Ann Math Artif Intell (2006) 47: 3–41
DOI 10.1007/s10472-006-9028-z

Equilibrium logic

David Pearce

Received: 1 February 2006 / Accepted: 23 June 2006 /
Published online: 14 September 2006

Abstract Equilibrium logic is a general purpose nonmonotonic reasoning formalism closely aligned with answer set programming (ASP). In particular it provides a logical foundation for ASP as well as an extension of the basic syntax of answer set programs. We present an overview of equilibrium logic and its main properties and uses.

Keywords answer set programming · equilibrium logic · intermediate logics · strong negation

Mathematics Subject Classifications (2000) 03B50 · 03B55 · 03B70 · 68N17 · 68T30

1 Introduction

Equilibrium logic is an approach to nonmonotonic reasoning that was developed by the author in the Spring of 1995, and first published in [75] and later in slightly revised form in [76]. Since that time it has been further developed and studied by several authors and applied in different directions, mainly to problems arising in the foundations of logic programming under answer set semantics. The aim of this paper is to present an overview of equilibrium logic and summarise its main features, properties and areas of application. I also defend the underlying methodology of equilibrium logic which one might summarise via the following thesis: intermediate and multi-valued logics provide a useful basis on which to construct nonmonotonic logics for programming and knowledge representation tasks.

Equilibrium logic is closely associated with answer set programming (ASP), a relatively new paradigm of declarative programming. ASP, sometimes called

Partially supported by CICyT project TIC-2003-9001-C02, URJC project PPR-2003-39 and WASP (IST-2001-37004).

D. Pearce (✉)
Computing Science and Artificial Intelligence, Universidad Rey Juan Carlos, Madrid, Spain
e-mail: davidandrew.pearce@urjc.es

A-Prolog, refers to a family of logic programming languages that implement and extend the stable model and answer set semantics of Gelfond and Lifschitz [33, 34]. The best known systems are `DLV` [51], `GnT` [46], `smodels` [94], `CMODELS` [52], `ASSAT` [57] and *nomore*++ [1] which now provide practical and viable environments for tasks of knowledge representation and declarative problem solving. AI applications include planning and diagnosis, as exemplified in a prototype decision support system for the space shuttle [4], a prototype system, INFOMIX, for the management of heterogenous and inconsistent data in information systems,[1] the representation of ontologies in the semantic web allowing for default knowledge and inference, as discussed in [14], as well as compact and fully declarative representations of hard combinatorial problems such as n-Queens, Hamiltonian paths, and so on.[2] Initially the language of ASP was that of normal logic programs. Successively, the approach was extended to handle integrity constraints, disjunction and a second, strong negation operator. Equilibrium logic further enriches this syntax to a general propositional language by providing a simple, minimal model characterisation of answer sets in a well-known nonclassical logic. It therefore provides a natural generalisation of logical consequence under answer set semantics as well as a mathematical foundation for answer set programming.

In the last few years the language of ASP has also been extended to a richer basic syntax, e.g., that of programs with nested expressions [55], as well as to include additional features such as choice rules, weight constraints and aggregates. There now exist several efficient answer set solvers for normal and disjunctive programs, together with various extensions and front-ends designed to enhance the reasoning capabilities of these systems. Extending the language of ASP is only one reason to be interested in equilibrium logic. Beyond this feature equilibrium logic provides:

1. A general methodology for building nonmonotonic logics;
2. A logical and mathematical foundation for ASP-type systems, enabling one to prove useful metatheoretic properties;
3. Further means of comparing ASP with other approaches to nonmonotonic reasoning.

Since equilibrium logic generalises reasoning with stable models and answer sets, *a fortiori* it relates closely to other applied logics such as default logic, autoepistemic logic or modal nonmonotonic systems. In this paper, however, I will focus mainly on those aspects that relate to logic programming and deductive databases, rather than more general knowledge representation formalisms. I assume the reader has some familiarity with the rudiments of ASP and do not repeat the usual definitions of answer sets. Many of the results presented below have been proved elsewhere. I include proofs for most of the material that is new.

2 The semantics of equilibrium logic

Equilibrium logic is based on the nonclassical logic of here-and-there, which we denote by HT, and on its least extension by strong negation, which we denote by

[1]See [50] and http://sv.mat.unical.it/infomix/.

[2]For these and other examples as well as a thorough introduction to ASP, see [5].

N_5. N is typically used to symbolise Nelson's constructive logic with strong negation, and our strengthening of N is a logic with five truth values; hence the designation N_5. Formulas of HT are built-up in the usual way using the logical constants: $\wedge$, $\vee$, $\rightarrow$, $\neg$, standing respectively for conjunction, disjunction, implication and weak (or intuitionistic) negation. The axioms and rules of inference for HT are those of intuitionistic logic, ie. the axiom schemata:

I1. $\alpha \rightarrow (\beta \rightarrow \alpha)$	I2. $(\alpha \wedge \beta) \rightarrow \alpha$
I3. $(\alpha \rightarrow (\beta \rightarrow \gamma)) \rightarrow ((\alpha \rightarrow \beta) \rightarrow (\alpha \rightarrow \gamma))$	I4. $(\alpha \wedge \beta) \rightarrow \beta$
I5. $(\alpha \rightarrow \beta) \rightarrow ((\alpha \rightarrow \gamma) \rightarrow (\alpha \rightarrow (\beta \wedge \gamma)))$	I6. $\alpha \rightarrow (\alpha \vee \beta)$
I7. $(\alpha \rightarrow \gamma) \rightarrow ((\beta \rightarrow \gamma) \rightarrow ((\alpha \vee \beta) \rightarrow \gamma))$	I8. $\beta \rightarrow (\alpha \vee \beta)$
I9. $(\alpha \rightarrow \neg\beta) \rightarrow (\beta \rightarrow \neg\alpha)$	I10. $\neg(\alpha \rightarrow \alpha) \rightarrow \beta$

and the rule of *modus ponens*, together with the axiom schema

$$(\neg\alpha \rightarrow \beta) \rightarrow (((\beta \rightarrow \alpha) \rightarrow \beta) \rightarrow \beta). \tag{1}$$

which characterises the three-valued here-and-there logic of Heyting [42] and Gödel [37]; hence it is sometimes known as Gödel's three-valued logic, though it was first axiomatised by Łukasiewicz [58]. Other axioms may be used in place of (1). An example is the axiom of Hosoi [43]:

$$\alpha \vee (\neg\beta \vee (\alpha \rightarrow \beta)) \tag{2}$$

The concept of *strong negation* was introduced into logic by Nelson [65] and later axiomatised by Vorob'ev[101, 102]. We denote the strong negation operator by '$\sim$'. N_5 is obtained by adding the new negation $\sim$ together with the following axiom schemata (where '$\alpha \leftrightarrow \beta$' abbreviates $(\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha)$):

N1. $\sim (\alpha \rightarrow \beta) \leftrightarrow \alpha \wedge \sim\beta$	N2. $\sim(\alpha \wedge \beta) \leftrightarrow \sim\alpha \vee \sim \beta$
N3. $\sim (\alpha \vee \beta) \leftrightarrow \sim\alpha \wedge \sim\beta$	N4. $\sim \sim\alpha \leftrightarrow \alpha$
N5. $\sim \neg\alpha \leftrightarrow \alpha$	N6. (for atomic α) $\sim\alpha \rightarrow \neg\alpha$

taken from the calculus of Vorob'ev [101, 102]. The derivability relation for HT (resp. N_5) is denoted by $\vdash_{HT}$ (resp. $\vdash_{N_5}$). However we usually drop the subscript whenever the context is clear.

The model theory of HT is based on the usual Kripke semantics for intuitionistic logic (see eg. [15]), but HT is complete for Kripke frames $\mathcal{F} = \langle W, \leq \rangle$ (where as usual W is the set of points or worlds and $\leq$ is a partial-ordering on W) having exactly two worlds say h ('here') and t ('there') with $h \leq t$. As usual a *model* is a frame together with an assignment i that associates to each element of W a set of *atoms*, such that if $w \leq w'$ then $i(w) \subseteq i(w')$. An assignment is then extended inductively to all formulas via the usual rules for conjunction, disjunction, implication and (weak) negation in intuitionistic logic, namely

$$(\varphi \wedge \psi) \in i(w) \text{ iff } \varphi \in i(w) \text{ and } \psi \in i(w)$$

$$(\varphi \vee \psi) \in i(w) \text{ iff } \varphi \in i(w) \text{ or } \psi \in i(w)$$

$$(\varphi \rightarrow \psi) \in i(w) \text{ iff for all } w' \text{ s.t. } w \leq w',\ \varphi \in i(w') \text{ implies } \psi \in i(w')$$

$$\neg\varphi \in i(w) \text{ iff for all } w' \text{ s.t. } w \leq w',\ \varphi \notin i(w')$$

Evidently the final clause means that $\neg\varphi$ is true at either world just in case $\varphi \notin i(t)$. It is convenient to represent an HT model as an ordered pair $\langle H, T\rangle$ of sets of atoms, where $H = i(h)$ and $T = i(t)$ under a suitable assignment i; by $h \leq t$, it follows that $H \subseteq T$.

We write $\mathcal{M}, w \models \varphi$ to denote that a formula φ is true in an HT model $\mathcal{M}$ at world w. A formula φ is simply *true* in an HT model $\mathcal{M}$, in symbols $\mathcal{M} \models \varphi$, if $\mathcal{M}, w \models \varphi$ holds for each world in $\mathcal{M}$, equivalently if $\mathcal{M}, h \models \varphi$. A formula φ is said to be *valid* in HT, in symbols $\models \varphi$, if it is true in all HT models. A set of HT or N_5 sentences is called a *theory*, and a theory is said to be *consistent* if it has a model. Logical consequence for HT is understood as follows: φ is said to be an HT consequence of a theory Π, written $\Pi \models \varphi$, iff for all models $\mathcal{M}$ and any world $w \in \mathcal{M}$, $\mathcal{M}, w \models \Pi$ implies $\mathcal{M}, w \models \varphi$. Equivalently this can equally be expressed by saying that φ is true in all models of Π.

A suitable semantics for N_5 is obtained by modifying the Kripke semantics as follows. Kripke here-and-there frames are defined as before, but now in models each world is assigned a set of *literals*, where the term 'literal' denotes an atom or atom prefixed by strong negation. The assignment is consistent in the sense that no world is assigned both an atom and its strong negation. The truth assignment is then extended inductively to all formulas using the following additional clauses governing strongly negated formulas (see e.g. [39]):

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

Analaogous to the case of HT, it is useful to represent an N_5-model as an ordered pair $\langle H, T\rangle$ of sets of literals, where again $H \subseteq T$. Truth, validity and logical consequence for N_5 are defined as before.

An important and useful property of HT is the fact that it is the strongest intermediate logic (i.e. strengthening of intuitionistic logic) that is properly contained in classical logic. Moreover it in turn properly contains all other such intermediate logics. The fact that N_5 is the least strong negation extension of HT means that it is a conservative extension of HT, in other words a formula without strong negation is a theorem of HT if and only if it is a theorem of N_5. It also means that many key metalogical properties, including interpolation [62], are carried over from HT. For a detailed study, see [48].

Here are some useful theorems and non-theorems of HT and N_5, easily established by using the model theory, that illustrate some key differences with respect to both classical and intuitionistic logic.

$$\begin{array}{ll}
\vdash \neg\alpha \vee \neg\neg\alpha & \nvdash \alpha \vee \neg\alpha\\
\vdash \neg(\alpha \wedge \beta) \leftrightarrow (\neg\alpha \vee \neg\beta) & \nvdash \neg\neg\alpha \rightarrow \alpha\\
\vdash \neg(\alpha \vee \beta) \leftrightarrow (\neg\alpha \wedge \neg\beta) & \nvdash ((\alpha \rightarrow \beta) \rightarrow \alpha) \rightarrow \alpha
\end{array}$$

Each of the above theorems fails in intuitionistic logic, while adding any of the above non-theorems to HT would yield classical two-valued logic. As another example we note that contraposition holds for weak negation but not for strong negation:

$$\alpha \rightarrow \beta \ \vdash \ \neg\beta \rightarrow \neg\alpha \qquad\qquad \alpha \rightarrow \beta \ \nvdash \ \sim\beta \rightarrow \sim\alpha$$

while for the law of double negation the situation is reversed:

$$\vdash {\sim}{\sim}\alpha \leftrightarrow \alpha \qquad\qquad \nvdash \neg\neg\alpha \leftrightarrow \alpha$$

A further feature of N_5 is worth mentioning here, since it will be useful later on. It is actually a property of the logic N shown by Gurevich [39] for the case of first-order logic. Adapted to propositional N_5 his observation amounts to the following. Let us say that a formula φ of N_5 is in *reduced form* if strong negation '$\sim$' is 'driven-in' to stand directly in front of atoms. Vorob'ev [101, 102] already showed that every formula is equivalent to a formula in reduced form. Let φ be in reduced form and for any propositional variable p in φ replace each occurrence of ${\sim}p$ by a new variable p', to form a strong negation-free formula, say φ'. Given any set of such formulas Π, set $\Pi' = \{\varphi' : \varphi \in \Pi\}$. Let S be the set of all formulas $p' \rightarrow \neg p$. Then for any formula φ,

$$\Pi \vdash_{N_5} \varphi \ \text{ iff } \ \Pi' \cup S \vdash_{HT} \varphi'. \tag{3}$$

This property is important because it can be used to reduce the completeness of N with respect to the generalised Kripke semantics to the ordinary completeness of intuitionistic logic.[3] It is also important for proving properties of N_5 and for implementing strong negation in ASP systems.

Lastly, let us mention here the extension of N_5 obtained by adding the axiom schema $\neg\neg\alpha \leftrightarrow \alpha$. As it is a 3-valued extension of N_5 we denote this logic by N_3. It has been studied in [39, 99] where it is called *classical logic with strong negation* (it is a conservative extension of classical logic). In the context of equilibrium logic, N_3 is important as it is precisely the logic of *total* models (see the next section) that are used in the equilibrium construction and in proof-theoretic systems.

2.1 Equilibrium logic

Equilibrium models are special kinds of minimal N_5 Kripke models. We first define a partial ordering $\trianglelefteq$ on N_5 models.

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

This leads to the following notion of equilibrium.

Definition 2 Let Π be a set of N_5 formulas and $\langle H, T\rangle$ a model of Π.

