_1 = \{p \vee q \leftarrow\} \quad \text{and} \quad \Pi_2 = \{p \leftarrow \neg q,\ q \leftarrow \neg p\}.$$

Indeed, both programs have $\{p\}$ and $\{q\}$ as their stable models, so they are ordinarily equivalent. But $\Pi_1$ and $\Pi_2$ are not strongly equivalent, since adding rules

$$\{p \leftarrow q,\ q \leftarrow p\}$$

to $\Pi_1$ yields a program whose only stable model is $\{p, q\}$, whereas adding the same rules to $\Pi_2$ results in a program having no stable model at all. However, $\Pi_1$ and $\Pi_2$ are uniformly equivalent, as can be seen as follows. Adding the fact $p \leftarrow$ to both programs leaves the resultant programs equivalent, with $\{p\}$ as their single stable model. Analogously, equivalence is preserved when fact $q \leftarrow$ is added; and, similarly, if both $p \leftarrow$ and $q \leftarrow$ are added as new facts. In the latter case, both extended programs have the single stable model $\{p, q\}$. Clearly, adding any other fact just adds the respective atom to the stable models of both programs.

In what follows, we recall some important characterisations forming the basis for our subsequent encodings of strong and uniform equivalence, respectively. To begin with, the following property is the central result of Lifschitz et al. (2001):

*Proposition 6*
Two theories are strongly equivalent iff they are equivalent in the logic of here-and-there.

It is worth noting that de Jongh and Hendriks (2003) showed that the weakest intermediate logic that can replace here-and-there in the characterisation of strong equivalence for nested logic programs is the logic KC of weak excluded middle (Kowalski 1968). The logic KC is axiomatised by intuitionistic logic together with the schema $\neg\varphi \vee \neg\neg\varphi$. Since HT is actually the greatest super-intuitionistic logic capturing strong equivalence, the results by Hendriks and de Jongh show that all and only the super-intuitionistic logics lying between KC and HT capture strong equivalence for nested programs.

Proposition 6 was also reformulated for the case of logic programs by Turner (2001; 2003), replacing equivalence in the logic HT by the condition that two programs have the same *SE-models.* The latter are, like HT-models, ordered pairs of interpretations, but they are defined in terms of the reduct of a program instead of referring to HT explicitly. In fact, as shown by Turner (2003), HT-models and SE-models are identical concepts for the programs under consideration (i.e., for programs without strong negation).

In subsequent work, Eiter and Fink (2003) used SE-models to characterise uniform equivalence for disjunctive logic programs. Their result was in turn extended by Pearce and Valverde (2004c) to nested logic programs (and indeed to equilibrium logic) using HT-models. To formulate this result, we introduce the following notation: For all theories $T_1$ and $T_2$, let $U(T_1, T_2)$ be the set consisting of all HT-interpretations $\langle I_H, I_T\rangle$, where $I_H, I_T \subseteq var(T_1 \cup T_2)$, such that whenever $\langle I_H, I_T\rangle$ is a non-total HT-model of $T_1$, then there is an interpretation $J$ such that $I_H \subseteq J \subset I_T$ and $\langle J, I_T\rangle$ is an HT-model of $T_2$. Then, the characterisation of Pearce and Valverde (2004c) can be formulated as follows:

*Proposition 7*
Two theories $T_1$ and $T_2$ are uniformly equivalent iff

(i) $T_1$ and $T_2$ are equivalent in classical logic and
(ii) every HT-interpretation $\langle I_H, I_T\rangle$, with $I_H, I_T \subseteq var(T_1 \cup T_2)$, belongs to $U(T_1, T_2) \cap U(T_2, T_1)$.

We now describe the encodings of the equivalence notions under consideration in terms of quantified propositional logic. To begin with, we give a characterisation of the set $U(T_1, T_2)$, for every theory $T_1$ and $T_2$, used for the encoding of checking uniform equivalence.

In what follows, we denote by $\tau''[\cdot]$ the mapping obtained from the translation $\tau[\cdot]$ of Definition 1 by replacing each unprimed atom $v$ in $\tau[\cdot]$ by a globally new atom $v''$. Recall that, as per our priming convention, for every set $V$ of atoms, $V'$ and $V''$ are defined as the sets $\{v' \mid v \in V\}$ and $\{v'' \mid v \in V\}$, respectively, being disjoint from $V$.

*Lemma 3*
Let $T_1$ and $T_2$ be theories, $V = var(T_1 \cup T_2)$, and $I_H, I_T \subseteq V$ interpretations. Then, $\langle I_H, I_T \rangle \in U(T_1, T_2)$ iff $I_H \cup I'_T$ is a model of

$$\mathcal{M}[T_1, T_2] = \big((V < V') \wedge \tau[T_1]\big) \supset \exists V''\big((V \leq V'') \wedge (V'' < V') \wedge \tau''[T_2]\big).$$

*Proof*
We show the following two properties:

($\alpha$) $\langle I_H, I_T \rangle$ is a non-total HT-model of $T_1$ iff $I_H \cup I'_T$ is a model of

$$\phi = (V < V') \wedge \tau[T_1].$$

($\beta$) $\langle J, I_T \rangle$ is an HT-model of $T_2$ with $I_H \subseteq J \subset I_T$ iff $I_H \cup I'_T \cup J''$ is a model of

$$\psi = (V \leq V'') \wedge (V'' < V') \wedge \tau''[T_2].$$

From this, by the semantics of the conditional $\supset$ and of existential quantification, as well as by the definition of $U(T_1, T_2)$, the claim of the lemma is an immediate consequence.

We start with proving Property ($\alpha$). By Lemma 2, we have that $\langle I_H, I_T \rangle$ is an HT-model of $T_1$ iff $I_H \cup I'_T$ is a model of $(V \leq V') \wedge \tau[T_1]$. Moreover, $\langle I_H, I_T \rangle$ is non-total iff $I_H \subset I_T$. So, by the semantics of $<$, $\langle I_H, I_T \rangle$ is non-total iff $I_H \cup I'_T$ is a model of $V < V'$. It follows that $\langle I_H, I_T \rangle$ is a non-total HT-model of $T_1$ iff $I_H \cup I'_T$ is a model of

$$(V < V') \wedge (V \leq V') \wedge \tau[T_1]. \tag{14}$$

But (14) is clearly equivalent to $(V < V') \wedge \tau[T_1]$, and thus Property ($\alpha$) holds.

We continue with the proof of Property ($\beta$). By Property ($\alpha$), it holds that $\langle J, I_T \rangle$ is an HT-model of $T_2$ such that $J \subset I_T$ iff $J \cup I'_T$ is a model of $(V < V') \wedge \tau[T_2]$. By a simple renaming we get that the latter holds iff $J'' \cup I'_T$ is a model of $(V'' < V') \wedge \tau''[T_2]$. Furthermore, the semantics of $<$ tells us that $I_H \subseteq J$ iff $I_H \cup J'$ is a model of $V \leq V'$. Applying again a simple renaming, we get that $I_H \subseteq J$ iff $I_H \cup J''$ is a model of $V \leq V''$. Combining these conditions, we arrive that $\langle J, I_T \rangle$ is an HT-model of $T_2$ satisfying $I_H \subseteq J \subset I_T$ precisely when (i) $I_H \cup J''$ is a model of $V \leq V''$ and (ii) $J'' \cup I'_T$ is a model of $(V'' < V') \wedge \tau''[T_2]$. But $V \leq V''$ contains no atoms from $V'$ and $(V'' < V') \wedge \tau''[T_2]$ contains no atoms from $V$, so, since $V$, $V'$, and $V''$ are pairwise distinct, it follows that $\langle J, I_T \rangle$ is an HT-model of $T_2$ satisfying $I_H \subseteq J \subset I_T$ iff $I_H \cup I'_T \cup J''$ is a model of

$$(V \leq V'') \wedge (V'' < V') \wedge \tau''[T_2],$$

which proves Property ($\beta$). $\square$

We are now prepared to state the main result of this section:

*Theorem 6*
Let $T_1$ and $T_2$ be theories and $V = var(T_1 \cup T_2)$. Then,

(i) $T_1 \equiv_o T_2$ iff $\models \forall V'(\mathcal{T}_E[T_1] \equiv \mathcal{T}_E[T_2])$,
(ii) $T_1 \equiv_u T_2$ iff $\models \forall V \forall V'\big((T_1 \equiv T_2) \wedge \mathcal{M}[T_1, T_2] \wedge \mathcal{M}[T_2, T_1]\big)$, and
(iii) $T_1 \equiv_s T_2$ iff $\models \forall V \forall V'\big((V \leq V') \supset (\tau[T_1] \equiv \tau[T_2])\big)$.

*Proof*
Concerning Part (i), from Theorem 1 we know that for every $I \subseteq V$, $\langle I, I\rangle$ is an equilibrium model of $T_i$ iff $I'$ is a model of $\mathcal{T}_E[T_i]$, for $i \in \{1,2\}$. Hence, $T_1$ and $T_2$ possess the same equilibrium models, i.e., $T_1 \equiv_o T_2$ holds, iff $\forall V'(\mathcal{T}_E[T_1] \equiv \mathcal{T}_E[T_2])$ is valid.

Now consider Part (ii). By Proposition 7, $T_1$ and $T_2$ are uniformly equivalent iff (a) $T_1$ and $T_2$ are equivalent in classical logic and (b) every HT-interpretation $\langle I_H, I_T\rangle$ with $I_H, I_T \subseteq \mathit{var}(T_1 \cup T_2)$ belongs to $U(T_1,T_2) \cap U(T_2,T_1)$. Condition (a) is clearly equivalent to the fact that $\models \forall V \forall V'(T_1 \equiv T_2)$, and, by Lemma 3, Condition (b) holds precisely in case that $\models \forall V \forall V'(\mathcal{M}[T_1,T_2] \wedge \mathcal{M}[T_2,T_1])$. So, (a) and (b) jointly hold iff

$$\forall V \forall V'(T_1 \equiv T_2) \wedge \forall V \forall V'(\mathcal{M}[T_1,T_2] \wedge \mathcal{M}[T_2,T_1]) \tag{15}$$

is valid. But (15) is easily seen to be equivalent to

$$\forall V \forall V'\big((T_1 \equiv T_2) \wedge \mathcal{M}[T_1,T_2] \wedge \mathcal{M}[T_2,T_1]\big),$$

which establishes Part (ii).