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

In other words a model $\langle H, T\rangle$ of Π is in equilibrium if it is total and there is no model $\langle H', T\rangle$ of Π with $H' \subset H$. Equilibrium logic is the logic determined by the equilibrium models of a theory. We define a notion of equilibrium entailment as follows.

[3] Actually, Gurevich showed this for the case of the empty background theory Π, but the more general form is easily deducible.

Definition 3 (Equilibrium entailment) The relation $\mid\!\sim$, called *equilibrium entailment*, is defined as follows. Let Π be a set of formulas.

1. If Π is non-empty and has equilibrium models, then $\Pi \mid\!\sim \varphi$ if every equilibrium model of Π is a model of φ in N_5.
2. If either Π is empty or has no equilibrium models, then $\Pi \mid\!\sim \varphi$ if $\Pi \models \varphi$.

If strong negation is not present in the language, in all the above the logic N_5 is replaced by HT.

A few words may help to explain the concept of equilibrium entailment. First, we define the basic notion of entailment as truth in every intended (equilibrium) model. In nonmonotonic reasoning this is a common approach and sometimes called a *sceptical* or *cautious* notion of entailment or inference; its counterpart *brave* reasoning being defined via truth in some intended model. Since equilibrium logic is intended to provide a logical foundation for the answer set semantics of logic programs, the cautious variant of entailment is the natural one to choose: the standard consequence relation associated with answer sets is given by truth in all answer sets of a program [53]. Note however that in ASP as a programming paradigm each answer set may correspond to a particular solution of the problem being modelled and is therefore of interest in its own right.

Secondly, it is useful to have a nonmonotonic consequence or entailment relation that is non-trivially defined for all consistent theories. As we shall see below, however, not all such theories possess equilibrium models. For such cases it is natural to use N_5 consequence as the entailment relation since, as we shall observe later on, N_5 is the strongest logic with the property that logically equivalent theories have the same equilibrium models. Evidently, situation 2 also handles correctly the cases that Π is empty or inconsistent.

2.2 Relation to answer sets

Stable models for normal logic programs, whose rules have atomic heads and may contain default negation in their bodies, were defined in [33]. The notion of answer set, introduced in [34, 35], extended this definition to cover disjunctive programs and programs permitting an additional, strong negation operator, representing explicit falsity (as in (4) below). In each case the definition involved made use of a fixpoint condition involving a certain 'reduct' operator. Subsequent extensions of the concept to cover other kinds of rules have also relied on a reduct operator of similar sort. For the original definitions, the reader is referred to the various papers cited.

For all the usual classes of logic programs equilibrium models correspond to answer sets. We consider here three kinds of logic program rules of increasing syntactic complexity. First. a ground rule of a disjunctive program has the form

$$K_1 \vee \ldots \vee K_k \leftarrow L_1, \ldots L_m, \mathrm{not} L_{m+1}, \ldots, \mathrm{not} L_n \tag{4}$$

where the L_i and K_j are literals. The 'translation' from the syntax of programs to N_5 propositional formulas is the trivial one, viz. (4) corresponds to the N_5 sentence

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

Second, we may consider *generalised disjunctive programs*, discussed in Lifschitz [53], whose rules have the form

$$K_1 \vee \ldots \vee K_k \vee \mathrm{not} K_{k+1} \vee \ldots \vee \mathrm{not} K_l \leftarrow L_1, \ldots L_m, \mathrm{not} L_{m+1}, \ldots, \mathrm{not} L_n \qquad (5)$$

i.e. weak negation is allowed in rule heads. Again the translation is obvious. Thirdly, in [55] Lifschitz, Tang and Turner extended answer set semantics to cover so-called programs with nested expressions. Let us call them *nested programs* for short. Written as formulas of N_5, the expressions of nested programs have the form

$$\varphi \rightarrow \psi \qquad (6)$$

where $\varphi, \psi \in \mathcal{L}(\wedge, \vee, \neg, \sim)$, i.e., the antecedent and consequent are implication-free.[4] Clearly, (5) is a special case of (6) and (4) is a special case of (5).

Under this trivial translation of program rules to propositional N_5, every ground logic program can be identified with an N_5 theory. Then we have:

Proposition 1 Let Π be a consistent logic program (of any kind). Then $\langle T, T \rangle$ is an equilibrium model of Π if and only if T is an answer set of Π.

This was first shown for disjunctive programs in [75, 76] and for nested programs in [81] (see also [54]).

Recently, using a new concept of reduct, Ferraris [31] has shown how answer sets may be defined for arbitrary propositional formulas. Again, as Ferraris shows, the correspondence to equilibrium models is exact, as in Proposition 1.

The standard notion of entailment or consequence for programs under the answer set semantics is that a query Q is entailed by a program Π is Q is true in all answer sets of Π, see eg [3, 53]. Let us denote this entailment or consequence relation by $\mid\!\sim_{AS}$. Evidently, literals are true in an answer set if and only if they belong to it. Conjunctions and disjunctions are handled in the obvious way (e.g., [3, 55]). Sometimes, as in [3], queries of the form *not a*, or in logical notation $\neg a$, are not explicitly dealt with. However it seems to be in keeping with the semantics to regard a formula of form *not* α or $\neg\alpha$ to be true in an answer set if and only if α is not true. Another way to express this would be to say that an answer set satisfies $\neg\alpha$ if it does not violate the constraint $\{\leftarrow \alpha\}$, where constraint violation is understood as in [55].

As mentioned earlier, not all consistent theories or programs possess equilibrium models or answer sets; those that do we shall call *coherent*. Then as an immediate consequence of Proposition 1 we have:

Corollary 1 Let Π be a coherent program and let $\mid\!\sim_{AS}$ be the relation of consequence under answer sets semantics. Then for any formula φ, $\Pi \mid\!\sim_{AS} \varphi$ iff $\Pi \mid\!\sim \varphi$.

[4] Actually they require strong negations to be driven-in to stand in front of atoms, but as we know from our logic this is no genuine restriction.

2.2.1 *Equilibrium logic as an extension of the language of ASP*

Evidently equilibrium logic extends the usual basic language of answer set programming. In fact it subsumes three main language extensions of ordinary logic programming. We might call them *hypothetical* logic programs, *nested programs* and programs with *nested rules*, respectively. All three types of syntax assume that programs comprise rules linking a body to a head. They differ however in the form in which the head and body are allowed to take. Towards the end of the 1980s it became quite common practice to consider logic programs with conditional goals and hence implications in the bodies of rules; this idea being especially well-developed in intuitionistic logic programming [59]. The resulting programs, containing implications and even embedded implications in rule bodies, are often called *hypothetical* programs.The main technical problem to overcome is how to combine this extended syntax with negation-as-failure. Several different approaches were explored, e.g. [7, 63], usually under certain restrictive assumptions about the programs concerned, and [63] in particular was able to use hypothetical programs to characterise parametric modules and abstract data types. Shortly afterwards Dung [23] analysed hypothetical programs in a framework extending the stable model semantics. He also provided examples (including the formalisation of a fragment of the British Nationality Act) to show the usefulness of this language extension of stable semantics. A more comprehensive approach to hypothetical programs was later offered by Giordano and Olivetti [36] and further generalisations in the context of stable models can be found in [93].

In the case of nested programs, as we have seen there are no occurrences of implication in the bodies or heads of rules. However, Lifschitz, Tang and Turner [55] who introduced this concept into ASP give examples to motivate the naturalness of allowing arbitrary nestings of conjunction, disjunction and negation in the bodies and heads of rules. A key argument for the practical value of nested programs was later provided in [30] where it was shown that so-called *weight constraints* can be represented using rules with nested expression.

A third type of language extension consists in allowing conditional *heads* of program rules; eg in a rule of form (4) each K_i may be a conditional whose antecedent is a conjunction of atoms and weakly negated atoms and whose consequent is an atom. These are called *programs with nested rules* in [38], where a semantics extending stable models is defined and complexity and expressiveness is studied. Again the extension is motivated by examples expressing certain types of everyday knowledge and reasoning.

Equilibrium logic permits the full use of the propositional connectives and, as mentioned, there has recently been a proposal for extending the definition of answer set to arbitrary propositional formulas in $\mathcal{L}(\wedge, \vee, \rightarrow \neg, \sim)$, [31], yielding a semantics equivalent to equilibrium logic. This work makes a strong case for the use of the full propositional language by showing that the important concept of *aggregate* in ASP, as developed in [29], can be represented by rules with embedded implications. In Section 4 we shall consider some examples showing the behaviour of embedded implications in program rules.

Although the syntax of equilibrium logic subsumes the three types of language extensions just mentioned, it is important to bear in mind that it was developed, first and foremost, as a natural logical foundation for the answer set semantics of disjunctive programs. There was therefore no attempt to match the intuitions lying behind some of the approaches say to hypothetical programs or to programs with nested rules, and so the resulting semantics may well be different (and can easily be seen to be different in some cases). However the fact that the extensions of answer set semantics described in [55] and [31] coincide with equilibrium logic shows that formally quite different approaches and methodologies may nevertheless converge to a similar result.

2.3 Some examples

Let us now consider some simple examples to see how equilibrium models can be computed in practice. We look briefly at some coherent and non-coherent formulas. A simple well-known instance of a non-coherent theory, Π_1, is the single formula $\neg a \rightarrow a$, where a is an atom. This formula is logically equivalent to the formula $\neg\neg a$. To validate this formula clearly a must be true 'there', but it need not be true 'here', hence $\langle \{\}, \{a\} \rangle$ is an N_5-model showing that Π_1 has no equilibrium model.

Consider the theory Π_2 comprising two formulas $\sim b$ and $\neg a \rightarrow b$. By the properties of strong negation and contraposition, we obtain $\sim b \vdash \neg b$ and $\neg a \rightarrow b \vdash \neg b \rightarrow \neg\neg a$, so we can also derive $\neg\neg a$ from Π_2, but we cannot derive a. So the candidate total model $\langle \{a, \sim b\}, \{a, \sim b\} \rangle$ has a 'smaller' model $\langle \{\sim b\}, \{a, \sim b\} \rangle$, showing again that Π_2 has no equilibrium model.

Consider, however, the modified theory Π_3 comprising $\sim b$ and $c \wedge \neg a \rightarrow b$. $\neg\neg a$ is no longer derivable and Π_3 is coherent with a single equilibrium model $\langle \{\sim b\}, \{\sim b\} \rangle$. As another example consider $\Pi_4 = \{a \rightarrow c; b \rightarrow c; \neg a \rightarrow b\}$. From the last formula any 't'-world must verify either a or b. So Π_4-models must have either $\{a, c\}$ or $\{b, c\}$ true at t. The least of these are the models $\langle \{\}, \{a, c\} \rangle$ and $\langle \{b, c\}, \{b, c\} \rangle$. Clearly the second cannot be further 'reduced' and is therefore the only equilibrium model of Π_4.

Lastly, consider the theory Π_5 comprising the single formula $\neg\neg a \rightarrow a$. Evidently, it is satisfied by a model having a true at the 't'-world, providing a is also true 'here'. So $\langle \{a\}, \{a\} \rangle$ is an equilibrium model. Alternatively, the formula is satisfied by any model in which a is false. The only candidate is $\langle \{\}, \{\} \rangle$, so Π_5 has two equilibrium models. Since one of the answer sets or equilibrium models is a proper subset of the other, this shows that Π_5 which has the form of a nested program is not logically equivalent to a disjunctive logic program, by a well-known property of answer sets. In Section 4 and Section 5 below, we shall consider some examples involving embedded implications.

2.4 Many-valued semantics for N_5

The Kripke semantics for N_5 logic can be easily characterised using a many-valued approach, specifically with a five-valued logic.

The set of truth values used in the many-valued characterisation is

$$5 = \{-2, -1, 0, 1, 2\}$$

where 2 is the designated value. A truth-value assignment σ behaves as follows. The connective $\wedge$ is interpreted as the minimum function, $\vee$ is the maximum function, and $\sigma(\sim\varphi) = -\sigma(\varphi)$,

$$\sigma(\alpha \to \beta) = \begin{cases} 2 & \text{if } \sigma(\alpha) \leq 0 \text{ or } \sigma(\alpha) \leq \sigma(\beta) \\ -1 & \text{if } \sigma(\alpha) = 1 \ \& \ \sigma(\beta) = -2 \\ \sigma(\beta) & \text{otherwise} \end{cases}$$

$$\sigma(\neg\alpha) = \begin{cases} 2 & \text{if } \sigma(\alpha) \leq 0 \\ -\sigma(\alpha) & \text{otherwise} \end{cases}$$

The truth tables for the connectives $\to$, $\neg$ and $\sim$ are therefore:

$\to$	-2	-1	0	1	2
-2	2	2	2	2	2
-1	2	2	2	2	2
0	2	2	2	2	2
1	-1	-1	0	2	2
2	-2	-1	0	1	2

$\neg$	
-2	2
-1	2
0	2
1	-1
2	-2

$\sim$	
-2	2
-1	1
0	0
1	-1
2	-2

Any N_5 model σ as a truth-value assignment can trivially be converted into a Kripke model $\langle H, T\rangle$, and *vice versa*. For example, if σ is an assignment and p is a propositional variable, then the corresponding Kripke model, denoted by $\mathcal{M}_\sigma$, is determined by the equivalences:

$$\begin{array}{rcl} \sigma(p) = 2 & \text{iff} & p \in H \\ \sigma(p) = 1 & \text{iff} & p \in T, p \notin H \\ \sigma(p) = 0 & \text{iff} & p \notin T, \sim p \notin T \\ \sigma(p) = -1 & \text{iff} & \sim p \in T, \sim p \notin H \\ \sigma(p) = -2 & \text{iff} & \sim p \in H \end{array}$$

The many-valued semantics and the Kripke semantics for N_5 are equivalent. In other words, if Π is a set of formulas in N_5 and ψ is a formula, then $\Pi \models \psi$ iff for every assignment σ in N_5, if $\sigma(\varphi) = 2$ for every $\varphi \in \Pi$, then $\sigma(\psi) = 2$. Note too that assignments or truth-value interpretations can also be considered partially ordered by the $\trianglelefteq$ relation. We then say for example that an assignment σ is greater than or equal to an assignment τ, if $\mathcal{M}_\tau \trianglelefteq \mathcal{M}_\sigma$.