It remains to show Part (iii). By Proposition 6, $T_1$ and $T_2$ are strongly equivalent iff the HT-models of $T_1$ and $T_2$ coincide. By Lemma 2, the latter is the case iff $\mathcal{T}_{HT}[T_1]$ and $\mathcal{T}_{HT}[T_2]$ are logically equivalent in classical logic, which in turn is equivalent to the condition that

$$\forall V V'\big(\mathcal{T}_{HT}[T_1] \equiv \mathcal{T}_{HT}[T_2]\big) \tag{16}$$

is valid in classical logic. Now, by the definition of $\mathcal{T}_{HT}[\cdot]$, Formula (16) is given by

$$\forall V V'\Big(\big((V' \leq V) \wedge \tau[T_1]\big) \equiv \big((V' \leq V) \wedge \tau[T_2]\big)\Big).$$

By simple manipulations in classical logic, the latter formula can be transformed in an equivalence-preserving way into

$$\forall V \forall V'\big((V \leq V') \supset (\tau[T_1] \equiv \tau[T_2])\big).$$

Thus, Part (iii) holds. $\square$

Note that the logical complexities of all encodings from Theorem 6 are quadratic in the logical complexities of the two theories compared. Furthermore, the encodings for ordinary and uniform equivalence possess one quantifier alternation, with a leading prefix of universal quantifiers, while the encoding for strong equivalence amounts to a validity test in classical propositional logic. We have to say more about the consequences of these properties in the next section.

Clearly, for comparing programs rather than theories, we can make use of the above encodings as well, just by taking $\hat{\Pi}$ for each program $\Pi$ as the respective argument. However, for testing ordinary equivalence, we can obtain a more compact encoding by directly resorting to $\mathcal{T}_S[\cdot]$ instead of $\mathcal{T}_E[\cdot]$. The corollary below summarises these observations:

*Corollary 3*
Let $\Pi_1$ and $\Pi_2$ be logic programs and $V = \mathit{var}(\Pi_1 \cup \Pi_2)$. Then,

1. $\Pi_1 \equiv_o \Pi_2$ iff $\models \forall V'(\mathcal{T}_S[\Pi_1] \equiv \mathcal{T}_S[\Pi_2])$,
2. $\Pi_1 \equiv_u \Pi_2$ iff $\models \forall V \forall V'\Big((\hat{\Pi}_1 \equiv \hat{\Pi}_2) \wedge \mathcal{M}[\hat{\Pi}_1, \hat{\Pi}_2] \wedge \mathcal{M}[\hat{\Pi}_2, \hat{\Pi}_1]\Big)$, and
3. $\Pi_1 \equiv_s \Pi_2$ iff $\models \forall V \forall V'\Big((V \leq V') \supset (\tau[\hat{\Pi}_1] \equiv \tau[\hat{\Pi}_2])\Big)$.

We remark that the logical complexities of all encodings in Corollary 3 are *linear* in $lc(\hat{\Pi}_1)$ and $lc(\hat{\Pi}_2)$. While this is rather obvious as far as the encoding for ordinary equivalence is concerned, since we already know that $lc(\mathcal{T}_S[\Pi])$ is linear in $lc(\hat{\Pi})$, for every program $\Pi$, for the other encodings this follows from the observation that $\hat{\Pi}$ does not involve nested implications and thus one can show that $lc(\tau[\hat{\Pi}])$ is still linear in $lc(\hat{\Pi})$, for every program $\Pi$.

Our above encoding for uniform equivalence is based on the characterisation given by Proposition 7. In previous work, Woltran (2004) showed that checking uniform equivalence between disjunctive logic programs can be reduced to checking ordinary equivalence between those kinds of programs. We now generalise this result to arbitrary theories of equilibrium logic, thus providing an alternative encoding for checking uniform equivalence. We start with extending the relevant result from Woltran (2004) to the case of arbitrary theories. As a preparatory step, we give a slight paraphrase of a result due to Ferraris (2005), which in turn is a generalisation of the well-known *Splitting-Set Theorem* for disjunctive logic programs (Lifschitz and Turner 1994; Eiter et al. 1997) to the case of equilibrium logic.

*Proposition 8* (*Splitting-Set Theorem for Equilibrium Logic*)
Let $S$ and $R$ be two theories such that each variable from $var(S)$ occurs only in antecedents of implications or negated in $S$. Then, $\langle I, I\rangle$ is an equilibrium model of $S \cup R$ iff

(i) $\langle I \cap var(S), I \cap var(S)\rangle$ is an equilibrium model of $S$ and
(ii) $\langle I, I\rangle$ is an equilibrium model of $(I \cap var(S)) \cup R$.

In what follows, for every set $V$ of variables, define $V^\circ = \{v^\circ \mid v \in V\}$ as a set of new variables corresponding to the variables in $V$.

*Lemma 4*
Let $T_1$ and $T_2$ be theories and $V = var(T_1 \cup T_2)$. Then, $T_1 \equiv_u T_2$ iff $T_1^\sharp \equiv_o T_2^\sharp$, where

$$T_i^\sharp = T_i \cup \{\neg\neg v^\circ \supset v^\circ,\, v^\circ \supset v \mid v \in V\},$$

for $i = 1, 2$.

*Proof*
The proof relies on the Splitting-Set Theorem for equilibrium logic. Define

$$S = \{\neg\neg v^\circ \supset v^\circ \mid v \in V\} \quad \text{and} \quad R_i = \{v^\circ \supset v \mid v \in V\} \cup T_i,$$

for $i = 1, 2$. Clearly, we have that $T_i^\sharp = S \cup R_i$, for $i = 1, 2$. Now, since all variables from $S$ occur in $R_1$ and $R_2$ only in implications of the form $v^\circ \supset v$, by the Splitting-Set Theorem we get the following property:

($\alpha$) for every interpretation $I$, $\langle I, I\rangle$ is an equilibrium model of $T_i^\sharp$ iff $\langle I \cap V^\circ, I \cap V^\circ\rangle$ is an equilibrium model of $S$ and $\langle I, I\rangle$ is an equilibrium model of $(I \cap V^\circ) \cup R_i$, for $i = 1, 2$.

Furthermore, we require the following two properties, which are easily verified:

($\beta$) The set of all equilibrium models of $S$ is given by $\{\langle J, J\rangle \mid J \subseteq V^\circ\}$.

($\gamma$) For any $F \subseteq V$ and $i = 1, 2$, if $\langle I, I\rangle$ is an equilibrium model of $T_i \cup F$, then $\langle I \cup F^\circ, I \cup F^\circ\rangle$ is an equilibrium model of $R_i \cup F^\circ$, and if $\langle J, J\rangle$ is an equilibrium model of $R_i \cup F^\circ$ then, $\langle V \cap J, V \cap J\rangle$ is an equilibrium model of $T_i \cup F$.

We proceed with the proof of the main result. By definition, $T_1 \equiv_u T_2$ iff

$$T_1 \cup F \equiv_o T_2 \cup F, \text{ for every set } F \text{ of atoms.} \tag{17}$$

Obviously, (17) is equivalent to the condition that

$$T_1 \cup F \equiv_o T_2 \cup F, \text{ for every } F \subseteq V. \tag{18}$$

However, Condition ($\gamma$) implies that, for every $F \subseteq V$, $T_1 \cup F \equiv_o T_2 \cup F$ iff $R_1 \cup F^\circ \equiv_o R_2 \cup F^\circ$. Hence, (18) is equivalent to

$$R_1 \cup F^\circ \equiv_o R_2 \cup F^\circ, \text{ for every } F \subseteq V. \tag{19}$$

Now, in view of Conditions ($\alpha$) and ($\beta$), it is a straightforward matter to check that (19) holds precisely in case that $R_1 \cup S \equiv_o R_2 \cup S$. But the latter relation just states that $T_1^\sharp \equiv_o T_2^\sharp$. So, in summary, the above chain of equivalences shows that $T_1 \equiv_u T_2$ iff $T_1^\sharp \equiv_o T_2^\sharp$. $\Box$

Exploiting the encoding of ordinary equivalence, as given by Theorem 6, we arrive at the following characterisation:

*Theorem 7*
Let $T_1$ and $T_2$ be theories and $V = var(T_1 \cup T_2)$. Furthermore, let $T_1^\sharp$ and $T_2^\sharp$ be defined as in Lemma 4. Then, $T_1 \equiv_u T_2$ iff $\models \forall V'(\mathcal{T}_E[T_1^\sharp] \equiv \mathcal{T}_E[T_2^\sharp])$.

A similar result can be shown for logic programs; we omit the obvious details.

In concluding this section, we compare our characterisation for testing strong equivalence with one by Lin (2002), which is devised for disjunctive logic programs, reducing the test of strong equivalence to checking entailment in classical propositional logic.[4] More specifically, Lin assigns to each disjunctive logic program $\Pi$ a set $\Gamma(\Pi)$ of formulas of classical propositional logic containing for each $r \in \Pi$ the two formulas

$$(B^+(r) \wedge B^-(r')) \supset H(r) \quad \text{and} \quad (B^+(r') \wedge B^-(r')) \supset H(r'),$$

where $B^+(r)$ is the conjunction of all positive literals in $B(r)$ and $B^-(r)$ is the

[4] Lin's result was developed independently from our previous preliminary report Pearce et al. (2001), where the characterisation for strong equivalence between theories of equilibrium logic or nested logic programs, respectively, in terms of classical logic was first reported. Also, Lin erroneously states in his discussion of our characterisation that we do not handle disjunctions.

conjunction of all negative literals in $B(r)$. Then, two programs, $\Pi_1$ and $\Pi_2$, with $V = var(\Pi_1 \cup \Pi_2)$, are strongly equivalent iff the following two assertions hold:

$$\{v \supset v' \mid v \in V\} \cup \Gamma(\Pi_1) \models \Gamma(\Pi_2), \tag{20}$$

$$\{v \supset v' \mid v \in V\} \cup \Gamma(\Pi_2) \models \Gamma(\Pi_1). \tag{21}$$

In terms of our notation, (20) and (21) can be restated thus:

$$\{(V \leq V') \wedge \tau[\hat{\Pi}_1]\} \models \tau[\hat{\Pi}_2], \tag{22}$$

$$\{(V \leq V') \wedge \tau[\hat{\Pi}_2]\} \models \tau[\hat{\Pi}_1]. \tag{23}$$

But these two assertions are just reformulations of our characterisation from Corollary 3. Indeed, by applying the deduction theorem from classical logic and some simple manipulations, (22) and (23) are equivalent to

$$\models (V \leq V') \supset \tau[\hat{\Pi}_1] \equiv \tau[\hat{\Pi}_2].$$

Adjoining universal quantifications for all variables in the above formula yields our characterisation. Hence, our encoding directly generalises Lin's method.[5]

## 4 Complexity

In the previous section, we discussed encodings of the basic reasoning tasks associated with equilibrium logic and nested logic programs, viz. we dealt with the consistency problem as well as with brave and skeptical reasoning for both languages. Additionally, we also captured the problem of checking ordinary, strong, and uniform equivalence between theories or programs. Now, we analyse the computational complexity of these tasks.

A particular advantage of our encoding technique is that the quantifier structure of our encodings yield in a direct manner upper complexity bounds for the corresponding problems. This follows by invoking well-known complexity results about QBFs and by observing that our translations are constructible in polynomial time. Moreover, for each of the upper bounds obtained in this fashion, we also show that they are *strict*, i.e., they possess a matching lower bound. The results presented here generalise well-known complexity results for disjunctive logic programs under the stable model semantics (Eiter and Gottlob 1995; Eiter and Fink 2003; Lin 2002).