2.5 Equilibrium logic as a fixpoint logic

Soon after its inception it became apparent that equilibrium logic could be given an alternative presentation in a more syntactical style, as a kind of fixpoint logic. The idea here is that instead of considering selected intended models of a theory, one considers certain kinds of extensions of the theory. This technique is well-known in default and autoepistemic logics.

Given a theory Π, we let At_Π and Lit_Π (usually dropping subscripts) denote the sets of atoms and literals, respectively, in the language of Π. In the usual way let us say that a theory Π is *complete* if for all sentences φ in the language of Π, we have $\Pi \vdash \varphi$ or $\Pi \vdash \neg\varphi$. Evidently every equilibrium model defines a complete theory. It turns out that this theory has a special form, as follows.

Definition 4 Let Π be a theory. A set E of sentences extending Π is said to be a *completion* of Π iff $E = \mathrm{Cn}(\Pi \cup \{\neg\varphi : \varphi \notin E\})$, where Cn is the consequence operator of N_5.

Equilibrium models correspond precisely to completions. For any model $\mathcal{M}$, set $Th(\mathcal{M}) = \{\varphi : \mathcal{M} \models \varphi\}$.

Proposition 2 [[77]] For any theory Π there is a one-one correspondence between the equilibrium models of Π and the completions of Π. In particular, E is a completion of Π iff $E = Th(\mathcal{M})$ for some equilibrium model $\mathcal{M}$ of Π. Similarly, $\mathcal{M} = \langle T, T\rangle$ is an equilibrium model of Π iff $T = E \cap Lit$, for some completion E of Π.

This characterisation of equilibrium logic is a useful one and we shall consider some applications of it below.

Let us lastly consider a refinement of Proposition 2. Let us say that a set E of sentences extending Π is an *atomic completion* of Π iff $E = \mathrm{Cn}(\Pi \cup \{\neg a : a \in \mathrm{At}\ \&\ a \notin E\})$. Then we have:

Proposition 3 [[78]] Let Π be a disjunctive logic program. Then there is a one-one correspondence between the equilibrium models (hence answer sets) of Π and the atomic completions of Π. In particular, E is an atomic completion of Π iff $E = Th(\mathcal{M})$ for some equilibrium model $\mathcal{M}$ of Π. Similarly, $\mathcal{M} = \langle T, T\rangle$ is an equilibrium model of Π iff $T = E \cap Lit$, for some atomic completion E of Π.

3 Some properties of equilibrium inference

Since the early days of research on nonmonotonic logics it became commonplace to study and compare logical systems with respect to general conditions on inference that they satisfy. The aim was not only to classify systems without monotonicity but also to select properties considered to be especially interesting or desirable. These properties were catalogued in works such as [49, 60] which established many of the standard conditions on inference that have been studied thereafter. For an authoritative account see [61].

Likewise it became a matter of routine to compare different approaches to the semantics of logic programs according to the abstract properties satisfied by their associated inference relations, see eg [19, 20] for the case of normal programs. In this respect the consequence relation associated with the stable model semantics did not fare particularly well, since it fails some 'desirable' properties like cumulativity and rationality (see below) that hold for some rivals such as the well-founded semantics [19]. For a while this was regarded by some critics as a negative feature of the semantics. Later, with the rise of efficient answer set solvers and the practical viability of ASP as a KR and programming paradigm, such criticisms no longer seem telling. One may be willing to forgo certain desirable properties of the inference relation if there is a trade-off with respect to expressiveness and applicability of the formalism.

Evidently it follows from Corollary 1 that these 'negative' features of the consequence relation of answer sets are inherited by equilibrium inference. However the news is by no means all bad: equilibrium entailment is in many respects well-behaved and some properties that fail do so as a result of the non-classical nature of the underlying logic.

Let us now consider some of the properties of $\mid\!\sim$ as a nonmonotonic inference relation in more detail. The three classical Tarski conditions on inference are *reflexivity*, *cut* and *monotony*. Equilibrium inference satisfies the first two.

Proposition 4 Equilibrium inference $\mid\!\sim$ satisfies reflexivity i.e. (a) If $\varphi \in \Pi$ then $\Pi \mid\!\sim \varphi$, and cut: (b) If $\forall i \in I$, $\Pi \mid\!\sim \psi_i$ and $\Pi \cup \{\psi_i : i \in I\} \mid\!\sim \varphi$, then $\Pi \mid\!\sim \varphi$.

Proof Left to the reader.

It is well known that the consequence relation of answer set semantics fails the following property of cautious monotony.

$$\Pi \mid\!\sim \varphi, \Pi \mid\!\sim \psi \Rightarrow \Pi \cup \varphi \mid\!\sim \psi$$

Hence this fails for equilibrium inference as well, so that $\mid\!\sim$ is not a cumulative inference relation. However $\mid\!\sim$ does satisfy the following two special cases.

Proposition 5 (a) If $\Pi \vdash \varphi$ or $\Pi \vdash \psi$, then $\Pi \mid\!\sim \varphi, \Pi \mid\!\sim \psi \Rightarrow \Pi \cup \varphi \mid\!\sim \psi$. (b) $\Pi \mid\!\sim \neg\varphi, \Pi \mid\!\sim \psi \Rightarrow \Pi \cup \neg\varphi \mid\!\sim \psi$.

Proof (a) Clearly, if $\Pi \vdash \psi$ then $\Pi \cup \varphi \mid\!\sim \psi$, for any φ. On the other hand, if $\Pi \vdash \varphi$ then Π and $\Pi \cup \varphi$ have the same equilibrium models. So if $\Pi \mid\!\sim \psi$ then $\Pi \cup \varphi \mid\!\sim \psi$. (b) It is easy to see that if a formula φ is false in all equilibrium models of a program Π, then Π and $\Pi \cup \neg\varphi$ have the same equilibrium models. □

The following three properties involve logical connectives. They are known respectively as *disjunction in the antecedent*, *proof by cases*, and *conditionalisation*.

$$\Pi \cup \varphi \mid\!\sim \alpha, \Pi \cup \psi \mid\!\sim \alpha \Rightarrow \Pi \cup (\varphi \vee \psi) \mid\!\sim \alpha$$

$$\Pi \cup \varphi \mid\!\sim \alpha, \Pi \cup {\sim}\varphi \mid\!\sim \alpha \Rightarrow \Pi \mid\!\sim \alpha$$

$$\Pi \cup \varphi \mid\!\sim \psi \Rightarrow \Pi \mid\!\sim \varphi \rightarrow \psi$$

Of these principles, the second, proof by cases, fails. In fact it is not a valid principle of constructive reasoning and fails already in the underlying monotonic logic N_5. To see this, let Π be empty and set $\alpha = (\varphi \vee {\sim}\varphi)$.

Proposition 6 Equilibrium inference satisfies disjunction in the antecedent and conditionalisation.

Proof For the first property observe that every equilibrium model of $\Pi \cup (\varphi \vee \psi)$ is a total model of Π in which either φ holds or ψ holds. It is easily seen that in the former case it must be an equilibrium model of $\Pi \cup \varphi$ and in the latter case it is an equilibrium model of $\Pi \cup \psi$. In each case by assumption α is true in the model. In case the theories concerned are consistent but have no equilibrium model, we apply the relation $\vdash$. Clearly the models of $\Pi \cup (\varphi \vee \psi)$ are contained in the union of the models of $\Pi \cup \varphi$ and $\Pi \cup \psi$; but in each such model α is true.

For the second property, suppose for the contradiction that $\Pi \not\mathrel{|\!\sim} \varphi \rightarrow \psi$. Then Π has an equilibrium model $\mathcal{M}$ in which φ is true and ψ is false. Clearly, $\mathcal{M}$ must be an equilibrium model of $\Pi \cup \varphi$, implying that $\Pi \cup \varphi \not\mathrel{|\!\sim} \psi$. If there are no equilibrium models we apply the deduction theorem for $\vdash_{N_5}$. □

Let us turn to some properties involving negation. One of the most commonly discussed is rational monotony or *rationality*, which can be formulated as a principle governing either weak or strong negation as follows:[5]

$$\Pi \mathrel{|\!\sim} \psi, \Pi \cup \varphi \not\mathrel{|\!\sim} \psi \Rightarrow \Pi \mathrel{|\!\sim} {\sim}\varphi$$

$$\Pi \mathrel{|\!\sim} \psi, \Pi \cup \varphi \not\mathrel{|\!\sim} \psi \Rightarrow \Pi \mathrel{|\!\sim} \neg\varphi$$

where clearly the former principle is stronger. Both principles fail in answer sets or equilibrium logic (see [74]). However the following, weaker form of rationality does hold.

Proposition 7 If (a) $\Pi \mathrel{|\!\sim} \psi$ and (b) $\Pi \cup \varphi \mathrel{|\!\sim} \neg\psi$ then $\Pi \mathrel{|\!\sim} \neg\varphi$.

Proof Assume (a) and (b) and suppose for the contradiction that $\Pi \not\mathrel{|\!\sim} \neg\varphi$. Then Π has an equilibrium model $\mathcal{M}$ in which φ is true. It is easily seen that $\mathcal{M}$ must be an equilibrium model of $\Pi \cup \varphi$, and so by (b) ψ is false in $\mathcal{M}$. This contradicts (a) that $\Pi \mathrel{|\!\sim} \psi$. □

An inverted form of weak rationality also holds:

$$\Pi \cup \varphi \mathrel{|\!\sim} \psi, \quad \Pi \mathrel{|\!\sim} \neg\psi \Rightarrow \Pi \mathrel{|\!\sim} \neg\varphi$$

as does the closely related principle of *modus tollens* (for weak negation):

$$\Pi \mathrel{|\!\sim} \varphi \rightarrow \psi, \quad \Pi \mathrel{|\!\sim} \neg\psi \Rightarrow \Pi \mathrel{|\!\sim} \neg\varphi$$

The proof is straightforward. These principles may be of interest in the context of induction and inverse entailment, see eg [89].

3.1 Deductive bases

We have seen how equilibrium logic, and thus answer set inference, can be viewed as a nonmonotonic system extending N_5. So N_5 can be considered as a prime candidate to form a monotonic base logic for $\mathrel{|\!\sim}$. What does it mean in general to say that a logic L forms a well-behaved monotonic basis for a given nonmonotonic inference relation? In the study of abstract properties of nonmonotonic inference, there is agreement that three main conditions should be fulfilled.[6] In the first place the candidate logic L should form a sublogic of $\mathrel{|\!\sim}$, so the latter is a genuine extension of the former. Secondly, we want the result of nonmonotonic inference to be closed under L in the sense that anything L-derivable from a nonmonotonic consequence

[5]See [61] for different but equivalent formulations of this principle.

[6]The term deductive base defined below is taken from [17, 18]; however similar ideas can be found in [49, 61] and elsewhere.

is itself a consequence. And thirdly nonmonotonic inference should be indifferent to L-equivalent inputs in the sense that monotonically equivalent theories have the same nonmonotonic consequences. Let $\Pi_1 \equiv_L \Pi_2$ denote that $\Pi_1 \vdash_L \Pi_2$ and $\Pi_2 \vdash_L \Pi_1$. And let us use $\Pi_1 \approx \Pi_2$ to denote nonmonotonic equivalence, ie that $\Pi_1 \mid\!\sim \varphi \Leftrightarrow \Pi_2 \mid\!\sim \varphi$, for any φ.

Definition 5 Let $\mid\!\sim$ be any nonmonotonic inference relation. We say that a logic L with monotonic inference relation $\vdash_L$ is a *deductive base* for $\mid\!\sim$ iff (a) $\vdash_L \subseteq \mid\!\sim$; (b) If $\Pi \mid\!\sim \varphi$ and $\varphi \vdash_L \psi$, then $\Pi \vdash \psi$; (c) If $\Pi_1 \equiv_L \Pi_2$ then $\Pi_1 \approx \Pi_2$.

Furthermore, for later reference let us say that a deductive base is *strong* if it satisfies the additional condition:

$$\Pi_1 \not\equiv_L \Pi_2 \Rightarrow \quad \text{there exists} \quad \Gamma \quad \text{such that} \quad \Pi_1 \cup \Gamma \not\approx \Pi_2 \cup \Gamma.$$

In terms of nonmonotonic consequence operations, (b) and (c) correspond to conditions known as left absorption and right absorption, respectively, see [61].[7] It is clear that by its semantic construction equilibrium logic has N_5 as a deductive base. This base is actually maximal.

Proposition 8 N_5 is a maximal deductive base for equilibrium entailment.

Proof By the definition of equilibrium entailment it is immediate that conditions (a–c) of Definition 5 hold. There are many examples of classically equivalent formulas that have different answer sets or equilibrium models, so clearly N_3, classical logic with strong negation, fails right absorption or condition (c). For theories without strong negation it follows that HT is a maximal base since there is no proper extension of it below classical logic. Where strong negation is present there is one more case to consider. Kracht [48] shows that there is just one non-trivial strengthening of N_5 contained in the logic N_3. This is the four-valued logic N_4 formed by adding to N_5 the axiom $\neg{\sim}\varphi \leftrightarrow \neg\neg\varphi$. It is easy to see that N_4 fails condition (c). Consider for instance the theory Π comprising the two formulas $\neg{\sim}a \rightarrow b$ and $\neg b \rightarrow {\sim}a$ for atomic a, b. Evidently there are two equilibrium models $\langle\{b\}, \{b\}\rangle$ and $\langle\{{\sim}a\}, \{{\sim}a\}\rangle$. Clearly $\Pi \cup \{\neg\neg a\}$ and $\Pi \cup \{\neg{\sim}a\}$ are N_4-equivalent theories, yet their equilibrium models are different. In the former case $\langle\{b\}, \{b\}\rangle$ is now the only equilibrium model, while the second theory has no equilibrium models. Hence condition (c) fails and N_5 is a maximal base. □

In ordinary logic programming under answer set semantics the above examples are easily seen to establish that N_4 is not a base logic. Consider the two expressions $\neg{\sim}a$ versus $\neg\neg a$ for some atom a. the former can be written as the integrity constraint $\leftarrow {\sim}a$; its effect in a program is to eliminate any answer sets that contain ${\sim}a$. The second formula can be written as $a \leftarrow \neg a$; its effect can be to leave a program devoid of answer sets unless there are additional rules supporting a.

[7]In the terminology of [17] we therefore require of $\vdash_L$ and $\mid\!\sim$ that they form a *fully absorbing inferential frame*.

An immediate corollary of Proposition 8 is that providing we restrict attention to coherent programs N_5 is also a deductive basis for the consequence relation associated with answer set semantics. The proviso is needed because $\mid\!\sim$ is defined for all consistent theories while $\mid\!\sim_{AS}$ is not.

If we consider only intermediate logics and their strong negation extensions as candidate deductive bases for answer set consequence, then clearly N_5 is the greatest such basis. This also justifies our choice of N_5-inference as the natural relation to capture answer set or equilibrium inference in the case of theories having no answer set (equilibrium model). Any stronger logic would not provide a well-behaved basis.

The significance of Proposition 8 is quite considerable in answer set programming, especially in the area of program transformations. It tells us that N_5 is the strongest logic we can use to transform one program rule into another, perhaps simpler but logically equivalent rule, without changing the semantics of the program. This applies both to (sets of) formulas as well as to substitutions of equivalent subformulas. Similarly, it is the strongest logic we can use to reduce or simplify a program by removing literals that are known to be logically derivable.

4 Some applications

We now turn to some of the ways in which equilibrium logic and its underlying deductive base can be applied, mainly in the area of the foundations of answer set programming. We normally state results for the most general case of propositional theories but in certain special cases we also look at particular syntactic classes of logic programs.

4.1 Defeasibility and monotonic inference

We start by considering some of the ways in which monotonic inference may play a role in the use of equilibrium logic and ASP. Since we are dealing with theories under a nonmonotonic entailment relation, not all consequences of a theory Π will be preserved under extensions of Π. In fact we can divide the consequences of a theory Π into two distinct sorts: those which continue to be true in any extension of the theory and those which may be revoked by a suitable choice of extension. Let us call the former persistent and the latter defeasible. Thus for instance if Π is a deductive database that is subject to consistent updating, the persistent consequences will be precisely those that remain true in all updates. To simplify matters we consider only coherent extensions.

Definition 6 Let $\Pi \mid\!\sim \varphi$. φ is said to be *persistent* (wrt Π) if $\Pi' \mid\!\sim \varphi$, for any coherent Π' such that $\Pi \subseteq \Pi'$. Otherwise φ is said to be *defeasible*.

It is trivial to see that any N_5-consequence of a theory must be persistent. But in general we can characterise persistence and, hence defeasibility, as follows.

Proposition 9 Suppose $\Pi \mid\!\sim \varphi$. Then φ is persistent iff $\Pi \vdash \neg\neg\varphi$.

Proof If $\Pi \vdash \neg\neg\varphi$ then $\neg\neg\varphi$ is true in every extension of Π and so φ is true in every equilibrium model of any such extension, if it is coherent. Suppose that $\Pi \not\vdash \neg\neg\varphi$. Then there is a model $\langle H, T\rangle$ of Π such that $\langle H, T\rangle \not\models \neg\neg\varphi$, so $\langle H, T\rangle \models \neg\varphi$. Consider, the extension $\Pi \cup T$. Clearly $\langle T, T\rangle$ is an equilibrium model of $\Pi \cup T$ in which φ is false. So φ is defeasible. □

Notice that if we want to ensure persistence for all extensions, even those without equilibrium models, then we should require $\Pi \vdash \varphi$. Notice too that the condition for defeasibility, $\Pi \not\vdash \neg\neg\varphi$, can be re-expressed by saying that $\Pi \cup \neg\varphi$ is (N_5) consistent.

In the usual manner we say that, given a theory Π, a formula φ is *decidable* if either $\Pi \vdash \varphi$ or $\Pi \vdash \neg\varphi$ and *undecidable* otherwise. Let Π be any generalised disjunctive program with rules of form (5).[8] Let us denote by $\mathrm{Neg}(\Pi)$ the set of all literals L that appear weakly negated in Π apart from the degenerate rule $\neg L$, and let $\mathrm{Und}(\Pi)$ denote the set of undecidable literals of Π.

Proposition 10 ([79]) Every generalised disjunctive program is logically equivalent to a program Π such that $Neg(\Pi) \subseteq Und(\Pi)$.

In fact there is an effective method to transform a program Γ into a logically equivalent one in which the decidable literals in $\mathrm{Neg}(\Gamma)$ have been eliminated, [79].

A different but related notion is that of *stability*, a concept due to van Dantzig [16], see also Dummett [22]. Originally introduced in the context of intuitionistic mathematics, it is applicable to any superintuitionistic logic (we state it for N_5):

Definition 7 A formula φ is said to be *stable* in a theory Π if $\Pi \vdash \neg\neg\varphi \rightarrow \varphi$.

Obviously a decidable literal is stable, but the converse need not hold. While the decidable literals in a program are relevant for determining the composition of its answer sets, the stable literals are relevant for guaranteeing the existence of answer sets. Again what is important here is not the collection of all literals of the program but those that are prefixed by weak negation. Before stating the relation between stability and coherence we note the following property. If Π is any theory and $\langle T, T\rangle$ is an equilibrium model of Π, then by definition there is no model $\langle H, T\rangle$ of Π with $H \subset T$. Now in the case that Π is a disjunctive program with rules of the form (4) one can easily check that in addition there cannot be a model $\langle H, T'\rangle$ of Π with $T' \subset T$ (otherwise $\langle T', T\rangle \models \Pi$, contradicting the initial assumption). We might say, therefore, that on disjunctive programs equilibrium models are *t-minimal*. Using this fact we can show:

Proposition 11 Let Π be a disjunctive program with rules of form (4). Π has an equilibrium model or answer set if each literal in $Neg(\Pi)$ is stable in Π.[9]

[8]Note that by the reduction method described in "Second order reduction" below, any propositional theory is logically equivalent to such a program.

[9]Here I would like to correct an error in [79] where this result was mistakenly stated to hold for generalised disjunctive programs with rules of form (5). In fact the proof uses the *t*-minimality property of equilibrium models for *disjunctive* rules.

Proof Suppose each $L \in Neg(\Pi)$ is stable but Π has no equilibrium model. Then for any t-minimal total model $\langle T, T\rangle$ of Π (in the sense explained above), there is an $H \subset T$ such that $\langle H, T\rangle \models \Pi$. So for each rule $r \in \Pi$ of form (4), we have $\langle H, T\rangle \models r$. In particular by the truth conditions for r at world h, this means

$$\{L_1, \dots L_m\} \subseteq H \ \ \& \ \ \{L_{m+1}, \dots, L_n\} \cap T = \emptyset \Rightarrow \{K_1, \dots, K_k\} \neq \emptyset$$

By assumption each weakly negated literal L in r is stable, so $\langle H, T\rangle \models \neg\neg L \rightarrow L$. Therefore $L \in H \Leftrightarrow L \in T$ for each $L \in Neg(\Pi)$. So the above truth condition can be re-written as

$$\{L_1, \dots L_m\} \subseteq H \ \ \& \ \ \{L_{m+1}, \dots, L_n\} \cap H = \emptyset \Rightarrow \{K_1, \dots, K_k\} \neq \emptyset$$

But this implies that $\langle H, H\rangle \models r$, which is impossible by the minimality of $\langle T, T\rangle$. This contradicts the initial assumption. □

It is easy to see that the condition of stability is quite different from purely syntactic conditions such as signings, stratifications etc. The reader may readily construct simple examples that satisfy the condition of Proposition 11 but which do not possess signings or stratifications. Proposition 11 provides an unexpected link between two historically distinct and apparently quite independent concepts of stability. Not surprisingly, we cannot strengthen Proposition 11 so as to provide also a necessary condition for the existence of an answer set. Any attempt is bound to fail on purely complexity-theoretic grounds. Deciding whether a literal is stable is a co-NP complete problem, while deciding whether a nested or disjunctive logic program has an answer set is Σ_2^P-complete [28, 81, 82].

4.2 Types of equivalence

We look now at some different kinds of equivalence between formulas and theories and how logical equivalence can be used as a tool in ASP.

Note first that in N_5 there are two natural kinds of logical equivalence between formulas. We say that φ is *equivalent* to ψ if $\vdash \varphi \leftrightarrow \psi$. A stronger notion is that of full equivalence: we let $\varphi \cong \psi$ abbreviate $(\varphi \leftrightarrow \psi) \wedge (\sim\varphi \leftrightarrow \sim\psi)$ and we say that φ and ψ are *fully equivalent* iff $\vdash \varphi \cong \psi$. While the former relation is an equivalence, only the latter relation is a congruence in the Lindenbaum algebra. Semantically the difference is this: equivalent formulas have the same models while fully equivalent formulas have the same truth values in all interpretations under the many-valued semantics, ie $\sigma(\varphi) = \sigma(\psi)$ for all truth assignments σ.

As before we say that theories Π and Σ are *logically equivalent*, in symbols $\Pi \equiv \Sigma$, if they have the same models or if they are inter-derivable, ie. $\Pi \vdash \Sigma$ and $\Sigma \vdash \Pi$, where $\Psi \vdash \Phi$ means that $\Psi \vdash \varphi$ for all $\varphi \in \Phi$.

Viewing theories nonmonotonically within equilibrium logic, we distinguish different kinds of equivalence as follows. Let X stand for the restriction of expressions of the language to those formed from a given subset of $\mathcal{L}(\wedge, \vee, \rightarrow, \neg, \sim)$. For instance, $X = At = \mathcal{L}(\emptyset)$ denotes the set of atoms, $X = Lit = \mathcal{L}(\sim)$ the set of literals, $X = Boole = \mathcal{L}(\wedge, \vee, \neg, \sim)$ the collection of implication-free formulas and $X = Sent = \mathcal{L}(\wedge, \vee, \rightarrow, \neg, \sim)$ the collection of all sentences.

Definition 8 Two theories Π and Π' are said to be *equivalent*, in symbols $\Pi \approx \Pi'$ if they have the same equilibrium models. Two theories Π and Π' are said to be X-equivalent iff $\Pi \cup \Sigma \approx \Pi' \cup \Sigma$ for all $\Sigma \subseteq X$. We distinguish two special cases of X-equivalence. If there is no restriction on X, ie $X = \text{Sent}$, we say that Π and Π' are *strongly* equivalent. If $X = At$ we say that Π and Π' are *uniform* equivalent.

It is well-known from ASP and database theory that these notions are different, in particular equivalent programs need not be strongly equivalent and hence not intersubstitutable in all contexts. The concept of uniform equivalence is useful in contexts where the rules or complex formulas of a program or database are fixed and only the facts (atoms) are allowed to change. Again there are simple examples showing that theories may be uniform equivalent yet not strongly equivalent.

Let us mention some replacement properties. Given any formula φ containing a propositional variable p, we let $\varphi(p/\alpha)$ denote the result of replacing every occurrence of p in φ by the formula α. The following is straightforward.

Proposition 12 If α and β are fully equivalent then so are $\varphi(p/\alpha)$ and $\varphi(p/\beta)$.

From the property that N_5 is a deductive basis for equilibrium logic and answer set inference, it follows at once that logically equivalent theories must be strongly equivalent. Slightly less obvious is the fact proved in [54] that the converse property also holds.

Proposition 13 Π and Π' are strongly equivalent if and only if $\Pi \equiv \Pi'$.

So N_5 actually forms a strong deductive base. Moreover we know from Proposition 8 that N_5 is a maximal logic with this property. This provides a further strong argument for regarding equilibrium logic as a natural generalisation of answer set inference and for taking N_5 entailment as the meaning of $\mid\!\sim$ when there are no answer sets or equilibrium models. Actually the proof of Proposition 13 in [54] shows a little more, namely that theories are strongly equivalent if and only if they are X-equivalent where $X = \mathcal{L}(\rightarrow)$ i.e., comprising only sentences built up from atoms using the implication sign. Proposition 13 has recently been improved by De Jongh and Hendricks [45] who have identified for logic programs without strong negation the minimal intermediate logic that characterises strong equivalence, namely the logic of weak excluded middle which adds to Heyting's logic the axiom $\neg\neg\varphi \vee \neg\varphi$. This determines therefore the family of all super-intuitionistic logics that form strong deductive bases for answer set inference over logic programs.

Recently also uniform equivalence has been given a semantic characterisation, see [24, 84]. Given any theory Π let us say that a model $\mathcal{M}$ of Π is *maximal incomplete* if it is maximal under the ordering $\trianglelefteq$ among the *non-total* models of Π, i.e. if $\mathcal{M} \trianglelefteq \mathcal{M}'$, then $\mathcal{M} = \mathcal{M}'$ or else $\mathcal{M}'$ is total. It is clear that finite theories possess maximal incomplete models, but this is not guaranteed for infinite theories.

Proposition 14 Finite theories Π and Π' are uniform equivalent if and only if they have the same total and maximal incomplete models.

For infinite theories a similar but weaker property can be shown. Furthermore, [84] have shown that uniform and strong equivalence exhaust all kinds of X-equivalence.

Proposition 15 Two theories are uniform equivalent if and only if they are X-equivalent for $X = Boole$.

In short, strong equivalence amounts to $\mathcal{L}(\rightarrow)$-equivalence and reduces to logical equivalence, while $\mathcal{L}(\wedge, \vee, \neg, \sim)$-equivalence collapses to $\mathcal{L}(\emptyset)$ or uniform equivalence.

Proposition 12 states conditions under which propositional variables can be replaced without loss in single formulas. Propositions 8 and 13 show how logically equivalent formulas or sets of formulas can be inter-substituted without loss in the context of nonmonotonic entailment. Finally, Proposition 15 shows how replacement can occur in a restricted context.

To illustrate the use of these conditions in practice, consider first an example of disjunctive programs allowing negation in the head such as

$$\Pi_1 = \{r \rightarrow (\neg p \vee q)\}$$

It is readily seen that Π_1 is not strongly equivalent to

$$\Pi_2 = \{(r \wedge p) \rightarrow q\}$$

The interpretation $\langle \{r\}, \{p, q, r\} \rangle$ is a model of Π_2 but not of Π_1 (and is the only distinguishing model), while $\langle \{q, r\}, \{p, q, r\} \rangle$ is a model of Π_1. Therefore Π_1 and Π_2 do not differ on maximal incomplete models and are therefore uniform equivalent.