In what follows, we assume that the reader is familiar with the basic concepts of complexity theory (cf., e.g., Papadimitriou (1994) for a comprehensive treatise on this subject). For convenience, we briefly recapitulate the definitions and some elementary properties of the complexity classes considered in our analysis. As usual, for any complexity class $C$, by co-$C$ we understand the class of all problems which are complementary to the problems in $C$.

Four complexity classes are relevant here, viz. NP, co-NP, $\Sigma_2^P$, and $\Pi_2^P$. In detail, the class NP consists of all decision problems which can be solved with a nondeterministic Turing machine working in polynomial time; $\Sigma_2^P$ is the class of all

[5] We note that Lin (2002) discusses also the case of disjunctive logic programs with variables; here, however, we do not consider theories or programs with variables.

problems solvable with a nondeterministic Turing machine working in polynomial time having access to an oracle for problems in NP; and $\Pi_2^P = \text{co-}\Sigma_2^P$.

Observe that the above classes are part of the *polynomial hierarchy* (Stockmeyer 1976), which is given by the following sequence of objects: The initial elements are

$$\Delta_0^P = \Sigma_0^P = \Pi_0^P = \mathrm{P},$$

and, for $i > 0$,

$$\Delta_i^P = \mathrm{P}^{\Sigma_{i-1}^P},\ \Sigma_i^P = \mathrm{NP}^{\Sigma_{i-1}^P}, \text{ and } \Pi_i^P = \text{co-NP}^{\Sigma_{i-1}^P}.$$

Here, P is the class of all problems solvable with a deterministic Turing machine working in polynomial time, and, for complexity classes $C$ and $A$, the notation $C^A$ stands for the *relativised version* of $C$, i.e., consisting of all problems which can be decided by Turing machines of the same sort and time bound as in $C$, only that the machines have access to an oracle for problems in $A$. It holds that $\Sigma_1^P = \mathrm{NP}$, $\Sigma_2^P = \mathrm{NP}^{\mathrm{NP}}$, and $\Pi_2^P = \text{co-NP}^{\mathrm{NP}}$. A problem is said to be at the *n-th level* of the polynomial hierarchy iff it is in $\Delta_{n+1}^P$ and either $\Sigma_n^P$-hard or $\Pi_n^P$-hard.

The next proposition describes the close relation between the complexity classes $\Sigma_n^P$ and $\Pi_n^P$, for $n \geq 1$, and QBFs having $n-1$ quantifier alternations. Preparatorily, we introduce some further notation.

Let $\Phi$ be a closed prenex QBF of form $\mathsf{Q}_1 V_1 \mathsf{Q}_2 V_2 \ldots \mathsf{Q}_n V_n\, \phi$, where $\phi$ is a propositional formula, $\mathsf{Q}_i \in \{\exists, \forall\}$ such that $\mathsf{Q}_i \neq \mathsf{Q}_{i+1}$ for $1 \leq i \leq n-1$, and $V_1, \ldots, V_n$ are pairwise disjoint sets of propositional variables. We call $\Phi$ an $(n, \exists)$-*QBF* if $\mathsf{Q}_1 = \exists$, and an $(n, \forall)$-*QBF* if $\mathsf{Q}_1 = \forall$. Furthermore, we refer to $\phi$ as the *matrix* of $\Phi$.

*Proposition 9* (*Wrathall 1976*)
For every $n \geq 0$, the following properties hold:

1. Deciding whether a given $(n, \exists)$-QBF $\Phi$ is valid is $\Sigma_n^P$-complete. The problem remains $\Sigma_n^P$-hard even if the matrix of $\Phi$ is in conjunctive normal form and $n$ is odd, or in disjunctive normal form and $n$ is even.
2. Deciding whether a given $(n, \forall)$-QBF $\Phi$ is valid is $\Pi_n^P$-complete. The problem remains $\Pi_n^P$-hard even if the matrix of $\Phi$ is in disjunctive normal form and $n$ is odd, or in conjunctive normal form and $n$ is even.

As special cases of these results, we have that the satisfiability problem of classical propositional logic is NP-complete, and that the validity problem of classical propositional logic is co-NP-complete. Indeed, for a propositional formula $\varphi$ with $V = var(\varphi)$, $\varphi$ is satisfiable in classical propositional logic iff $\exists V \varphi$ is valid in quantified propositional logic, and $\varphi$ is valid in classical propositional logic iff $\forall V \varphi$ is valid in quantified propositional logic.

In view of Proposition 9, we can estimate upper complexity bounds for our considered decision problems simply by inspecting the quantifier order of the respective encodings. This can be argued as follows. First of all, by applying the transformation rules described in Proposition 4, each of our encodings can be transformed in polynomial time into a closed QBF in prenex form. Then, by invoking Proposition 9 and observing that completeness of a decision problem $D$ for a complexity class $C$ implies membership of $D$ in $C$, the quantifier order of the resultant QBFs determines in which class of the polynomial hierarchy the corresponding decision problem lies.

Before dealing with the decision problems associated with equilibrium logic and nested logic programs, we analyse the complexity of the logic of here-and-there.

*Theorem 8*

1. Deciding whether a given propositional formula is HT-satisfiable is NP-complete.
2. Deciding whether a given propositional formula is HT-valid is co-NP-complete.

*Proof*

Membership for each of the two tasks in the respective complexity classes follows from the polynomial-time constructible reduction $\mathcal{T}_{HT}[\cdot]$ into classical propositional logic. Concerning NP-hardness of the HT-satisfiability problem, we show that the NP-hard problem of checking whether a given formula in conjunctive normal form is satisfiable in classical propositional logic can be reduced to it in polynomial time. This can be done as follows.

Let $\varphi$ be a formula in conjunctive normal form, let $V = \{v_1, \ldots, v_n\}$ be the set of atoms occurring in $\varphi$, and let $W = \{w_1, \ldots, w_n\}$ be a set of new atoms. Furthermore, let $\varphi^+$ result from $\varphi$ by replacing each negative literal $\neg v_i$ in $\varphi$ by $w_i$, and define

$$\mathcal{S}[\varphi] = \bigwedge_{i=1}^{n} \big((v_i \vee w_i) \wedge (\neg v_i \vee \neg w_i)\big) \wedge \varphi^+.$$

Obviously, $\mathcal{S}[\varphi]$ is constructible from $\varphi$ in polynomial time (actually, in linear time). Furthermore, it holds that $\varphi$ is satisfiable in classical propositional logic iff $\mathcal{S}[\varphi]$ is HT-satisfiable.

To see this, assume first that $I \subseteq V$ is a model of $\varphi$. Define $J_I = I \cup \{w_i \mid v_i \in V \setminus I\}$. Then, since $\varphi^+$ is an expression without negations, by Part 3 of Proposition 1, we get that $\langle J_I, J_I \rangle$ is an HT-model of $\varphi^+$. But $\langle J_I, J_I \rangle$ is also an HT-model of $\bigwedge_{i=1}^{n} \big((v_i \vee w_i) \wedge (\neg v_i \vee \neg w_i)\big)$, so it follows that $\mathcal{S}[\varphi]$ is HT-satisfiable.

Conversely, consider $I, J \subseteq V \cup W$ such that $\mathcal{I} = \langle I, J \rangle$ is an HT-model of $\mathcal{S}[\varphi]$. So, $\nu_{\mathcal{I}}(H, \mathcal{S}[\varphi]) = 1$, and therefore $\nu_{\mathcal{I}}(H, \bigwedge_{i=1}^{n}((v_i \vee w_i) \wedge (\neg v_i \vee \neg w_i))) = 1$ and $\nu_{\mathcal{I}}(H, \varphi^+) = 1$. Clearly, the former condition entails that $v_i \in I$ iff $w_i \notin I$, for $1 \leq i \leq n$, and by again invoking Part 3 of Proposition 1, from $\nu_{\mathcal{I}}(H, \varphi^+) = 1$ we get that $\nu_I(\varphi) = 1$. Hence, $I$ is a model of $\varphi$.

It remains to show that checking HT-validity is co-NP-hard. For this, the co-NP-hard problem of deciding whether a given formula in disjunctive normal form is valid in classical propositional logic can be reduced in polynomial time to the problem of deciding HT-validity.

Let $\varphi$ and $\varphi^+$ be as above, and define

$$\mathcal{V}[\varphi] = \Big(\bigwedge_{i=1}^{n} \big((v_i \vee w_i) \wedge (\neg v_i \vee \neg w_i)\big)\Big) \supset \varphi^+.$$

Then, analogously to Part 1, it can be shown that $\varphi$ is valid in classical propositional logic iff $\mathcal{V}[\varphi]$ is HT-valid. Moreover, $\mathcal{V}[\varphi]$ is constructible from $\varphi$ in linear time. $\square$

We remark that, since the logic of here-and-there is equivalent to the three-valued logic of Heyting (1930) and Gödel (1932), the above results are also implicit as part of the general characterisation of the complexity of many-valued logics due to Mundici (1987).

We now turn to the complexity of the main reasoning tasks associated with nested logic programs.

*Theorem 9*
Both the consistency problem and brave reasoning for nested logic programs are $\Sigma_2^P$-complete, and skeptical reasoning for nested logic programs is $\Pi_2^P$-complete.

*Proof*
The membership conditions for each of the three decision problems are obtained by virtue of Proposition 9 and the encodings given in Corollary 2. In detail, both the encoding for the consistency problem as well as the encoding for brave reasoning can be transformed in polynomial time into a $(2, \exists)$-QBF. Hence, these tasks are in $\Sigma_2^P$. Analogously, the encoding for skeptical reasoning can be transformed in polynomial time into a $(2, \forall)$-QBF, and so skeptical reasoning lies in $\Pi_2^P$.

The matching lower bounds are a direct consequence of the complexity of the corresponding reasoning tasks for disjunctive logic programs. To wit, as shown by Eiter and Gottlob (1995), both the consistency problem and brave reasoning for disjunctive logic programs are $\Sigma_2^P$-complete, and skeptical reasoning for disjunctive logic programs is $\Pi_2^P$-complete. $\square$

For equilibrium logic, the complexity behaves analogously:

*Theorem 10*
Both the consistency problem and brave reasoning for equilibrium logic are $\Sigma_2^P$-complete, and skeptical reasoning for equilibrium logic is $\Pi_2^P$-complete.

*Proof*
The membership conditions are argued in the same way as in the case of logic programs, by using the encodings of Corollary 1.

As for the hardness parts, these follow from the respective hardness results from Theorem 9, by observing that Proposition 3 implies that, for any logic program $\Pi$, it holds that $\Pi$ has a stable model iff $\hat{\Pi}$ has an equilibrium model, and $\Pi \vdash_\varepsilon \varphi$ iff $\hat{\Pi} \vdash_\varepsilon \varphi$, for $\varepsilon \in \{b, s\}$ and any formula $\varphi$. $\square$

We note that $\Sigma_2^P$-hardness of the consistency problem for equilibrium logic was also previously shown by Pearce et al. (2000a).