A distinguishing feature of equilibrium logic is that it allows us to express theories or programs with embedded implications, e.g., programs with conditional rules containing implication in the body, of the form $(p \rightarrow q) \rightarrow r$. It is interesting to consider when such rules can be replaced by ‘ordinary’ nested rules. Consider for instance:

$$\Pi_1 = \{((p \rightarrow q) \wedge s) \rightarrow r\}$$

versus

$$\Pi_2 = \{((\neg p \vee q) \wedge s) \rightarrow r\}$$

It is easy to see that these formulas are equivalent but not strongly equivalent and therefore not interchangeable in all contexts. In particular

$$\langle \{s\}, \{p, q, r, s\} \rangle \models \Pi_2,$$

but

$$\langle \{s\}, \{p, q, r, s\} \rangle \not\models \Pi_1.$$

This is the only interpretation that distinguishes the two formulas but it is not a maximal model of Π_2, since $\langle \{p, s\}, \{p, q, r, s\} \rangle$ is also a Π_2 model. Consequently the formulas are uniform equivalent. We may also consider the case where implication is permitted in the heads of rules. But one can easily see that in the simple case of $r \rightarrow (p \rightarrow q)$ there is strong equivalence with respect to the normal rule $(r \wedge p) \rightarrow q$.

Further types of equivalence have been studied in the context of ASP or equilibrium logic. A partial concept of strong equivalence, where the equivalence is restricted to a certain sub-vocabulary of the languages, has been defined and characterised for logic programs in [105], and a still more general concept of (solution) correspondence between programs is examined in [27]. A concept of strong equivalence called *synonymy*, suitable for the case where the languages of two theories or programs are completely distinct, is defined and studied in [85]. For programs with variables, versions of strong equivalence are studied in [56, 86] and in greater detail in [26] which includes also uniform equivalence and complexity analyses. Further study of replacement properties of formulas with strong negation and full equivalence can be found in [69].

Since all these different notions of equivalence specify contexts in which one theory or program may be replaced without loss by another, possibly 'simpler', theory or program, they may be of practical value in program simplification and optimisation. This idea is explored e.g. in [25, 70, 79].

5 Complexity, reductions and implementations

In this section, we look at the following issues. How complex is reasoning with (standard) equilibrium logic and how can we implement it in an automated reasoning system? Anticipating the results to be discussed below, the answer to the first question is that complexity is similar to that of disjunctive logic programs under answer set semantics. On the matter of implementation on the other hand there seem to be three viable approaches worth exploring: (a) direct implementation of a proof system for equilibrium logic, eg based on tableaux, sequent or other calculus; (b) reduction to second-order logic; or (c) reduction to ordinary disjunctive logic programs. Each of these methods has been developed at least partially in working prototype systems.

5.1 Tableaux systems

A proof theory for equilibrium logic can be constructed via semantic tableaux.[10] It is convenient to use the many-valued semantics as mentioned in Section 2.4. In the many-valued version of N_5, the ordering $\sigma_1 \trianglelefteq \sigma_2$ among models σ_1 and σ_2 of Π holds iff for every propositional variable p occurring in Π the following properties hold:

1. $\sigma_1(p) = 0$ if and only if $\sigma_2(p) = 0$.
2. If $\sigma_1(p) \geq 1$, then $\sigma_1(p) \leq \sigma_2(p)$.
3. If $\sigma_1(p) \leq -1$, then $\sigma_1(p) \geq \sigma_2(p)$.

This yields characterisations of total model and equilibrium model equivalent to the originals. Let $\Pi = \{\varphi_1, \ldots, \varphi_n\}$ be a set of formulas. Then a model σ of Π in N_5 is clearly a *total model* if $\sigma(p) \in \{-2, 0, 2\}$ for every propositional variable p in Π, that is, if it is a model of Π in classical logic with strong negation [99] which we denote as before by N_3.

[10]I assume the reader is familiar with the basic ideas of semantic tableaux; see in particular [41] for the treatment of multi-valued logics.

$$\frac{\{2\}{:}\varphi \to \psi}{\begin{array}{l|l|l}\{-2,-1,0\}{:}\varphi & \{2\}{:}\psi & \{-2,-1,0,1\}{:}\varphi \\ & & \{1,2\}{:}\psi\end{array}} \qquad \frac{\{-2,-1,0,1\}{:}\varphi \to \psi}{\begin{array}{c|c}\{1,2\}{:}\varphi & \{2\}{:}\varphi \\ \{-2,-1,0\}{:}\psi & \{-2,-1,0,1\}{:}\psi\end{array}}$$

$$\frac{\{1,2\}{:}\varphi \to \psi}{\{-2,-1,0\}{:}\varphi \mid \{1,2\}{:}\psi} \qquad \frac{\{-2,-1,0\}{:}\varphi \to \psi}{\begin{array}{c}\{1,2\}{:}\varphi \\ \{-2,-1,0\}{:}\psi\end{array}}$$

$$\frac{\{0,1,2\}{:}\varphi \to \psi}{\{-2,-1,0\}{:}\varphi \mid \{0,1,2\}{:}\psi} \qquad \frac{\{-2,-1\}{:}\varphi \to \psi}{\begin{array}{c}\{1,2\}{:}\varphi \\ \{-2,-1\}{:}\psi\end{array}}$$

$$\frac{\{-1,0,1,2\}{:}\varphi \to \psi}{\{-2,-1,0\}{:}\varphi \mid \{-1,0,1,2\}{:}\psi} \qquad \frac{\{-2\}{:}\varphi \to \psi}{\begin{array}{c}\{2\}{:}\varphi \\ \{-2\}{:}\psi\end{array}}$$

$$\frac{\{2\}{:}\neg\varphi}{\{-2,-1,0\}{:}\varphi} \qquad \frac{\{1,2\}{:}\neg\varphi}{\{-2,-1,0\}{:}\varphi} \qquad \frac{\{0,1,2\}{:}\neg\varphi}{\{-2,-1,0\}{:}\varphi} \qquad \frac{\{-1,0,1,2\}{:}\neg\varphi}{\{-2,-1,0,1\}{:}\varphi}$$

$$\frac{\{-2\}{:}\neg\varphi}{\{2\}{:}\varphi} \qquad \frac{\{-2,-1\}{:}\neg\varphi}{\{1,2\}{:}\varphi} \qquad \frac{\{-2,-1,0\}{:}\neg\varphi}{\{1,2\}{:}\varphi} \qquad \frac{\{-2,-1,0,1\}{:}\neg\varphi}{\{1,2\}{:}\varphi}$$

Figure 1 Tableau expansion rules in N_5 for $\to$ and $\neg$.

Standard methods for generating proof systems for multi-valued logics based on semantic tableaux can be found in [40, 41]. A tableau system for the logic of here-and-there is described in [2]. For its strong negation extension, N_5, a tableau system can be constructed using the following rules (for more details, see [80]). For the connectives $\to$ and $\neg$, the rules are given in (figure 1). Since the other three connectives, $\wedge$, $\vee$ and $\sim$ are what are known as *regular* connectives, standard expansion rules can be applied to them (see [41] for details).

The rules are for tableaux with *signed* formulas, where each formula is associated with values in a subset of 5. For example, consider the first rule for implication. We start with the formula $\varphi \to \psi$ labeled as true, i.e. with the sign $\{2\}$. This expression is expanded by three branches according to the three ways in which $\varphi \to \psi$ can take the value 2 (cf. the truth tables in “Many-valued semantics for N_5”), namely that φ takes a value in the set $\{-2, -1, 0\}$, that ψ takes the value 2, or that φ has a value in $\{-2, -1, 0, 1\}$ while ψ takes the value 2. We proceed similarly for each of the other signs or subsets of truth-values that $\varphi \to \psi$ can take and expand according to the truth-tables.

$$\frac{\{2\}{:}\varphi \to \psi}{\{-2,0\}{:}\varphi \mid \{2\}{:}\psi} \qquad \frac{\{-2,0\}{:}\varphi \to \psi}{\begin{array}{c}\{2\}{:}\varphi \\ \{-2,0\}{:}\psi\end{array}} \qquad \frac{\{0,2\}{:}\varphi \to \psi}{\{-2,0\}{:}\varphi \mid \{0,2\}{:}\psi} \qquad \frac{\{-2\}{:}\varphi \to \psi}{\begin{array}{c}\{2\}{:}\varphi \\ \{-2\}{:}\psi\end{array}}$$

$$\frac{\{2\}{:}\neg\varphi}{\{-2,0\}{:}\varphi} \qquad \frac{\{-2,0\}{:}\neg\varphi}{\{2\}{:}\varphi} \qquad \frac{\{0,2\}{:}\neg\varphi}{\{-2,0\}{:}\varphi} \qquad \frac{\{-2\}{:}\neg\varphi}{\{2\}{:}\varphi}$$

Figure 2 Tableau expansion rules in N_3 for the connectives $\to$ and $\neg$.

To study the validity of an inference $\varphi_1, \ldots, \varphi_n \models \psi$ in N_5 one begins with an initial tableau for

$$(\{\varphi_1, \ldots, \varphi_n\}, \psi): \qquad \begin{array}{c} \hline \{2\}{:}\varphi_1 \\ \ldots \\ \{2\}{:}\varphi_n \\ \{-2,-1,0,1\}{:}\psi \end{array}$$

As usual, a branch in a tableau is called *closed* if it contains a variable p with two signs, $S{:}p$, $S'{:}p$, such that $S \cap S' = \varnothing$; a non-closed branch is called *open*. A tableau is said to be *closed* if every branch is closed. The full tableau system for N_5 is complete in the sense that the entailment $\varphi_1, \ldots, \varphi_n \models \psi$ is valid if and only if there exists a closed tableau for $(\{\varphi_1, \ldots, \varphi_n\}, \psi)$ [80]. Notice that this system is therefore sufficient for checking the property of strong equivalence.

A similar tableau system for the logic N_3 can be used to check the property of being a total model of a set of sentences $\Pi = \{\varphi_1, \ldots, \varphi_n\}$. The initial tableau for Π is

$$\begin{array}{c} \hline \{2\}{:}\varphi_1 \\ \ldots \\ \{2\}{:}\varphi_n \end{array}$$

The tableau expansion rules are described in a standard way; in (figure 2) we again illustrate the rules for the connectives $\rightarrow$ and $\neg$.

A tableau T is called *terminated* if every branch is either closed or open. The total models for a set Π are generated from any terminated tableau, as described in the following:

Proposition 16 ([80]) Let T be a terminated tableau for $\Pi = \{\varphi_1, \ldots, \varphi_n\}$, and $\{S_1{:}p_1, \ldots, S_n{:}p_n\}$ be the set of signed literals in an open branch. Then every assignment σ verifying $\sigma(p_i) \in S_i$, for all i, is a total model of Π. Moreover, all the total models of Π are generated from T in this way.

A tableau system for computing equilibrium models can now be constructed via a generate and check mechanism. The system for N_3 generates the total models σ of a theory, while an auxiliary tableau system checks whether such a σ is minimal. A suitable auxiliary system is described in [80]. It is based on certain sublogics of N_5, using the fact that not all subsets of values in 5 are needed to verify minimality once a total model has been generated.

Based on these ideas a prototype system, `tabeql`, has been implemented at the University of Málaga [100]. The system can be used to generate equilibrium models and check equilibrium entailment. It also serves as a checker for strong equivalence and, by a suitable extension of the tableau system just mentioned, also for uniform equivalence.

5.2 Second order reduction

We consider first the language without strong negation and introduce the following notation. If V is a set of propositional variables or atoms, we denote by V' the disjoint alphabet $V' = \{p' : p \in V\}$. For any formula φ with variables from V, let φ' be the result of replacing each variable $p \in V$ by p'. If $V = \{p_1, \ldots, p_n\}$ and

$U = \{q_1, \ldots, q_n\}$ are indexed sets of atoms, then we write $V \leq U$ as an abbreviation for $\bigwedge_{i=1}^{n}(p_i \rightarrow q_i)$, and $V < U$ as an abbreviation for $(V \leq U) \& \neg(U \leq V)$. Lastly, for any formula φ with variables from V we define a translation φ^* recursively as follows.

1. If φ is an atom or $\bot$, then $\varphi^* = \varphi$;
2. if $\varphi = (\varphi_1 \circ \varphi_2)$, for $\circ \in \{\wedge, \vee\}$, then $\varphi^* = {\varphi_1}^* \circ {\varphi_2}^*$;
3. if $\varphi = (\varphi_1 \rightarrow \varphi_2)$, then $\varphi^* = ({\varphi_1}^* \rightarrow {\varphi_2}^*) \wedge (\varphi_1' \rightarrow \varphi_2')$.

Then the following characterisations of HT and equilibrium models can be obtained, [82].

Proposition 17 Let φ be a formula with atoms in V and let $H, T \subseteq V$ be interpretations. Then, $\langle H, T \rangle$ is an HT-model of φ iff $H \cup T'$ is a (classical) model of $(V \leq V') \wedge \varphi^*$.

Intuitively, the primed formulas in φ^* play the role of formulas evaluated in the 'there' world and unprimed formulas correspond to those evaluated 'here', while the condition $V \leq V'$ expresses the requirement that truth persists from 'here' to 'there'.

Proposition 18 Let φ be a formula with atoms in V. Then, $\langle T, T \rangle$ is an equilibrium model of φ iff T' is a model of

$$\varphi' \wedge \neg\exists V\big((V < V') \wedge \varphi^*\big). \tag{7}$$

A formula of type (7) is a special kind of second-order expression known as a *quantified boolean formula* or QBF. QBFs generalise ordinary propositional formulas by allowing quantification over propositional variables. Informally, a QBF of the form $\forall p\, \exists q\, \Phi$ means that for all truth assignments of p there is a truth assignment of q such that Φ is true. For a full account of what it means for a propositional interpretation to be a (classical) model of a QBF, see e.g. [47]. Similar encodings by QBFs can be formulated for the special cases of disjunctive or nested logic programs. Moreover QBFs can be used to express properties such as whether a theory has an equilibrium model or whether a formula is an equilibrium consequence of a given theory.

This kind of reduction has two main applications. First, since efficient QBF-solvers are readily available, these encodings can be used as a basis for implementing equilibrium logic. A prototype system has been developed at the Vienna University of Technology, [104]. Secondly, much is known about the complexity classes associated with QBFs of different kinds; therefore QBF-reductions often provide useful information about the complexity of the various reasoning tasks being encoded. For each of the relevant tasks upper complexity bounds can be obtained by the structure of the corresponding encoding scheme. In fact since in this case the translations are polynomial in the size of the given underlying problem instance, well-known results about the complexity of evaluating these formulas yield straightforward membership results.

For the basic concepts of complexity theory see, e.g., [72]. The following table summarises the main results of analysing the quantifier order of the different QBF encodings. Each row associates a complexity class for a decision problem wrt to (disjunctive) logic programs, nested logic programs and propositional theories in the logic of here-and-there, respectively. In each case the decision problem is *complete* for the class in question. From top to bottom the decision problems are: existence of an HT-model, existence of an equilibrium model, whether a formula is an equilibrium consequence of a theory or program, and, in turn, the problems of checking equivalence, uniform equivalence and strong equivalence.[11]

	LPs	NLPs	Theories
Model existence	NP	NP	NP
Equil-model existence	Σ_2^P	Σ_2^P	Σ_2^P
Equil-consequence	Π_2^P	Π_2^P	Π_2^P
Equivalence	Π_2^P	Π_2^P	Π_2^P
u-equivalence	Π_2^P	Π_2^P	Π_2^P
s-equivalence	coNP	coNP	coNP

Two features of this table stand out. First, it is evident that strong equivalence, which amounts to checking equivalence in HT or N_5, is computationally more tractable than checking equivalence or uniform equivalence. Second, the complexity classes for theories and logic programs are always the same. Consequently we might expect to find polynomial reductions of theories and nested programs to disjunctive programs. We now turn to this topic in more detail.

5.3 Reductions to logic programs

Evidently. if there is an efficient means to transform theories in equilibrium logic into ordinary logic programs, then standard ASP solvers can be used as the main engine to compute equilibrium models and implement associated reasoning tasks.

Quite independent of equilibrium logic, a substantial part of a transformation was already described in [55]. What this work showed is how a nested program with rules of kind (6) can be converted into an equivalent (in fact strongly equivalent) program with rules of sort (5) and subsequently into an ordinary disjunctive program.

For simplicity let us treat the language without strong negation. Then in a nested rule of form

$$\alpha \rightarrow \beta \tag{8}$$

α, β are boolean expressions in $\mathcal{L}(\wedge, \vee, \neg)$ and we can consider HT as our underlying logic. For the time being let us regard literals as comprising atoms and weakly negated atoms. Since the familiar de Morgan laws are theorems of HT, we can actually drive negation into a formula so that it stands directly in front of a literal (let us say that such a formula is then in negation normal form, or nnf for short). At the same time, multiple odd-length sequences of '$\neg$' are reduced by virtue of $(\neg\neg\neg\varphi \leftrightarrow \neg\varphi)$ being a theorem of intuitionistic logic and its extensions. Without

[11]The main references areas follows. For the complexity of satisfiability in many-valued logics such as HT, see [64]. For the complexity of reasoning tasks associated with disjunctive logic programs, see [28]. For SE of logic programs, see [82] and also independently some results of [56] and [97]. For uniform equivalence of logic programs, see Eiter and Fink [24]. For other results and the full details of the reduction method sketch here, see [82].

loss of generality we can also assume that in the expression (8), the formula α is a conjunction and the formula β is a disjunction. The complex formulas on the left and right side of the implication are reduced via the following equivalences.
Left side rules

$$\neg\neg\varphi \wedge \alpha \rightarrow \beta \;\Leftrightarrow\; \{\, \alpha \rightarrow \neg\varphi \vee \beta \,\} \tag{9}$$

$$(\varphi \vee \psi) \wedge \alpha \rightarrow \beta \;\Leftrightarrow\; \begin{Bmatrix} \varphi \wedge \alpha \rightarrow \beta \\ \psi \wedge \alpha \rightarrow \beta \end{Bmatrix} \tag{10}$$

Right side rules

$$\alpha \rightarrow \neg\neg\varphi \vee \beta \;\Leftrightarrow\; \{\, \neg\varphi \wedge \alpha \rightarrow \beta \,\} \tag{11}$$

$$\alpha \rightarrow (\varphi \wedge \psi) \vee \beta \;\Leftrightarrow\; \begin{Bmatrix} \alpha \rightarrow \varphi \vee \beta \\ \alpha \rightarrow \psi \vee \beta \end{Bmatrix} \tag{12}$$

Although the authors of [55] verified these reduction rules purely in terms of the conventional definition of answer sets, it is easy to check that each corresponds to a valid logical equivalence in HT, so clearly they convert rules into strongly equivalent sets of rules. Notice that by applying these rules we obtain sets of formulas corresponding to generalised program rules of form (5).

To obtain a full reduction of arbitrary propositions into sets of formulas corresponding to (5), we proceed as follows. To obtain formulas in nnf, we add the equivalence $\neg(\varphi \rightarrow \psi) \Leftrightarrow \neg\neg\varphi \wedge \neg\psi$. To remove nested implications we add two more rules, for the left and right sides, respectively.

$$(\varphi \rightarrow \psi) \wedge \alpha \rightarrow \beta \;\Leftrightarrow\; \begin{Bmatrix} \neg\varphi \wedge \alpha \rightarrow \beta \\ \psi \wedge \alpha \rightarrow \beta \\ \alpha \rightarrow \varphi \vee \neg\psi \vee \beta \end{Bmatrix} \tag{13}$$

$$\alpha \rightarrow (\varphi \rightarrow \psi) \vee \beta \;\Leftrightarrow\; \begin{Bmatrix} \varphi \wedge \alpha \rightarrow \psi \vee \beta \\ \neg\psi \wedge \alpha \rightarrow \neg\varphi \vee \beta \end{Bmatrix} \tag{14}$$

Again it is easy to prove logical equivalence in HT; a more detailed account can be found in [13].

The above reduction is not in general polynomial. In fact it can be shown that there is no polynomial time reduction of arbitrary theories to disjunctive programs in the same vocabulary [13]. To obtain a polynomial reduction, as suggested by the complexity results, we must therefore consider transformations that do not preserve precisely the vocabulary of the original theory. Nevertheless, we want the transformed theory to be in a strong sense equivalent to the original and such that the equilibrium models of the original can be retrieved from it.

As in the case of similar reductions in classical logic, this can be achieved by introducing labels (new atoms) for each (non-constant) formula in the original language; say L_φ is the label corresponding to the formula φ. Let $\mathcal{L}_V$ be the language of HT over a signature V. For any signature $U \supseteq V$ and HT model $\mathcal{M}$ for $\mathcal{L}_U$, we de note by $\mathcal{M}\restriction_V$ the restriction of $\mathcal{M}$ to atoms in V. For any theory Π, let $subf(\Pi)$ denote the set of all subformulas of Π. The translations we consider use a signature V_L that contains an atom (a label) for each non-constant formula in the original language $\mathcal{L}_V$, that is:

$$V_L = \{L_\varphi \mid \varphi \in \mathcal{L}_V\}$$

It is convenient to set $L_\varphi \overset{\text{def}}{=} \varphi$ when φ is an atom $p \in V$ so that we can treat V_L as a superset of V.

For any non-atomic formula $\varphi \bullet \psi$ built with a binary connective $\bullet$, we call its *definition*, $df(\varphi \bullet \psi)$, the formula:

$$L_{\varphi\bullet\psi} \leftrightarrow L_\varphi \bullet L_\psi$$

Similarly $df(\neg\varphi)$ represents the formula $L_{\neg\varphi} \leftrightarrow \neg L_\varphi$. Now, given any theory Π, we can define a translation $\sigma(\Pi)$ to comprise the labels for all formulas in Π plus the definitions for all subformulas in Π; i.e., we set

$$\sigma(\Pi) = \{L_\varphi \mid \varphi \in \Pi\} \cup \bigcup_{\gamma \in sub\,f(\Pi)} df(\gamma) \tag{15}$$

It is straightforward to show that the equilibrium models of a theory Π correspond precisely to the restrictions of the equilibrium models of $\sigma(\Pi)$ to the vocabulary of Π. In fact we have more generally

Proposition 19 Let Π be a theory in $\mathcal{L}_V$ and σ the translation defined above. The restriction functor $\lceil_V$ determines a one-one correspondence between the models of $\sigma(\Pi)$ and the models of Π.

Since the correspondence is preserved when Π is enlarged by the addition of new formulas in the same language, the equivalence of Π and $\sigma(\Pi)$ holds in a strong sense (the translation is sometimes said to be *faithful*), see [13].

However, $\sigma(\Pi)$ does not have the shape of a logic program: it contains double implications where the implication symbol may occur nested. These can be unfolded in linear time without changing the signature V_L. For each definition $df(\gamma)$, we define the strongly equivalent set (understood as the conjunction) of logic program rules $\pi(\gamma)$ as shown in (figure 3). The fact $df(\gamma) \equiv_s \pi(\gamma)$ can be easily checked in HT. For the case of nested programs without embedded implications a similar procedure is described in [83]. The set of rules for implication, $\pi(\varphi \to \psi)$, is used in [71] to unfold nested implications in an arbitrary theory.

γ	$df(\gamma)$	$\pi(\gamma)$	γ	$df(\gamma)$	$\pi(\gamma)$
$\varphi\wedge\psi$	$\mathbf{L}_{\varphi\wedge\psi} \leftrightarrow \mathbf{L}_\varphi \wedge \mathbf{L}_\psi$	$\mathbf{L}_{\varphi\wedge\psi} \to \mathbf{L}_\varphi$ $\mathbf{L}_{\varphi\wedge\psi} \to \mathbf{L}_\psi$ $\mathbf{L}_\varphi \wedge \mathbf{L}_\psi \to \mathbf{L}_{\varphi\wedge\psi}$	$\neg\varphi$	$\mathbf{L}_{\neg\varphi} \leftrightarrow \neg\mathbf{L}_\varphi$	$\neg\mathbf{L}_\varphi \to \mathbf{L}_{\neg\varphi}$ $\mathbf{L}_{\neg\varphi} \to \neg\mathbf{L}_\varphi$
$\varphi\vee\psi$	$\mathbf{L}_{\varphi\vee\psi} \leftrightarrow \mathbf{L}_\varphi \vee \mathbf{L}_\psi$	$\mathbf{L}_\varphi \to \mathbf{L}_{\varphi\vee\psi}$ $\mathbf{L}_\psi \to \mathbf{L}_{\varphi\vee\psi}$ $\mathbf{L}_{\varphi\vee\psi} \to \mathbf{L}_\varphi \vee \mathbf{L}_\psi$	$\varphi\to\psi$	$\mathbf{L}_{\varphi\to\psi} \leftrightarrow (\mathbf{L}_\varphi \to \mathbf{L}_\psi)$	$\mathbf{L}_{\varphi\to\psi} \wedge \mathbf{L}_\varphi \to \mathbf{L}_\psi$ $\neg\mathbf{L}_\varphi \to \mathbf{L}_{\varphi\to\psi}$ $\mathbf{L}_\psi \to \mathbf{L}_{\varphi\to\psi}$ $\mathbf{L}_\varphi \vee \neg\mathbf{L}_\psi \vee \mathbf{L}_{\varphi\to\psi}$

Figure 3 Transformation $\pi(\gamma)$ generating a generalised disjunctive logic program.

Lastly we note that occurrences of default negation in the heads of program rules can be eliminated by a well-known technique of [46] or by a direct modification of $\pi(\Pi)$ to yielding a disjunctive program [13].

A reduction of the above kind was described for nested programs in [83] and on this basis a system called `nlp` [92] for compiling nested logic programs has been made available as a front-end to `DLV`. A full account of the above reduction with proofs of equivalence and complexity can be found in [13].

6 Mutations and extensions

The basic equilibrium construction lends itself to natural variations, generalisations and extensions obtained by changing the underlying logic, modifying the concept of equilibrium model or enriching the language say by adding modal operators or quantifiers. Let us look at some of these variations in turn.

6.1 Some variants

Consider the propositional language without strong negation. Then equilibrium logic is based on the logic of here-and-there, also known as Gödel's three-valued logic. Let us use the values 1 for true, 0 for false and 1/2 for the third value. If we examine the equilibrium construction in terms of these three truth-values, it is evidently based on the following idea: first fix the set of false atoms assigned the value 0 (equivalently fix the atoms true 'there'), then minimise the set of true atoms assigned 1 (i.e., those verified 'here'); equilibrium is reached if the resulting set of true atoms is the complement of the set of false atoms, ie no atoms are assigned 1/2. Clearly this construction can be applied to any three-valued logic (and with certain provisos extended to n-valued logics). Do different equilibrium logics result? It is straightforward to verify that the answer is positive. Consider again the theory Π_5 from Section 2.3:

$$\neg\neg p \rightarrow p. \tag{16}$$

As we saw, in standard equilibrium logic based on here-and-there, this formula has two equilibrium models; $\langle\{\},\{\}\rangle$ and $\langle\{p\},\{p\}\rangle$. So the strongest formula entailed by this theory is just $p \vee \neg p$. On the other hand, consider the same equilibrium construction applied to Łukasiewicz three-valued logic. It is easily checked that the only equilibrium model is the model assigning p the value 0, i.e., that which corresponds to the first of the two here-and-there models. The reason is that in Łukasiewicz logic, the interpretation that assigns p the value 1/2 is also a model of the formula, so the model assigning p the value 1 is not in equilibrium. Consequently, for this theory the Łukasiewicz version of equilibrium logic is stronger, yielding the consequence $\neg p$. Notice that a similar effect in this case would be obtained for standard equilibrium logic if we were to choose not all equilibrium models. but rather only the minimal ones. Alternatively, consider the situation in terms of theory-completions as in Many-valued semantics for N_5. The formula (16) has two completions, corresponding to the addition of $\neg p$ or of $\neg\neg p$. In the former case we add the negation of an atom, in the latter case the negation of a complex formula. The fact that the first of these corresponds to choosing the minimal equilibrium model is part of a general pattern. Let Π be a theory. Let us say that a set E of sentences extending Π is an *atomic completion* of Π iff $E = Cn(\Pi \cup \{\neg a : a \in At \;\; \& \;\; a \notin E\})$.

Proposition 20 The atomic completions of a theory Π correspond to the minimal equilibrium models of Π.

Proof Left to the reader.

In the case where Π is a disjunctive logic program, all equilibrium models are 'minimal' in the above sense and so completions and atomic completions coincide [78]. Proposition 14 seems to be of purely technical interest since we know of no theoretical motivation for choosing the stronger, 'minimal model' variant for practical reasoning purposes. On the other hand, the implications of working with the Łukasiewicz or other similar variants of equilibrium logic remain to be systematically explored.