We now turn to analysing the complexity of checking equivalence between theories or logic programs. Again, we start with the case of logic programs.

*Theorem 11*

1. Deciding ordinary equivalence between two given programs is $\Pi_2^P$-complete, and likewise for deciding uniform equivalence.
2. Deciding strong equivalence between two given programs is co-NP-complete.

*Proof*

Again, the membership conditions for each of the three tasks follow from their respective encodings from Corollary 3. That is, both the encoding for checking ordinary equivalence as well as the encoding for checking uniform equivalence can be transformed in polynomial time into a $(2, \forall)$-QBF, and the test for strong equivalence is encoded in terms of checking the validity of a formula of classical propositional logic (or, equivalently, checking strong equivalence can be transformed in polynomial time into a $(1, \forall)$-QBF).

$\Pi_2^P$-hardness for checking ordinary equivalence between logic programs holds because of the well-known fact that this is already the case for checking equivalence between disjunctive logic programs.[6] Likewise, checking uniform equivalence between disjunctive logic programs is known to be $\Pi_2^P$-complete (Eiter and Fink 2003), so $\Pi_2^P$-hardness for the corresponding problem for nested problems follows *a fortiori*.

It remains to show that checking strong equivalence is co-NP-hard. This can be seen as follows: Consider a formula $\varphi$ and the reduction $\mathcal{S}[\varphi]$ from the proof of Theorem 8. Then, one can show that $\varphi$ is unsatisfiable iff the logic program $\{\mathcal{S}[\varphi] \leftarrow \top\}$ is strongly equivalent to the program $\{\bot \leftarrow \top\}$. Since the problem of checking whether a given formula is unsatisfiable in classical propositional logic is co-NP-complete, the co-NP-hardness of checking strong equivalence between programs follows. $\square$

For disjunctive logic programs, co-NP-completeness for testing strong equivalence was independently derived by Lin (2002), and co-NP-membership in the case of nested logic programs is also shown by Turner (2003).

Finally, as a strengthening of Theorem 11, we have the following results for checking equivalence in equilibrium logic.

*Theorem 12*

1. Deciding whether two given theories of equilibrium logic are ordinarily equivalent is $\Pi_2^P$-complete, and likewise for deciding whether they are uniformly equivalent.
2. Deciding whether two given theories of equilibrium logic are strongly equivalent is co-NP-complete.

The complexity results of this section tell us that our translations adequately match, in some sense, the inherent complexity of the respective problems. In more formal terms, following Besnard et al. (2005), let us call a translation $\mathcal{T}_D(\cdot)$, mapping

[6] This result follows easily by inspecting the $\Sigma_2^P$-hardness proof for the consistency problem for disjunctive logic programs given by Eiter and Gottlob (1995). An explicit proof is given, e.g., by Oikarinen and Janhunen (2004).

instances of a decision problem $D$ into QBFs, *adequate* if the following criteria are met:

1. For each instance $I$ of $D$, $\mathcal{T}_D(I)$ is valid iff $I$ is a yes-instance of $D$ (i.e., $\mathcal{T}_D(\cdot)$ is *faithful*);
2. for each instance $I$ of $D$, $\mathcal{T}_D(I)$ is computable in polynomial time; and
3. determining the truth values of the QBFs resulting from $\mathcal{T}_D(\cdot)$ is not computationally harder than the problem $D$ itself.

It is a straightforward matter to check that all our encodings are indeed adequate in the above sense. More specifically, the following result holds:

*Theorem 13*
All of the encodings described in Corollaries 1 and 2, Theorem 6, Corollary 3, and Theorem 7 are adequate.

## 5 Adding Strong Negation

Strong negation was introduced into logic by Nelson (1949) as a syntactic counterpart to the idea of *constructible falsity.* Later, Vorob'ev (1952) showed how to axiomatise strong negation and provided a reduction technique by which strong negations could be eliminated in favour of additional predicates—a technique later used by Gurevich (1977) to obtain completeness theorems. More recently, strong negation was introduced into logic programming by Pearce and Wagner (1991) and Gelfond and Lifschitz (1991), though the latter called their operator *classical* negation. In their answer-set semantics for extended logic programs, Gelfond and Lifschitz applied the same technique as Vorob'ev to eliminate strong negation and reduce properties of answer sets for extended programs to those of stable models for programs with only one negation.

Equilibrium logic is also defined for theories with two kinds of negation. As before, the basis is the nonclassical logic of here-and-there now augmented by adding a new negation operator, $\sim$, together with the Vorob'ev axioms, see Pearce (1997) and Pearce et al. (2000a). This yields a logic called *here-and-there with strong negation*, denoted by $N_5$. Interpretations are defined to be here-and-there models as before except that now in each world both atoms and strongly negated atoms may be verified. An atom possibly prefixed by one occurrence of strong negation is called an *objective literal* and a set of objective literals is called *consistent* if it does not contain both $v$ and $\sim v$ for some atom $v$. Hence, an $N_5$*-interpretation*, $\mathcal{I}$, is an ordered pair $\langle I_H, I_T\rangle$ of consistent sets of objective literals such that as before $I_H \subseteq I_T$. Then, the truth value, $\nu_{\mathcal{I}}(w, \varphi)$, of a formula $\varphi$ in a world $w \in \{H, T\}$ in an $N_5$-interpretation $\mathcal{I} = \langle I_H, I_T\rangle$ is recursively defined as previously, but now with the addition of new clauses for evaluating strong negation. As well, $N_5$-models are defined as in HT. The concept of equilibrium model, then, is also defined analogously. To wit, an equilibrium model of a set $T$ of formulas is an $N_5$-model $\langle I, J\rangle$ of $T$ such that (i) $I = J$ and (ii) for every proper subset $J$ of $I$, $\langle J, I\rangle$ is not an $N_5$-model of $T$.

Enriched in this fashion, equilibrium logic continues to be a conservative extension of the stable model or answer-set semantics for logic programs. If $\Pi$ is a (nested) logic program possibly containing strong negation, as before we denote by $\hat{\Pi}$ the set of formulas corresponding to the rules of $\Pi$. Then, we have the following relation, extending Proposition 3 and shown by Lifschitz et al. (2001):

*Proposition 10*
For any program $\Pi$, $I$ is an answer set of $\Pi$ iff $\langle I, I\rangle$ is an equilibrium model of $\hat{\Pi}$.

Our previous transformations can easily be extended to cover strong negation by deploying the reduction technique of Vorob'ev and Gurevich. The method involves two steps and is carried out as follows. First, let us say that a formula of $N_5$ is in *reduced form* if any occurrence of strong negation appears directly in front of an atom. The first step is then, given any formula $\varphi$, to convert $\varphi$ into an equivalent formula $\varphi^*$ in reduced form. The translation '$\cdot^*$' is recursively defined as follows:

1. $v^* = v$ and $({\sim}v)^* = {\sim}v$, if $v$ is an atom;
2. $(\neg\varphi)^* = \neg\varphi^*$;
3. $(\varphi \circ \psi)^* = \varphi^* \circ \psi^*$, for $\circ \in \{\land, \lor, \supset\}$;
4. ${\sim}(\varphi \supset \psi)^* = \varphi^* \land {\sim}\psi^*$;
5. ${\sim}(\varphi \land \psi)^* = {\sim}\varphi^* \lor {\sim}\psi^*$;
6. ${\sim}(\varphi \lor \psi)^* = {\sim}\varphi^* \land {\sim}\psi^*$; and
7. $({\sim}\neg\varphi)^* = ({\sim}{\sim}\varphi)^* = \varphi^*$.

It is easy to establish that for any formula $\varphi$, $\varphi^*$ is in reduced form and moreover $\varphi \equiv \varphi^*$ is provable in $N_5$. Furthermore, it is quite obvious that $\varphi^*$ can be obtained from $\varphi$ in linear time.

In the second step, we extend the language by adding, for each atom $v$, a new atom $\overline{v}$ not in the language. Let $\varphi$ be a formula. Then, set $c_\varphi = \bigwedge_{v \in var(\varphi)} \neg(v \land \overline{v})$. Furthermore, for any $N_5$-interpretation $\mathcal{I} = \langle I_H, I_T\rangle$, let $\overline{\mathcal{I}}$ be the HT-interpretation obtained from $\mathcal{I}$ by replacing each occurrence of a strongly negated literal ${\sim}v$ by $\overline{v}$. Finally, for any formula $\varphi$ of $N_5$, let $r(\varphi)$ be the formula that results from replacing each occurrence of ${\sim}v$ in $\varphi^*$ by $\overline{v}$. Clearly, $r(\varphi)$ is a formula of HT.

Then, we have:

*Lemma 5*
For any formula $\varphi$ of $N_5$ with $V = var(\varphi)$, any $N_5$-interpretation $\mathcal{I} = \langle I_H, I_T\rangle$, and any world $w \in \{H, T\}$, $\varphi$ is true in $w$ under $\mathcal{I}$ iff $c_\varphi \land r(\varphi)$ is true in $w$ under $\overline{\mathcal{I}}$.

*Proof*
By the validity of $(\varphi \equiv \varphi^*)$ in $N_5$, we need only show that $\nu_{\mathcal{I}}(w, \varphi^*) = 1$ iff $\nu_{\overline{\mathcal{I}}}(w, c_\varphi \land r(\varphi)) = 1$. This follows by an easy induction on $\varphi^*$. The cases for objective literals follow from the definition of $\overline{\mathcal{I}}$. For the other connectives, note that, since $\varphi^*$ is in reduced form, we only need to check the satisfaction conditions for $\land$, $\lor$, $\supset$, and $\neg$. But these are the same for $\mathcal{I}$ as for $\overline{\mathcal{I}}$. Lastly, the fact that $\mathcal{I}$ is a (consistent) interpretation guarantees that $c_\varphi$ is true in $w$ under $\overline{\mathcal{I}}$. $\square$

Let $\mathcal{I}$ be an HT-interpretation whose atoms belong to $V \cup \overline{V}$, where $\overline{V} = \{\overline{v} \mid v \in V\}$. We define $\mathcal{I}^\dagger$ to be the $N_5$-interpretation obtained by uniformly replacing each occurrence of $\overline{v}$ for an atom $v$ by $\sim v$. By the definition of equilibrium model and the above lemma, we obtain:

*Theorem 14*
For any formula $\varphi$ of $N_5$, the following conditions hold:

1. If an $N_5$-interpretation $\mathcal{I}$ is an equilibrium model of $\varphi$, then $\overline{\mathcal{I}}$ is an equilibrium model of $c_\varphi \wedge r(\varphi)$.
2. If an HT-interpretation $\mathcal{I}$ is an equilibrium model of $c_\varphi \wedge r(\varphi)$, then $\mathcal{I}^\dagger$ is an equilibrium model of $\varphi$.