6.2 Paraconsistent answer sets and partial equilibrium logic

There are other ways to modify equilibrium logic by changing the underlying monotonic base. Nelson's constructive logic N has also been extensively studied in a paraconsistent version, sometimes called N^{-}, that omits the axiom $\varphi \rightarrow (\sim\varphi \rightarrow \psi)$. In paraconsistent versions of N, the 'weak' negation $\neg\varphi$ is no longer definable, but can be introduced as an abbreviation for $\varphi \rightarrow \bot$ where $\bot$ is a constant interpreted as false in each world. Extending this version of the logic N^{-} by adding the Łukasiewicz axiom $(\neg\alpha \rightarrow \beta) \rightarrow (((\beta \rightarrow \alpha) \rightarrow \beta) \rightarrow \beta)$ for here-and-there results in a nine-valued logic that we can denote by N_9. By defining a suitable equilibrium construction for N_9-models we can capture precisely the paraconsistent version of answer sets for disjunctive programs, as studied in [90]. The method is described in detail in [66].

A somewhat more radical departure is obtained by changing the underlying models. For the language without strong negation, we can consider here-and-there models equipped with an operation $*$ such that the usual worlds h and t have companion worlds h^* and t^* accessible from the originals; moreover just as $h \leq t$ we have $h^* \leq t^*$. By defining a suitable ordering and minimality condition on these extended here-and-there models, one can obtain partial versions of equilibrium models that correspond to the partial stable models of a logic program as defined in [88]. It is natural to call the resulting logic *partial equilibrium logic*. The underlying monotonic base logic in this case is six-valued and belongs to a family of logics with weak negation studied some time ago by Došen [21]. It was only recently axiomatised, [8, 9]. In partial equilibrium logic, the well-founded model for normal logic programs can be represented as a minimal model in the ordinary logical sense. Properties of partial equilibrium logic, including strong equivalence theorems, have been recently explored in [8–12].

6.3 Fixpoint logics

Consider once again for simplicity the language without strong negation. The characterisation of equilibrium logic in terms of theory-completions (Section 2.4) is an obvious candidate for generalisation, simply by changing the consequence relation. In other words, let us say that E is an L-completion of Π if $E = \mathrm{Cn}_{\mathrm{L}}(\Pi \cup \{\neg\varphi : \varphi \notin E\})$, where Cn_{L} is the consequence operator of a superintuitionistic logic L. We may expect to obtain different nonmonotonic logics by considering the consequences of all L-completions of a theory, for varying L. Evidently if we set L equal to classical logic we just obtain monotonic classical logic, since every classical model corresponds to a complete theory. With L equal to here-and-there, we obtain equilibrium logic for the language without strong negation. However, somewhat surprisingly it turns out that no other nonmonotonic logics are obtained by varying L. Mauricio Osorio and his group in Puebla have shown that for any L between intuitionistic logic and here-and-there, the L-completions coincide.

6.4 Safe beliefs

The group of Mauricio Osorio has studied the fixpoint characterisation of Section 2.5 at length and in several papers proposed an alternative system of nonmonotonic reasoning called *safe beliefs*; see e.g., [71]. It consists in a small change to Definition 4. For a set of atoms X, denote by $\overline{X}$ the complement of X, set $\neg X = \{\neg a : a \in X\}$ and similarly $\neg\neg X = \{\neg\neg a : a \in X\}$. Let Π be a theory and M a set of atoms, then M is said to be a set of safe beliefs (wrt Π) if (a) $\Pi \cup \neg\neg M \cup \neg\overline{M}$ is consistent and (b) $\Pi \cup \neg\neg M \cup \neg\overline{M} \vdash M$. $\vdash$ can be taken as the inference relation of here-and-there (or N_5 if strong negation is present) and it is assumed that atoms not in M are false. It is straightforward to check

Proposition 21 For any theory Π, M is a set of safe beliefs for Π iff it is an equilibrium model of Π.

Proof Suppose that $\Pi \cup \neg\overline{M} \cup \neg\neg M \models M$ where $\Pi \cup \neg\overline{M} \cup \neg\neg M$ is consistent. By consistency $\Pi \cup \neg\overline{M} \cup \neg\neg M$ has a here-and-there model say $\langle H, T\rangle$. Recall that H comprises the true atoms in the model and the complement of T, $\overline{T}$, defines the false atoms and we always have $H \subseteq T$. So since M is true in all models of $\Pi \cup \neg\overline{M} \cup \neg\neg M$, we must have $M \subseteq H$. Similarly, since $\langle H, T\rangle \models \neg\overline{M}$ we must have $\overline{M} \subseteq \overline{T}$ and hence $T \subseteq M$. So clearly $H = T = M$ so $\langle M, M\rangle$ (or just M) is a model of Π. Suppose that M is not in equilibrium. Then by the definition of equilibrium model, there is a model $\langle H, T\rangle$ of Π such that $T = M$ but H is a proper subset of M. Now since $T = M$ it is clear that $\langle H, T\rangle \models \neg\neg M$ by the here-and-there semantics. And likewise every atom a not in M is false in $\langle H, T\rangle$, so $\langle H, T\rangle \models \neg\overline{M}$. But then $\langle H, T\rangle \models \Pi \cup \neg\overline{M} \cup \neg\neg M$. Since H is a proper subset of M, this contradicts the assumption that M is true in every model of $\Pi \cup \neg\overline{M} \cup \neg\neg M$.

Conversely, let us suppose that M is an equilibrium model of Π. Then by the semantics, we have that $\langle M, M\rangle \models \Pi$, $\langle M, M\rangle \models \neg\overline{M}$ and $\langle M, M\rangle \models \neg\neg M$ so $\langle M, M\rangle \models \Pi \cup \neg\overline{M} \cup \neg\neg M$. It is easy to see that $\Pi \cup \neg\overline{M} \cup \neg\neg M$ has no other model than M. For suppose it had another model $\langle H, T\rangle$. Then since $\langle H, T\rangle \models \neg\overline{M}$ we must have $\overline{M} \subseteq \overline{T}$ and therefore $T \subseteq M$. And since $a \in M$ implies $\langle H, T\rangle \models \neg\neg a$, we have $M \subseteq T$. So $T = M$. But since M is in equilibrium by definition also $H = M$, so M is the only model of $\Pi \cup \neg\overline{M} \cup \neg\neg M$ and trivially M is true in all models of $\Pi \cup \neg\overline{M} \cup \neg\neg M$, ie $\Pi \cup \neg\overline{M} \cup \neg\neg M \models M$ which is what we wanted to check. □

Consequently, both the system of safe beliefs as well as the versions of fixpoint logic described in “Fixpoint logics” above constitute only notational variants of equilibrium logic.

6.5 Modal extensions

The basic formalism of ASP, as exemplified by disjunctive logic programs, has been augmented in various ways by the addition of such items as modal operators for knowledge and belief, and causal or other operators for describing the results of actions (see e.g., [6]). An early extension of the former kind is that of *epistemic specifications*, studied by Gelfond [32], but not further developed into an implemented system. In this formalism, each literal in a disjunctive rule of the form (4) may be prefixed by a knowledge operator ‘K’. Collections of such rules are interpreted not via sets of literals, as in the case of answer sets, but by families of sets of literals, called *world views*. These are defined, as in the non-epistemic case, by means of reduct operators.

Recently Wang and Zhang [103] have described an epistemic version of equilibrium logic that precisely captures reasoning with epistemic specifications and permits generalisation to arbitrary theories (they specifically look at nested programs). When strong negation is absent, the underlying syntax is that of here-and-there logic, with the additional formation rule: if φ is a formula so is $K\varphi$. Interpretation is via *epistemic* here-and-there models, $\langle \mathcal{A}, H, T\rangle$, where as before H and T are sets of atoms with $H \subseteq T$, and $\mathcal{A}$ is a collection of sets of atoms (there is no requirement that $H, T \in \mathcal{A}$). The crucial semantic clause is that $\langle \mathcal{A}, H, T\rangle$ verifies $K\varphi$ at world w, if for all $H', T' \in \mathcal{A}$ with $H' \subseteq T'$, $\langle H', T'\rangle, w \models \varphi$ in the usual sense of here-and-there logic. Equilibrium models for an epistemic theory Φ are ‘total’ models $\langle \mathcal{A}, T, T\rangle$ of Φ such that there is no model $\langle \mathcal{A}, H, T\rangle$ of Φ with $H \subset T$. Lastly, Gelfond’s world views defining the semantics of epistemic programs Φ correspond to maximal collections $\mathcal{A}$ of sets of atoms satisfying $\mathcal{A} = \{I : \langle \mathcal{A}, I, I\rangle \text{ is an equilibrium model of } \Phi\}$. Wang and Zhang call these maximal collections *equilibrium views*.

The work of [103] can be considered a first step towards an epistemic equilibrium logic. It remains to be seen how the underlying epistemic version of N_5 can be axiomatised and how appropriate proof systems can be built.

6.6 Quantified equilibrium logic

First-order versions of the logic HT of here-and-there and its strong negation extension, N_5, have received little attention in the literature. If we regard N_5 as a many-valued logic and add quantifiers using the standard approach, we obtain the following semantics: an interpretation is a pair $\langle D, \sigma\rangle$ where $\mathcal{C}$ is the set of constants of the language, $D \supseteq \mathcal{C}$ is a non-empty *domain*, and $\sigma :$ At $\to 5$ is the *assignment*. If $\mathcal{T}$ is the set of ground terms that can be built from predicate, constant and function symbols of the language, the notion of model is extended to quantified sentences as follows:

$$\sigma(\forall x\varphi(x)) = \min\{\sigma(\varphi(t)); t \in \mathcal{T}\} \qquad \sigma(\exists x\varphi(x)) = \max\{\sigma(\varphi(t)); t \in \mathcal{T}\}$$

It is straightforward to show that in this semantics the formulas $\forall x(P(x) \vee Q(a))$ and $\forall x P(x) \vee Q(a)$ are equivalent. Consequently the resulting logic corresponds to that of Kripke models with *constant domains*, i.e., in a first-order here-and-there Kripke model the worlds ‘here’ and ‘there’ are assigned the same domain. We can denote this logic by QN_5^c. For the version without strong negation, axioms and a completeness proof have been provided by Ono [68], who also proves that the logic has the Interpolation Property. Ono’s methods and interpolation result can easily be shown to extend to QN_5^c, [86]. Completeness of a quantified version of N_5 without the constant domain assumption is provided in [86].

There are clearly different ways to specify an equilibrium condition for first-order N_5 depending on which quantified version of the logic is chosen and what type of minimality condition is preferred. If we are guided by the desideratum that first-order equilibrium should dovetail with the semantics for answer set programs with variables, then the logic QN_5^c seems to be an adequate choice. The natural equilibrium condition is then a straightforward adaptation of the propositional case: equilibrium models are total Kripke models of a theory such that no other model in the same domain verifies the same non-false literals and fewer true literals. Expressed in terms of the many-valued semantics this condition amounts to the following. An assignment σ is total if $\sigma(L) \in \{-2, 0, 2\}$ for all ground literals L; the ordering $\sigma_1 \trianglelefteq \sigma_2$ holds iff for every literal L in the language the following properties hold:

1. $\sigma_1(L) = 0$ iff $\sigma_2(L) = 0$.
2. If $\sigma_1(L) \geq 1$, then $\sigma_1(L) \leq \sigma_2(L)$
3. If $\sigma_1(L) \leq -1$, then $\sigma_1(L) \geq \sigma_2(L)$

Equilibrium models are then total models of a theory that are minimal under $\trianglelefteq$. As far as ASP is concerned the adequacy of this equilibrium concept can be seen as follows. If Π is a logic program with variables, its answer sets are identified with the answer sets of the grounding of Π with respect to the Herbrand universe of Π. Let Π be a theory in the signature comprising the set of constants $\mathcal{C}$, the set of functions $\mathcal{F}$ and the set of predicates $\mathcal{P}$. A model of Π can be represented in the form $\mathcal{M} = \langle \mathcal{D}, H, T \rangle$, where $\mathcal{D}$ is the domain of $\mathcal{M}$ and H, T are the sets of atoms verified at the worlds ‘h’ and ‘t’ respectively. As usual an Herbrand model of Π is a model in $QN_5^c(\mathcal{C}, \mathcal{F}, \mathcal{P})$ with domain $\mathcal{H} = \mathcal{T}(\mathcal{C}, \mathcal{F})$, called the *Herbrand universe* of Π.

Regarding a program Π with variables as a first-order theory in QN_5^c, we can consider the total Herbrand models of the universal closure of Π.

Proposition 22 Let Π be an open logic program with Herbrand universe $\mathcal{H}$. A total Herbrand model $\langle \mathcal{H}, T, T \rangle$ of the universal closure of Π is an equilibrium model of Π iff T is an answer set of Π.

A precise statement and proof of this fact can be found in [86].

Quantified equilibrium logic seems to be a useful direction to explore that may be of some benefit for future generation ASP implementations. In particular it may offer ways of transforming and simplifying programs, or even provide partial answer set computation, prior to the process of grounding.

7 Methodological discussion

Before concluding, let us consider some other examples that illustrate the utility of equilibrium logic and its underlying methodology.

7.1 Strong negation

In [34, 35] answer sets were defined for logic programs that include a second negation operator '$\sim$' representing direct and explicit falsity. It was labeled *classical* negation to distinguish it from the weaker negation-as-failure or negation by default, *not*. It was then shown how '$\sim$' could be eliminated by adding new predicates to the language. There are two steps to this 'reduction': (a) replace each occurrence of a negated atom $\sim p$ by a new atom, say p'; (b) add the integrity constraint $\leftarrow p, p'$. Starting from a program Π containing strong negation, one arrives at a transformed program, say $\Pi' \cup S$, free of strong negation, where Π' is the set of transformed rules and S the set of corresponding integrity constraints. Gelfond and Lifschitz [34, 35] showed that, modulo a consistency requirement, the answer sets of Π correspond precisely to the stable models of $\Pi' \cup S$. Any implementation of stable models can therefore be easily extended to answer sets. Subsequently, when answer set semantics was extended to cover disjunctive programs, or more recently programs with nested expressions, it was shown that this 'reduction' property continues to hold. Each time a fresh proof was needed because the basic definition of the semantics had been expanded. It seems fair to ask, why does this property seem to hold for every (reasonable) extension of answer set semantics, while it doesn't hold for rival semantics of programs with explicit negation, such as WFSX, [87]? Is it just by chance, or is it the result of some more basic systematic property of $\sim$?