In this manner, we can extend the previous reduction technique for equilibrium logic to the enriched system with strong negation. We need only convert each formula $\varphi$ to reduced form and add the conjunction $c_\varphi = \bigwedge_{v \in var(\varphi)} \neg(v \wedge \overline{v})$. The same applies to logic programs with strong negation. However, in the latter case, under the answer-set semantics of Lifschitz et al. (1999), it is assumed that strong negations are placed directly before atoms, hence the corresponding formulas are already in reduced form. We can therefore dispense with step one. The procedures $\mathcal{T}_E[\hat{\Pi}]$ and the optimised variant $\mathcal{T}_S[\Pi]$, for each program $\Pi$, are extended to the notion of answer sets again by adding the conjunction $\bigwedge_{v \in var(\Pi)} \neg(v \wedge \overline{v})$.

An additional frequently used notion is that of a *complete* answer set. An answer set $I$ of a logic program $\Pi$ is complete iff, for each atom $v$ in $\Pi$, either $v$ or $\sim v$ is contained in $I$. To deal with complete answer sets, it suffices to add a second conjunction, $\bigwedge_{v \in var(\Pi)}(v \vee \overline{v})$. Extensions to encode brave and skeptical reasoning for logic programs containing strong negation, as well as encoding consequence relations defined in terms of complete answer sets, can be defined analogously.

## 6 Related Work

Capturing one formalism in terms of another is a natural issue for theoretical discourse, and accordingly various translatability results appeared in the logic-programming literature. We first discuss investigations dealing with translations to classical logic—these are clearly closest in spirit to our work. Afterwards, we consider translations from one syntactic subclass of programs to another. Finally, we give some pointers concerning work on equivalence checking.

In face of the inherent complexity of equilibrium logic and nested logic programs, and assuming the usual proviso that the polynomial hierarchy does not collapse, translations of reasoning tasks associated with any of these languages into classical logic usually fall in one of the following two categories:

- either one is interested in *efficient* translations, in which case these can be realised only for subclasses of programs or theories matching the complexity of classical logic,
- or else one deals with translations for *arbitrary* programs or theories, for which an exponential blow-up of the translations must be faced in the worst case.

Concerning the first kind of results, early efforts in that direction involve the work of Clark (1978), defining the *completion* of a program, and of Fages (1994). Ben-Eliyahu and Dechter (1994) defined a syntactic subclass of disjunctive logic programs, viz. *head-cycle free programs*, along with a translation of these programs into theories of classical logic constructible in polynomial time. This translation was recently optimised by Janhunen (2004) for *normal logic programs*,[7] and Linke et al. (2004) provide a generalisation of the notion of head-cycle freeness to nested logic programs, together with an extension of the translation of Ben-Eliyahu and Dechter (1994). As well, Linke et al. (2004) give a generalisation of Fages' theorem to so-called *tight nested logic programs*, encompassing earlier results due to Erdem and Lifschitz (2001; 2003).

A prominent technique for realising translations of programs without any additional syntactic proviso into classical logic, but instead accepting an exponential blow-up in the worst case, are based on adding to Clark's completion so-called *loop formulas*, guaranteeing equivalence between the stable models of the given program and the classical models of the resultant theory. This idea was first pursued by Lin and Zhao (2002) for normal programs and subsequently extended by Lee and Lifschitz (2003) for disjunctive programs with nested formulas in rule bodies. Further extensions of this approach were put forth by Lee (2005) and Gebser et al. (2006). Implementations of the loop-formula approach are the `ASSat` system (Lin and Zhao 2002) and the `CMODELS` system (Lierler 2005). An interesting relation between our QBF encodings and loop formulas has recently been analysed by Ferraris et al. (2006).

Given that the main reasoning tasks associated with equilibrium logic and nested logic programs have the same complexity as the corresponding problems for disjunctive logic programs entails that the former can be efficiently reduced to the latter. A polynomial translation from nested logic programs to disjunctive ones has been realised by Pearce et al. (2002), based on a labelling technique and exploiting a translation from *generalised disjunctive logic programs* to disjunctive programs due to Janhunen (2001).[8] Although already Lifschitz et al. (1999) provide a translation from nested programs to disjunctive ones, their translation is exponential in the worst case. Furthermore, the translation by Pearce et al. (2002) satisfies *strong faithfulness*, expressing that, for every program $\Pi_1$ and $\Pi_2$, there is a one-to-one correspondence between the answer sets of $\Pi_1 \cup \Pi_2$ and sets of form $I \cap var(\Pi_1 \cup \Pi_2)$, where $I$ is an answer set of $\rho(\Pi_1) \cup \Pi_2$, with $\rho(\Pi_1)$ being translation of $\Pi_1$. Note that strong faithfulness is a refined version of strong equivalence, taking differing alphabets into account, as the translation $\rho$ introduces new atoms.

As for translating equilibrium logic to disjunctive logic programs, this was put forth by Cabalar et al. (2005), making use of a technique due to Osorio et al. (2005). In subsequent work, Cabalar and Ferraris (2006) show that each theory can also be

[7] Normal logic programs are that subclass of disjunctive programs where rule heads are atoms only.

[8] In a generalised disjunctive logic program, rule heads are disjunctions of literals and rule bodies are conjunctions of literals.

rewritten into a *strongly* equivalent nested logic program, but with an exponential blow-up in general.

Attention was also given in the literature to translations of disjunctive logic programs to normal programs. To wit, Ben-Eliyahu and Dechter (1994) provide a method to transform head-cycle free disjunctive programs equivalently into normal programs by shifting head atoms into the body.[9] For instance, a rule of form $p \vee q \leftarrow r$ is replaced by this method by the two rules $p \leftarrow r \wedge \neg q$ and $q \leftarrow r \wedge \neg p$. This shifting technique was later extended by Linke et al. (2004) to head-cycle free nested logic programs, thus providing a translation of the latter to normal programs. It is worth mentioning that this technique partly relies on a translation due to You et al. (2003) and extends ideas by Inoue and Sakama (1998). A general investigation under which conditions disjunctions can be eliminated from disjunctive logic programs was carried out by Eiter et al. (2004a). This was done for ordinary, uniform, and strong equivalence, and makes use of the above shifting technique.

We now turn to work regarding equivalence checking. Besides the notions analysed in this paper, more refined ones recently appeared in the literature. On the one hand, Woltran (2004) defines *relativised* notions of uniform and strong equivalence for disjunctive logic programs, in which the alphabet of the rules added for the comparison is taken into account. On the other hand, this investigation was subsequently extended by Eiter, Tompits, and Woltran (2005), defining a general framework for specifying program correspondence, and providing model-theoretic characterisations as well as complexity results. This framework allows for the specification of parameterised equivalence notions, taking, e.g., projected answer sets into account where auxiliary atoms are ignored for program comparison. We note that the latter feature is important towards realising modular programming. Concerning the complexity of relativised uniform and strong equivalence, this was thoroughly analysed by Eiter, Fink, and Woltran (2005). Modularity aspects for program comparison under ordinary equivalence in the presence of auxiliary atoms are also analysed by Oikarinen and Janhunen (2006). Inoue and Sakama (2004) define equivalence under program *updates*, and generalised characterisations of strong and uniform equivalence for language extensions like cardinality constraints and preferences was put forth by Liu and Truszczynski (2005) and Faber and Konczak (2005), respectively. Finally, the notion of *synonymous theories* in the context of equilibrium logic was introduced by Pearce and Valverde (2004a).

Concerning work on implementational aspects, let us first remind that our axiomatisations straightforwardly enable the development of provers for equilibrium logic using extant QBF solvers as back-end inference engines. In any case, a concrete implementation for equilibrium logic was developed by Valverde (2004), relying on a tableau calculus due to Pearce et al. (2000b). As well, an implementation for nested logic programs, based on the reduction to disjunctive logic programs put forth by (Pearce et al. 2002) and using the underlying solver `DLV` (Leone et al. 2006), is discussed by Sarsakov et al. (2004). A reduction approach to disjunctive

[9] This shifting method is also discussed by Dix et al. (1996) and Gelfond et al. (1991).

logic programs is also realised by several systems for equivalence checking. To begin with, the system `dlpeq` implements a tester for ordinary and strong equivalence. It was first developed for normal logic programs (Janhunen and Oikarinen 2002) and afterwards extended to disjunctive programs (Oikarinen and Janhunen 2004). A similar idea to decide strong equivalence using the language of logic programs itself was independently suggested by Turner (2003) and some aspects are carried out by Eiter et al. (2004b) as well. Another implementational approach is pursued by the system `SELP` (Chen et al. 2005) for checking strong equivalence, using the reductions to classical propositional logic described by Lin (2002) (see also our discussion in Section 3.4 about these translations), and also by the work of Eiter, Faber, and Traxler (2005). Finally, the family of equivalence notions from the framework of Eiter, Tompits, and Woltran (2005) are implemented by the system `cc`$\top$ (Oetsch et al. 2006a; Oetsch et al. 2006b), using reductions to quantified propositional logic put forth by Tompits and Woltran (2005), similar to the reduction approach followed in the present paper. We note that testing strong equivalence is one of the different notions handled by `cc`$\top$, and the reduction used for computing this task is equivalent to the one used by `SELP`.

## 7 Concluding Remarks

In this paper, we discussed how different reasoning problems in the context of equilibrium logic and nested logic programs can be expressed in a uniform framework by means of quantified propositional logic. We have started by introducing basic formulas that are used as building blocks for modeling differing reasoning tasks. Our results thus provide new axiomatisations for the formalisms under consideration.

The overall approach has several benefits. First of all, it allows us to compare different problems in a single formal language. This is of particular interest for the three different notions of equivalence considered here, viz. ordinary, uniform, and strong equivalence. Moreover, the axiomatisations imply upper complexity bounds in a direct manner for the problems under consideration, which we all strengthened to completeness results, thus providing strict complexity bounds. Lastly, the axiomatisations provide executable specifications that can be fed into extant QBF solvers, yielding prototypical implementations for the encoded problems. In view of the considerable sophistication offered by current QBF solvers, one obtains a viable approach for rapid prototyping.

Concerning future work, we mention two issues which we believe worthwhile to be pursued. First and foremost, we plan to lift our characterisations to the non-ground case, where theories or programs are defined over a function-free first-order language. Some work in this area has already been done: Pearce and Valverde (2004b) introduce a first-order variant of the logic HT and equilibrium logic, Eiter, Fink, Tompits, and Woltran (2005) introduce non-ground versions of strong and uniform equivalance, along with model-theoretic characterisations for them and analysing decidability and complexity issues, and a forthcoming paper by Ferraris et al. (2007) discusses an axiomatisation of the stable semantics of logic programs in terms of second-order logic. The last work is particularly noteworthy, as liftings of our encodings to the case with variables yield (general) formulas of second-order logic, yet the semantics used by Ferraris et al. (2007) differs from the standard one for non-ground programs as no unique-names assumption is stipulated. An interesting further research issue would be to extend the framework of Eiter, Tompits, and Woltran (2005) not only to the non-ground case but also to full equilibrium logic, and providing characterisations in the spirit of the present paper.

*Acknowledgments.*

This work was 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.

## Appendix A  Proof of Lemma 2

The proof proceeds on the number of connectives $\wedge$, $\vee$, and $\supset$ in $\varphi$ which are not in the scope of a negation. Denote this number by $lc^+(\varphi)$. First, observe that, for $I_H, I_T \subseteq V$, the pair $\langle I_H, I_T\rangle$ is an HT-interpretation (i.e., $I_H \subseteq I_T$) iff $I_H \cup I'_T$ is a model of $V \leq V'$.

*Induction Base.* Assume $lc^+(\varphi) = 0$. Then, $\varphi$ is either an atom, one of $\top$ or $\bot$, or of form $\neg\psi$, for some formula $\psi$. If $\varphi$ is one of $\top$ or $\bot$, then the statement holds trivially.

Consider now the case that $\varphi = p$, for some atom $p$. Then, $V = \{p\}$ and $\tau[p] = p$. Hence,

$$\mathcal{T}_{HT}[\varphi] = (p \supset p') \wedge p.$$

Assume that $\mathcal{I} = \langle I_H, I_T\rangle$ is an HT-model of $\varphi$, i.e., we have that $I_H \subseteq I_T \subseteq V$ and $p \in I_H$. From this, we immediately get that $I_H = I_T = V = \{p\}$, and so $I_H \cup I'_T = \{p, p'\}$, which is clearly a model of $\mathcal{T}_{HT}[\varphi]$. Conversely, if $I_H \cup I'_T \subseteq V \cup V'$ is a model of $\mathcal{T}_{HT}[\varphi] = (p \supset p') \wedge p$, then clearly $I_H = I_T = \{p\}$. Hence, $\mathcal{I} = \langle I_H, I_T\rangle$ is an HT-interpretation. Moreover, in view of $p \in I_H$, $\mathcal{I}$ is an HT-model of $\varphi$.

Now assume that $\varphi = \neg\psi$, for some formula $\psi$. By definition of $\tau[\cdot]$,

$$\mathcal{T}_{HT}[\varphi] = (V \leq V') \wedge \neg\psi'.$$

Suppose that $\mathcal{I} = \langle I_H, I_T\rangle$ is an HT-model of $\varphi = \neg\psi$. Then, $\nu_{\mathcal{I}}(w, \psi) = 0$, for each $w \in \{H, T\}$. By Proposition 1, $\nu_{\mathcal{I}}(T, \psi) = 0$ implies $\nu_{I_T}(\psi) = 0$. Hence, $\psi'$ is false under $I'_T$, and consequently $\neg\psi'$ is true under $I_H \cup I'_T$. Moreover, since $\mathcal{I}$ is an HT-interpretation, $V \leq V'$ is true under $I_H \cup I'_T$. Thus, $I_H \cup I'_T$ is a model of $\mathcal{T}_{HT}[\varphi]$.

Conversely, if $I_H \cup I'_T$ is a model of $\mathcal{T}_{HT}[\varphi] = (V \leq V') \wedge \neg\psi'$, then $\psi'$ is false under $I'_T$. From this, a simple renaming yields $\nu_{I_T}(\psi) = 0$. Now, since $I_H \cup I'_T$ is also a model of $(V \leq V')$, $\mathcal{I} = \langle I_H, I_T\rangle$ must be an HT-interpretation, and therefore Part 1 of Proposition 1 implies $\nu_{\mathcal{I}}(T, \psi) = 0$. But Part 2 of the same proposition yields $\nu_{\mathcal{I}}(H, \psi) = 0$. Consequently, $\nu_{\mathcal{I}}(H, \neg\psi) = 1$, and so $\mathcal{I}$ is an HT-model of $\varphi$.

*Induction Step.* Assume $lc^+(\varphi) > 0$, and let the statement hold for all formulas $\psi$

such that $lc^+(\psi) < lc^+(\varphi)$. We have to consider several cases, depending on the structure of $\varphi$.

*Case 1.* Assume that $\varphi = (\varphi_1 \wedge \varphi_2)$. Then,

$$\mathcal{T}_{HT}[\varphi] \quad = \quad (V \leq V') \wedge \tau[\varphi_1 \wedge \varphi_2],$$

with $\tau[\varphi_1 \wedge \varphi_2] = \tau[\varphi_1] \wedge \tau[\varphi_2]$. Since $V \leq V'$ is equivalent in classical logic to $(V_1 \leq V_1') \wedge (V_2 \leq V_2')$, where $V_i = var(\varphi_i)$, for $i = 1, 2$, it follows that $\mathcal{T}_{HT}[\varphi]$ is equivalent in classical logic to

$$\big((V_1 \leq V_1') \wedge \tau[\varphi_1]\big) \wedge \big((V_2 \leq V_2') \wedge \tau[\varphi_2]\big),$$

which in turn represents $\mathcal{T}_{HT}[\varphi_1] \wedge \mathcal{T}_{HT}[\varphi_2]$.
Suppose now that $\mathcal{I} = \langle I_H, I_T\rangle$ is an HT-model of $\varphi = (\varphi_1 \wedge \varphi_2)$. Hence, $\mathcal{I}$ is an HT-model of $\varphi_1$ and $\varphi_2$. Since $lc^+(\varphi_i) < lc^+(\varphi)$, for $i = 1, 2$, by induction hypothesis we get that $I_H \cup I_T'$ is a model of both $\mathcal{T}_{HT}[\varphi_1]$ and $\mathcal{T}_{HT}[\varphi_2]$, and thus also of $\mathcal{T}_{HT}[\varphi_1] \wedge \mathcal{T}_{HT}[\varphi_2]$. It follows that $I_H \cup I_T'$ is a model of $\mathcal{T}_{HT}[\varphi]$. The proof of the converse direction proceeds analogously.

*Case 2.* Assume that $\varphi = (\varphi_1 \vee \varphi_2)$. Similar to the above, $\tau[\varphi] = \tau[\varphi_1] \vee \tau[\varphi_2]$, hence $\mathcal{T}_{HT}[\varphi]$ is given by

$$(V \leq V') \wedge (\tau[\varphi_1] \vee \tau[\varphi_2]). \tag{A1}$$

By taking $V_i = var(\varphi_i)$, for $i = 1, 2$, it follows that (A1) is classically equivalent to

$$(V \leq V') \wedge \Big(\big((V_1 \leq V_1') \wedge \tau[\varphi_1]\big) \vee \big((V_2 \leq V_2') \wedge \tau[\varphi_2]\big)\Big),$$

which represents $(V \leq V') \wedge (\mathcal{T}_{HT}[\varphi_1] \vee \mathcal{T}_{HT}[\varphi_2])$.
Suppose now that $\mathcal{I} = \langle I_H, I_T\rangle$ is an HT-model of $\varphi = (\varphi_1 \vee \varphi_2)$. We get that $\mathcal{I}$ is an HT-model of $\varphi_1$ or of $\varphi_2$. Without loss of generality, assume that $\mathcal{I}$ is an HT-model of $\varphi_1$. Since $lc^+(\varphi_1) < lc^+(\varphi)$, by induction hypothesis it follows that $I_H \cup I_T'$ is a model of $\mathcal{T}_{HT}[\varphi_1]$. Hence, $I_H \cup I_T'$ is also a model of $\mathcal{T}_{HT}[\varphi_1] \vee \mathcal{T}_{HT}[\varphi_2]$. Furthermore, since $\mathcal{I}$ is an HT-interpretation, $I_H \cup I_T'$ is a model of $V \leq V'$. Therefore, $I_H \cup I_T'$ is a model of $(V \leq V') \wedge (\mathcal{T}_{HT}[\varphi_1] \vee \mathcal{T}_{HT}[\varphi_2])$. Since the last formula is equivalent to $\mathcal{T}_{HT}[\varphi]$, we obtain that $I_H \cup I_T'$ is a model of $\mathcal{T}_{HT}[\varphi]$. The converse direction follows in essentially the same way.

*Case 3.* Assume that $\varphi = (\varphi_1 \supset \varphi_2)$. Then,

$$\mathcal{T}_{HT}[\varphi] = (V \leq V') \wedge (\tau[\varphi_1] \supset \tau[\varphi_2]) \wedge (\varphi_1' \supset \varphi_2').$$

It is easy to see that $\mathcal{T}_{HT}[\varphi]$ is equivalent in classical logic to

$$(V \leq V') \wedge \Big(\big((V_1 \leq V_1') \wedge \neg\tau[\varphi_1]\big) \vee \big((V_2 \leq V_2') \wedge \tau[\varphi_2]\big)\Big) \wedge (\varphi_1' \supset \varphi_2'), \tag{A2}$$

where $V_i = var(\varphi_i)$, for $i = 1, 2$.
Assume that $\mathcal{I} = \langle I_H, I_T\rangle$ is an HT-model of $\varphi$. This means that (i) $\nu_{\mathcal{I}}(H, \varphi_1) = 0$ or $\nu_{\mathcal{I}}(H, \varphi_2) = 1$, and (ii) $\nu_{\mathcal{I}}(T, \varphi_1) = 0$ or $\nu_{\mathcal{I}}(T, \varphi_2) = 1$. We show that each conjunct of (A2) is true under $I_H \cup I_T'$.
Clearly, $V \leq V'$ is true under $I_H \cup I_T'$ because $\mathcal{I}$ is an HT-interpretation. Moreover, Part 1 of Proposition 1 implies that (ii) is equivalent to the condition that

$\nu_{I_T}(\varphi_1) = 0$ or $\nu_{I_T}(\varphi_2) = 1$. From the latter we obtain by a simple renaming that $I_H \cup I'_T$ is a model of $(\varphi'_1 \supset \varphi'_2)$. It remains to show that $I_H \cup I'_T$ is a model of

$$\big((V_1 \leq V'_1) \wedge \neg\tau[\varphi_1]\big) \vee \big((V_2 \leq V'_2) \wedge \tau[\varphi_2]\big). \tag{A3}$$

From (i) we know that either $\nu_{\mathcal{I}}(H, \varphi_1) = 0$ or $\nu_{\mathcal{I}}(H, \varphi_2) = 1$, or both. Similar to arguments in the induction base, one can show that $\nu_{\mathcal{I}}(H, \varphi_1) = 0$ implies that $I_H \cup I'_T$ is a model of $(V_1 \leq V'_1) \wedge \neg\tau[\varphi_1]$. Now assume that $\nu_{\mathcal{I}}(H, \varphi_2) = 1$, i.e., $\mathcal{I}$ is an HT-model of $\varphi_2$. Since $lc^+(\varphi_2) < lc^+(\varphi)$, by induction hypothesis we get that $I_H \cup I'_T$ is a model of $(V_2 \leq V'_2) \wedge \tau[\varphi_2]$. Hence, in either case, we can conclude that $I_H \cup I'_T$ is a model of (A3). This shows that if $\mathcal{I}$ is an HT-model of $\varphi$, then $I_H \cup I'_T$ is a model of $\mathcal{T}_{HT}[\varphi]$. The proof of the converse direction follows in a similar fashion.

## References

Arieli, O. and Denecker, M. 2003. Reducing preferential paraconsistent reasoning to classical entailment. *Journal of Logic and Computation 13,* 4, 557–580.

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

Ben-Eliyahu, R. and Dechter, R. 1994. Propositional semantics for disjunctive logic programs. *Annals of Mathematics and Artificial Intelligence 12,* 53–87.

Besnard, P., Schaub, T., Tompits, H., and Woltran, S. 2005. Representing paraconsistent reasoning via quantified propositional logic. In *Inconsistency Tolerance*, L. Bertossi, A. Hunter, and T. Schaub, Eds. LNCS, vol. 3300. Springer, 84–118.

Cabalar, P. and Ferraris, P. 2006. Propositional theories are strongly equivalent to logic programs. Submitted draft.

Cabalar, P., Pearce, D., and Valverde, A. 2005. Reducing propositional theories in equilibrium logic to logic programs. In *Proceedings of the 12th Portuguese Conference on Artificial Intelligence* (*EPIA 2005*), C. Bento, A. Cardoso, and G. Dias, Eds. LNCS, vol. 3808. Springer, 4–17.

Chen, Y., Lin, F., and Li, L. 2005. SELP - A system for studying strong equivalence between logic programs. In *Proceedings of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2005*), C. Baral, G. Greco, N. Leone, and G. Terracina, Eds. LNCS, vol. 3662. Springer, 442–446.

Church, A. 1956. *Introduction to Mathematical Logic, Volume I.* Princeton University Press.

Clark, K. L. 1978. Negation as failure. In *Logic and Data Bases*, H. Gallaire and J. Minker, Eds. Plenum Press, 127–138.

de Jongh, D. and Hendriks, L. 2003. Characterizations of strongly equivalent logic programs in intermediate logics. *Theory and Practice of Logic Programming 3,* 3, 259–270.

Delgrande, J., Schaub, T., Tompits, H., and Woltran, S. 2004. On computing solutions to belief change scenarios. *Journal of Logic and Computation 14,* 6, 801–826.

Dix, J., Gottlob, G., and Marek, V. 1996. Reducing disjunctive to non-disjunctive semantics by shift-operations. *Fundamenta Informaticae XXVIII,* (1/2), 87–100.

Egly, U., Eiter, T., Tompits, H., and Woltran, S. 2000. Solving advanced reasoning tasks using quantified Boolean formulas. In *Proceedings of the 17th National Conference on Artificial Intelligence* (*AAAI 2000*). AAAI Press/MIT Press, 417–422.

Egly, U., Seidl, M., Tompits, H., Woltran, S., and Zolda, M. 2004. Comparing different prenexing strategies for quantified Boolean formulas. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing (SAT 2003). Selected Revised Papers*, E. Giunchiglia and A. Tacchella, Eds. LNCS, vol. 2919. 214–228.

Eiter, T., Faber, W., and Traxler, P. 2005. Testing strong equivalence of nonmonotonic datalog programs - implementation and examples. In *Proceedings of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2005)*, C. Baral, G. Greco, N. Leone, and G. Terracina, Eds. LNCS, vol. 3662. Springer, 437–441.

Eiter, T. and Fink, M. 2003. Uniform equivalence of logic programs under the stable model semantics. In *Proceedings 19th International Conference on Logic Programming (ICLP 2003)*, C. Palamidessi, Ed. LNCS, vol. 2916. Springer, 224–238.

Eiter, T., Fink, M., Tompits, H., and Woltran, S. 2004a. On eliminating disjunctions in stable logic programming. In *Proceedings of the 9th International Conference on Principles of Knowledge Representation and Reasoning (KR 2004)*. AAAI Press, 447–458.

Eiter, T., Fink, M., Tompits, H., and Woltran, S. 2004b. Simplifying logic programs under uniform and strong equivalence. In *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR 2003)*, V. Lifschitz and I. Niemelä, Eds. LNCS, vol. 2923. Springer, 87–99.

Eiter, T., Fink, M., Tompits, H., and Woltran, S. 2005. Strong and uniform equivalence in answer-set programming: Characterizations and complexity results for the non-ground case. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, M. Veloso and S. Kambhampati, Eds. AAAI Press / The MIT Press, 695–700.

Eiter, T., Fink, M., and Woltran, S. 2005. Semantical characterizations and complexity of equivalences in stable logic programming. Tech. Rep. INFSYS RR-1843-05-01, Institut für Informationssysteme, Technische Universität Wien, Austria. Accepted for publication in *ACM Transactions on Computational Logic.*

Eiter, T. and Gottlob, G. 1995. On the computational cost of disjunctive logic programming: Propositional case. *Annals of Mathematics and Artificial Intelligence 15,* 3–4, 289–323.

Eiter, T., Gottlob, G., and Mannila, H. 1997. Disjunctive Datalog. *ACM Transactions on Database Systems 22,* 3, 364–418.

Eiter, T., Klotz, V., Tompits, H., and Woltran, S. 2002. Modal nonmonotonic logics revisited: Efficient encodings for the basic reasoning tasks. In *Proceedings of the 11th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX 2002)*, U. Egly and C. Fermüller, Eds. LNCS, vol. 2381. Springer, 100–114.

Eiter, T., Tompits, H., and Woltran, S. 2005. On solution correspondences in answer set programming. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, L. Pack Kaelbling and A. Saffiotti, Eds. Professional Book Center, 97–102.

Erdem, E. and Lifschitz, V. 2001. Fages' theorem for programs with nested expressions. In *Proceedings of the 18th International Conference on Logic Programming (ICLP 2001)*, P. Codognet, Ed. LNCS, vol. 2237. Springer, 242–254.

Erdem, E. and Lifschitz, V. 2003. Tight logic programs. *Theory and Practice of Logic Programming 3,* 4–5, 499–518.

Faber, W. and Konczak, K. 2005. Strong equivalence for logic programs with preferences. In *Proceedings of the 19th International Joint Conference on Artificial Intelli-*

*gence* (*IJCAI 2005*), L. Pack Kaelbling and A. Saffiotti, Eds. Professional Book Center, 430–435.

Fages, F. 1994. Consistency of Clark's completion and existence of stable models. *Methods of Logic in Computer Science 1*, 51–60.

Ferraris, P. 2005. Answer sets for propositional theories. In *Proceedings of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2005*), C. Baral, G. Greco, N. Leone, and G. Terracina, Eds. LNCS, vol. 3662. Springer, 119–131.

Ferraris, P., Lee, J., and Lifschitz, V. 2006. A generalization of the Lin-Zhao theorem. *Annals of Mathematics and Artificial Intelligence*. To appear.

Ferraris, P., Lee, J., and Lifschitz, V. 2007. A new perspective on stable models. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (*IJCAI 2007*). To appear.

Garey, M. R. and Johnson, D. S. 1979. *Computers and Intractability.* W. H. Freeman.

Gebser, M., Lee, J., and Lierler, Y. 2006. Elementary sets for logic programs. In *Proceedings of the 21st National Conference on Artificial Intelligence* (*AAAI 2006*). AAAI Press.

Gelfond, M. and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing 9,* 3–4, 365–385.

Gelfond, M., Lifschitz, V., Przymusinska, H., and Truszczyński, M. 1991. Disjunctive defaults. In *Proceedings of the 2nd Conference on Principles of Knowledge Representation and Reasoning* (*KR '91*), J. Allen, R. Fikes, and B. Sandewall, Eds. Morgan Kaufmann, 230–237.

Gelfond, M., Przymusinska, H., and Przymusinski, T. C. 1989. On the relationship between circumscription and negation as failure. *Artificial Intelligence 38,* 1, 75–94.

Gelfond, M., Przymusinska, H., and Przymusinski, T. C. 1990. On the relationship between CWA, minimal model, and minimal herbrand model semantics. *International Journal of Intelligent Systems 5*, 549–564.

Gödel, K. 1932. Zum intuitionistischen Aussagenkalkül. *Anzeiger der Akademie der Wissenschaften in Wien*, 65–66.

Gurevich, Y. 1977. Intuitionistic logic with strong negation. *Studia Logica 36,* 1-2, 49–59.

Heyting, A. 1930. Die formalen Regeln der intuitionistischen Logik. *Sitzungsberichte der Preussischen Akademie der Wissenschaften*, 42–56. Reprint in *Logik-Texte: Kommentierte Auswahl zur Geschichte der Modernen Logik*, Akademie-Verlag, 1986.

Inoue, K. and Sakama, C. 1998. Negation as failure in the head. *Journal of Logic Programming 35,* 1, 39–78.

Inoue, K. and Sakama, C. 2004. Equivalence of logic programs under updates. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence* (*JELIA 2004*), J. J. Alferes and J. A. Leite, Eds. LNCS, vol. 3229. Springer, 174–186.

Janhunen, T. 2001. On the effect of default negation on the expressiveness of disjunctive rules. In *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2001*), T. Eiter, W. Faber, and M. Truszczynski, Eds. LNCS, vol. 2173. Springer, 93–106.

Janhunen, T. 2004. Representing normal programs with clauses. In *Proceedings of the 16th Eureopean Conference on Artificial Intelligence* (*ECAI 2004*), R. L. de Mántaras and L. Saitta, Eds. IOS Press, 358–362.

Janhunen, T. and Oikarinen, E. 2002. Testing the equivalence of logic programs under stable model semantics. In *Proceedings of the 8th European Conference on Logics in Artificial Intelligence* (*JELIA 2002*), S. Flesca, S. Greco, N. Leone, and G.Ianni, Eds. LNCS, vol. 2424. Springer, 493–504.

Kowalski, V. 1968. The calculus of the weak "law of excluded middle". *Mathematics of the USSR 8*, 648–658.

Le Berre, D., Narizzano, M., Simon, L., and Tacchella, A. 2005. The second QBF solvers comparative evaluation. In *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing* (*SAT 2004*)*, Revised Selected Papers*, H. Hoos and D. Mitchell, Eds. LNCS, vol. 3542. Springer, 376–392.

Lee, J. 2005. A model-theoretic counterpart of loop formulas. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence* (*IJCAI 2005*), L. Pack Kaelbling and A. Saffiotti, Eds. Professional Book Center, 503–508.

Lee, J. and Lifschitz, V. 2003. Loop formulas for disjunctive logic programs. In *Proceedings of the 19th International Conference on Logic Programming* (*ICLP 2003*), C. Palamidessi, Ed. LNCS, vol. 2916. Springer, 451–465.

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., and Scarcello, F. 2006. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic 7,* 3, 499–562.

Leśniewski, S. 1929. Grundzüge eines neuen System der Grundlagen der Mathematik. *Fundamenta Mathematica 14*, 1–81.

Lierler, Y. 2005. Disjunctive answer set programming via satisfiability. In *Proceedings of the 8th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2005*), C. Baral, G. Greco, N. Leone, and G. Terracina, Eds. LNCS, vol. 3662. Springer, 447–451.

Lifschitz, V. 1994. Circumscription. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, D. M. Gabbay, C. J. Hogger, and J. A. Robinson, Eds. Clarendon Press, 297–352.

Lifschitz, V., Pearce, D., and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Transactions on Computational Logic 2,* 4, 526–541.

Lifschitz, V., Tang, L., and Turner, H. 1999. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence 25,* 3-4, 369–389.

Lifschitz, V. and Turner, H. 1994. Splitting a logic program. In *Proceedings of the 11th International Conference on Logic Programming* (*ICLP '94*). MIT Press, 23–38.

Lin, F. 1991. A Study of Nonmonotonic Reasoning. Ph.D. thesis, Stanford University, California, USA.

Lin, F. 2002. Reducing strong equivalence of logic programs to entailment in classical propositional logic. In *Proceedings of the 8th International Conference on Principles of Knowledge Representation and Reasoning* (*KR 2002*), D. Fensel, F. Giunchiglia, D. McGuinness, and M.-A. Williams, Eds. Morgan Kaufmann, 170–176.

Lin, F. and Zhao, Y. 2002. ASSAT: Computing answer sets of a logic program by SAT solvers. In *Proceedings of the 18th National Conference on Artificial Intelligence* (*AAAI 2002*). AAAI Press/MIT Press, 112–117.

Linke, T., Tompits, H., and Woltran, S. 2004. On acyclic and head-cycle free nested logic programs. In *Proceedings of the 20th International Conference on Logic Programming* (*ICLP 2004*), B. Demoen and V. Lifschitz, Eds. LNCS, vol. 3132. Springer, 225–239.

Liu, L. and Truszczynski, M. 2005. Properties of programs with monotone and convex constraints. In *Proceedings of the 20th National Conference on Artificial Intelligence* (*AAAI 2005*), M. Veloso and S. Kambhampati, Eds. AAAI Press, 701–706.

Łukasiewicz, J. and Tarski, A. 1930. Untersuchungen über den Aussagenkalkül. *Comptes Rendus Séances Société des Sciences et Lettres Varsovie 23,* Cl. III, 30–50.

Maher, M. J. 1988. Equivalence of logic programs. In *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed. Morgan Kaufmann Pub., 627–658.

McCarthy, J. 1980. Circumscription - a form of nonmonotonic reasoning. *Artificial Intelligence 13*, 27–39.

Meyer, A. R. and Stockmeyer, L. J. 1973. Word problems requiring exponential time. In *ACM Symposium on Theory of Computing* (*STOC '73*). ACM Press, 1–9.

Mundici, D. 1987. Satisfiability in many-valued sentential logic is NP-complete. *Theoretical Computer Science 52,* 1-2, 145–153.

Nelson, D. 1949. Constructible falsity. *Journal of Symbolic Logic 14,* 2, 16–26.

Oetsch, J., Seidl, M., Tompits, H., and Woltran, S. 2006a. cc$\top$: A correspondence-checking tool for logic programs under the answer-set semantics. In *Proceedings of the 10th European Conference on Logics in Artificial Intelligence* (*JELIA 2006*), M. Fisher, W. van der Hoek, B. Konev, and A. Lisitsa, Eds. LNCS, vol. 4160. Springer, 502–505.

Oetsch, J., Seidl, M., Tompits, H., and Woltran, S. 2006b. cc$\top$: A tool for checking advanced correspondence problems in answer-set programming. In *Proceedings of the 15th International Conference on Computing* (*CIC 2006*), A. Gelbukh and S. S. Guerra, Eds. IEEE Computer Society Press, 3–10.

Oikarinen, E. and Janhunen, T. 2004. Verifying the equivalence of logic programs in the disjunctive case. In *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2003*), V. Lifschitz and I. Niemelä, Eds. LNCS, vol. 2923. Springer, 180–193.

Oikarinen, E. and Janhunen, T. 2006. Modular equivalence for normal logic programs. In *Proceedings of the 11th International Workshop on Nonmonotonic Reasoning* (*NMR 2006*), J. Dix and A. Hunter, Eds. University of Clausthal, Department of Informatics, Techical Report, IfI-06-04, 10–18.

Osorio, M., Navarro, J. A., and Arrazola, J. 2005. Safe beliefs for propositional theories. *Annals of Pure and Applied Logic 134,* 1, 63–82.

Papadimitriou, C. 1994. *Computational Complexity.* Addison-Wesley.

Pearce, D. 1997. A new logical characterisation of stable models and answer sets. In *Non-Monotonic Extensions of Logic Programming*, J. Dix, L. Pereira, and T. Przymusinski, Eds. LNCS, vol. 1216. Springer, 57–70.

Pearce, D. 1999. From here to there: Stable negation in logic programming. In *What is Negation?*, D. Gabbay and H. Wansing, Eds. Kluwer, 161–181.

Pearce, D. 2004. Simplifying logic programs under answer set semantics. In *Proceedings of the 20th International Conference on Logic Programming* (*ICLP 2004*), B. Demoen and V. Lifschitz, Eds. LNCS, vol. 3132. Springer, 210–224.

Pearce, D. 2006. Equilibrium logic. *Annals of Mathematics and Artificial Intelligence 47*, 3–41.

Pearce, D., de Guzmán, I., and Valverde, A. 2000a. Computing equilibrium models using signed formulas. In *Proceedings of the 1st International Conference on Computational Logic* (*CL 2000*), J. Lloyd, V. Dahl, U. Furbach, M. Kerber, K.-K. Lau, C. Palamidessi, L. M. Pereira, Y. Sagiv, and P. J. Stuckey, Eds. LNCS, vol. 1861. Springer, 688–702.

Pearce, D., de Guzmán, I., and Valverde, A. 2000b. A tableau calculus for equilibrium entailment. In *Proceedings of the 9th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods* (*TABLEAUX 2000*), R. Dyckhoff, Ed. LNCS, vol. 1847. Springer, 352–367.

Pearce, D., Sarsakov, V., Schaub, T., Tompits, H., and Woltran., S. 2002. A polynomial translation of logic programs with nested expressions into disjunctive logic programs: Preliminary report. In *Proceedings of the 19th International Conference on Logic Programming* (*ICLP 2002*), P. Stuckey, Ed. LNCS, vol. 2401. Springer, 405–420.

Pearce, D., Tompits, H., and Woltran., S. 2001. Encodings for equilibrium logic and logic programs with nested expressions. In *Proceedings of the 10th Portuguese Conference on Artificial Intelligence* (*EPIA 2001*), P. Brazdil and A. Jorge, Eds. LNCS, vol. 2258. Springer, 306–320.

Pearce, D. and Valverde, A. 2004a. Synonymous theories in answer set programming and equilibrium logic. In *Proceedings 16th European Conference on Artificial Intelligence* (*ECAI 2004*). 388–392.

Pearce, D. and Valverde, A. 2004b. Towards a first order equilibrium logic for nonmonotonic reasoning. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence* (*JELIA 2004*), J. J. Alferes and J. A. Leite, Eds. LNCS, vol. 3229. Springer, 147–160.

Pearce, D. and Valverde, A. 2004c. Uniform equivalence for equilibrium logic and logic programs. In *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2003*), V. Lifschitz and I. Niemelä, Eds. LNCS, vol. 2923. Springer, 194–206.

Pearce, D. and Wagner, G. 1991. Logic programming with strong negation. In *Workshop on Extensions of Logic Programming, Proceedings*, P. Schroeder-Heiste, Ed. LNAI, vol. 475. Springer, 311–326.

Rintanen, J. 1999. Constructing conditional plans by a theorem prover. *Journal of Artificial Intelligence Research 10*, 323–352.

Russell, B. 1906. The theory of implication. *American Journal of Mathematics 28,* 2, 159–202.

Sagiv, Y. 1988. Optimising datalog programs. In *Foundations of Deductive Databases and Logic Programming*, J. Minker, Ed. Morgan Kaufmann, 659–698.

Sarsakov, V., Schaub, T., Tompits, H., and Woltran, S. 2004. nlp: A compiler for nested logic programming. In *Proceedings of the 7th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2003*), V. Lifschitz and I. Niemelä, Eds. LNCS, vol. 2923. Springer, 361–364.

Simons, P., Niemelä, I., and Soininen, T. 2002. Extending and implementing the stable model semantics. *Artificial Intelligence 138*, 181–234.

Srzednicki, J. and Stachniak, Z., Eds. 1998. *Lesniewski's Systems Protothetic.* Dordrecht.

Stockmeyer, L. J. 1976. The polynomial-time hierarchy. *Theoretical Computer Science 3,* 1, 1–22.

Tompits, H. and Woltran, S. 2005. Towards implementations for advanced equivalence checking in answer-set programming. In *Proceedings of the 21st International Conference on Logic Programming* (*ICLP 2005*), M. Gabbrielli and G. Gupta, Eds. LNCS, vol. 3668. Springer, 189–203.

Turner, H. 2001. Strong equivalence for logic programs and default theories (made easy). In *Proceedings of the 6th International Conference on Logic Programming and Nonmonotonic Reasoning* (*LPNMR 2001*), T. Eiter, W. Faber, and M. Truszczynski, Eds. LNCS, vol. 2173. Springer, 81–92.

Turner, H. 2003. Strong equivalence made easy: Nested expressions and weight constraints. *Theory and Practice of Logic Programming 3,* 4-5, 609–622.

Valverde, A. 2004. tabeql: A tableau based suite for equilibrium logic. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence* (*JELIA 2004*), J. J. Alferes and J. A. Leite, Eds. LNCS, vol. 3229. Springer, 734–737.

van Dalen, D. 1986. Intuitionistic logic. In *Handbook of Philosophical Logic, Volume III: Alternatives to Classical Logic*, D. Gabbay and F. Guenthner, Eds. Synthese Library, vol. 166. D. Reidel Publishing Co., Dordrecht, Chapter III.4, 225–339.

VOROB'EV, N. 1952. A constructive propositional calculus with strong negation (in Russian). *Doklady Akademii Nauk SSR 85*, 689–692.

WHITEHEAD, A. N. AND RUSSELL, B. 1910–1913. *Principia Mathematica.* Vol. 1–3. Cambridge University Press.

WOLTRAN, S. 2003. Quantified Boolean Formulas - From Theory to Practice. Ph.D. thesis, Technische Universität Wien, Institut für Informationssysteme.

WOLTRAN, S. 2004. Characterizations for relativized notions of equivalence in answer set programming. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence* (*JELIA 2004*), J. J. Alferes and J. A. Leite, Eds. LNCS, vol. 3229. Springer, 161–173.

WOLTRAN (ED.), S. 2005. Answer set programming: Model applications and proofs-of-concept. Tech. Rep. WP5, Working Group on Answer Set Programming (WASP, IST-FET-2001-37004). Available at `http://www.kr.tuwien.ac.at/projects/WASP/`.

WRATHALL, C. 1976. Complete sets and the polynomial-time hierarchy. *Theoretical Computer Science 3,* 1, 23–33.

YOU, J.-H., YUAN, L.-Y., AND MINGYI, Z. 2003. On the equivalence between answer sets and models of completion for nested logic programs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence* (*IJCAI 2003*), G. Gottlob and T. Walsh, Eds. Morgan Kaufmann, 859–865.