We have already seen that the second answer is the correct one. What Gelfond and Lifschitz had inadvertently added to stable model semantics was not classical negation, which displays different properties and behaviour, but rather *strong* negation. Strong negation '$\sim$' under the Vorob'ev axioms ($N_1 - N_6$) has the following property: add it and just these axioms to a superintuitionistic logic and one obtains a conservative extension of the system; so strong negation is highly robust, the least extension by strong negation does not change the theorems of the original system, it only adds new theorems involving the new operator. Perhaps the most striking evidence for the fact that '$\sim$' is not classical negation is that one can add it to the superintuitionistic *classical* logic to obtain a three-valued logic equivalent to that of Łukasiewicz, see Vakarelov [99].

We saw already in condition (3) of Section 2 that Gelfond and Lifschitz's elimination of strong negation under answer set semantics is part of a more general pattern of life. The Gurevich reduction (3) implies that for any theory Π there is an exact correspondence between the N_5-models of Π and the HT-models of $\Pi' \cup S$ and therefore the minimal N_5-models or answer sets of Π correspond to the minimal models or stable models of $\Pi' \cup S$. So the reduction holds in any extension of answer set semantics that continues to extend Nelson's logic. Since equilibrium logic is a form of minimal model reasoning in an extension of N, any such properties of N holding in all models continue to hold in the intended minimal models: one doesn't need to re-examine the actual definition of answer set or minimal model.

7.2 Contrapositives

The subject of contrapositive reasoning in logic programming has been fraught with controversy and misunderstandings. For a long time it was accepted wisdom in logic programming to regard program formulas or rules as having a unique direction and not to obey any form of contraposition. The clearest evidence came from simple formulas such as

$$a \leftarrow \neg b \tag{17}$$

that behave differently from their 'contrapositives'

$$b \leftarrow \neg a. \tag{18}$$

The first has $\{a\}$ as a stable model, the second has $\{b\}$. So it seems that $\leftarrow$ is directional, more like an inference rule, and contraposition in general fails. The case of strong negation is apparently similar. While

$$a \leftarrow \sim b \tag{19}$$

and

$$b \leftarrow \sim a \tag{20}$$

have the same (empty) answer set, as soon as we add say $\sim b$ to each, their behaviour is quite different: a becomes derivable in the first case and not in the second. One might even think that the failure of contraposition is a result of the special properties of $\leftarrow$ making it different from ordinary logical implication. But nothing could be further from the truth. First there is nothing special about $\leftarrow$ as an implication connective providing one observes that its underlying logic is nonclassical. Second, the behaviours of $\neg$ and $\sim$ are actually quite different. While the former is perfectly 'contrapositive', the latter is not. The source of the confusion and common misunderstanding stems from wrongly interpreting (18) as the contrapositive of (17) by supposing that the underlying logic is classical. Providing weak negation $\neg$ is understood as intuitionistic, it is clear that the contrapositive form of (17) is actually

$$\neg\neg b \leftarrow \neg a \tag{21}$$

which is neither intuitionistically nor HT-equivalent to (18). Actually ordinary logic programming does not admit formulas of the form (21) and so cannot express all contrapositive forms. But in many cases it is useful and even necessary to do so. As we have seen, in equilibrium logic there is no problem in representing (21) and proving that it is equivalent to (17) in all contexts. We need no special properties of answer sets, merely the fact that equilibrium logic is a form of minimal model reasoning in an extension of intutionistic logic. Alternatively, (21) can be regarded as a rule with nested expressions as in [55], and equivalence with (17) can be proved using properties of the answer sets of such programs. So weak negation '$\neg$' is contrapositive but (17) and (18) are non-equivalent because $\neg$ does not obey the law of double-negation. In the case of strong negation exactly the converse holds: '$\sim$' obeys the double-negation law but fails contraposition. (20) is a contrapositive of (19) but not equivalent to it. This has to do with the strong, constructive nature of the negation. The constructive meaning of (19) is that any refutation of b can be effectively converted into a proof of *a*. This is quite different from (20) which states that a refutation of *a* can be converted into a proof of *b*.

7.3 Strong equivalence

The topic of strong equivalence is one that places equilibrium logic in an especially favourable light. The fact that formulas such as (17) and (21) are strongly equivalent was for quite some time laboriously proven by applying the definition of answer sets. But if we know that answer set semantics, or equilibrium logic, is a form of minimal model reasoning in the logic N_5, then formulas or theories equivalent in N_5 are automatically strongly equivalent in ASP: if they have the same models, then they have the same minimal models no matter what additional formulas (or 'background' theories) are added. And, as we saw, formulas or theories are equivalent in N_5 if and only if they are strongly equivalent.

In some ways this property was surprising. For it showed that while ordinary (answer set) equivalence is a computationally hard problem to check, strong equivalence amounts to the simpler (coNP-complete) problem of checking inter-derivability in a monotonic, multivalued logic. A spin-off of this result has been an effort to look more generally at strong equivalence in nonmonotonic logics and to see whether in other cases there may also exist computationally simpler, monotonic checks for strong equivalence, see eg [96, 98] for the cases of default logic and causal theories, respectively, and [95] for a general framework applicable to different logics.

Even the basic characterisation of strong equivalence in ASP may still be of further interest. Since ASP can be combined with abduction and applied in contexts such as planning and diagnosis, it is natural to consider when two answer set programs are equivalent in such contexts, eg equivalent with respect to the explanations, plans, etc that they generate. Inoue and Sakama [44] have recently studied the issue of abductive equivalence and conclude that in ASP explanatory equivalence, for example, coincides with (relativised) strong equivalence. They have also [91] looked at types of equivalence in an inductive logic programming setting. Again, for answer set programs they conclude that strong equivalence captures an important type of inductive equivalence.

8 Concluding remarks

Equilibrium logic generalises the language of answer set programming to full propositional logic and admits a natural extension to the first-order case or to programs with variables. While all the usual definitions of answer sets employ an 'operational', fixpoint condition, equilibrium logic is characterised in terms of minimal models, more in the style of circumscription. It provides a mathematical foundation for ASP, and its base logics, HT and N_5, are maximal logics capturing (strongly) equivalent theories or programs. In the propositional case, equilibrium logic can be implemented in several different ways. The main paths explored up to now have been those of tableaux systems, QBF-reductions and transformations to logic programs. The equilibrium condition in terms of minimal models can be applied to other base logics, to capture known logic programming semantics such as paraconsistent answer sets or well-founded semantics, or to define new systems (eg equilibrium logic over Łukasiewicz 3-valued matrices). The equilibrium condition over here-and-there frames can readily be combined with other relations on possible worlds to yield modal or epistemic extensions of the logic, while the usual Gödel translation of intermediate logics defines embeddings of the logics HT and N_5 into extensions of modal S4 which can be lifted to the equilibrium case (a topic explored elsewhere and still under study[12]).

In future work, the study of first-order and modal extensions is likely to take priority, together with efforts to improve the efficiency of equilibrium logic solvers. At the same time there appears to be still scope for applying equilibrium logic to the foundations and computation of answer sets. The topic of strong equivalence and its variants is still an area of active research and there remain open issues to which equilibrium logic may contribute.

Acknowledgements The current state of research in equilibrium logic would not have been attained without the effort, cooperation and support of a number of colleagues over a period of several years. I owe a very special word of thanks to Agustín Valverde who took to the subject with enthusiasm at a time when there was more promise than fulfilment and who has kept momentum going ever since. I am also very grateful to co-authors Pedro Cabalar, Sergei Odintsov, Torsten Schaub, Hans Tompits and Stefan Woltran; I hope the references here do justice to their significant contributions. I am especially indebted to another co-author, Vladimir Lifschitz, who with typical foresight identified an avenue of opportunity for equilibrium logic and motivated our study of strong equivalence. To Michael Gelfond I am grateful for his long-standing encouragement to continue this line of research at a time when many others expressed scepticism. Marcus Kracht laid a foundation stone for equilibrium logic by helping me to understand its underlying logic, N_5 [48].

References

1. Anger, C., Linke, M.T., Neumann, A., Schaube, T.: The *nomore*++ System. In: Baral, C., et al (eds.) Proc. LPNMR 2005, LNAI, pp. 422–426. Springer, Berlin Heidelberg New York (2005)
2. Avellone, A., Ferrari, M., Miglioli, P.: Duplication-free tableau calculi and related cut-free sequent calculi for the interpolable propositional intermediate logics. Log. J. IGPL **7**(4), 447–480 (1999)
3. Balduccini, M., Gelfond, M., Noguiera, M.: A-Prolog as a tool for declarative programming. In: Proc. 12th Int. Conf. on Software Engineering and Knowledge Engineering, SEKE 2000, Chicago, IL (2000)
4. Balduccini, M., Gelfond, M., Noguiera, M.: The USA-Advisor: a case study in answer set planning.Logic Programming and Nonmonotonic Reasoning, LPNMR 2001, LNAI 2173, Springer, Berlin Heidelberg New York (2001)
5. Baral, C.: Knowlewdge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press, UK (2003)
6. Baral, C., Gelfond, M.: Reasoning about intended actions. In: Proceedings of the Twentieh National Conference on Artificial Intelligence. AAAI, Menlo Park, California
7. Bonner, A., McCarty, L.T.: Adding negation-as-failure to intuitionistic logic programming. In: Proc. NorthAmerican Conference on Logic Programming. pp. 681–703 MIT, Cambridge, Massachusetts (1990)
8. Cabalar, P., Odintsov, S., Pearce, D.: A logic for reasoning about well-founded semantics: preliminary report. In: Marín, R. et al (eds.) CAEPIA 2005: Actas, Volumen **1**: 183–192 Santiago de Compostela, Spain (2005)

[12]An early effort to study modal embeddings for ASP based on the GŽdel translation can be found in [73]. A more complete analysis of modal embeddings of equilibrium logic is part of on-going research.

9. Cabalar, P., Odintsov, S., Pearce, D.: Logical foundations of well-founded semantics. In: Proc. KR ’06, Lake District, UK (2006)
10. Cabalar, P., Odintsov, S., Pearce, D.: Strong negation in well-founded and partial stable semantics for logic programs. In: Proc. Iberamia 06, LNAI, Springer, Berlin Heildelberg New York (2006)
11. Cabalar, P., Odintsov, S., Pearce, D., Valverde, A.: On the Logic and Computation of Partial Equilibrium Models. Proc. Jelia 06, LNAI, Springer, Berlin Heildelberg New York (2006)
12. Cabalar, P., Odintsov, S., Pearce, D., Valverde, A.: Analysing and extending well-founded and partial stable semantics using partial equilibrium logic. Proc. ICLP 06, LNAI, Springer, Berlin Heidelberg New York (2006)
13. Cabalar, P., Pearce, D., Valverde, A.: Reducing propositional theories in equilibrium logic to logic programs. In: Bento, C. et al (eds.) Proceedings EPIA 2005, LNAI 3808, pp. 4–17, Springer, Berlin Heildelberg New York (2005)
14. Calimeri, F., Galizia, S., Rullo F., Calimeri, S., Galizia, M., Ruffolo, P.: Enhancing disjunctive logic programming for ontology specification. Proceedings AGP 2003 (2003)
15. van Dalen, D.: Intuitionistic logic. In: Handbook of Philosophical Logic, Volume III: Alternatives to Classical logic, D. Reidel, Dordrecht (1986)
16. van Dantzig, D.: On the principles of intuitionistic and affirmative mathematics. Indag. Math. **9**, 429–440; 506–517 (1947)
17. Dietrich, J.: Deductive bases of nonmonotonic inference operations. NTZ Report, University of Leipzig, Germany (1994)
18. Dietrich, J.: Inferenzframes. Doctoral dissertation, University of Leipzig (1995)
19. Dix, J.: A classification-theory of semantics of normal logic programs. I. strong properties. Fundam. Inform. **22**(3), 227–255 (1995)
20. Dix, J.: A classification-theory of semantics of normal logic programs. II. weak properties. Fundam. Inform. **22**(3), 257–288 (1995)
21. Došen, K.: Negation as a modal operator. Rep. Math. Log. **20**, 15–27 (1986)
22. Dummett, M.: Elements of Intuitionism. Clarendon, Oxford (1977)
23. Dung, P.M.: Declarative semantics of hypothetical logic programing with negation as failure. In: Proceedings ELP 92, 99. 45–58 (1992)
24. Eiter, T., Fink, M.: Uniform equivalence of logic programs under the stable model semantics. In: Int. Conf. in Logic Programming, ICLP’03, Mumbay, India. Springer, Berlin Heidelberg New York (2003)
25. Eiter, T., Fink, M., Tompits, H., Woltran, S.: Simplifying logic programs under uniform and strong equivalence. In: Lifschitz, V., Niemela, I. (eds.) Proceedings LPNMR 2004, LNAI 2923, Springer, Berlin Heildelberg New York (2004)
26. Eiter, T., Fink, M., Tompits, H., Woltran, S.: Strong and uniform equivalence in answer set programming. Proceedings AAAI 2005, Pittsburg, Pennsylvannia 2005
27. Eiter, T., Tompits, H., Woltran, S.: On solution correspondences in answer-set programming. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05), pp. 97–102. Professional Book Center, Colorado (2005)
28. Eiter, T., Gottlob, G.: On the computational cost of disjunctive logic programming: propositional case. Ann. Math. Artif. Intell., **15**(3–4), 289–323 (1995)
29. Faber, W., Leone, N., Pfeifer, G.: Recursive aggregates in disjunctive logic programs: semantics and complexity. In: Alferes, J.J., Leite, J. (eds.) Logics In Artificial Intelligence. Proceedings JELIA’04, LNAI 3229, pp. 200–212 Springer, Berlin Heidelberg New York (2004)
30. Ferraris, P., Lifschitz, V.: Weight constraints as nested expressions. Theor. Pract. Log. Prog. **5**, 45–74, (2005)
31. Ferraris, P.: Answer sets for propositional theories. In: Baral, C. et al (eds.) Proceedings on Logic Programming and Nonmonotonic Reasoning 05, LNAI 3662, Springer, Berlin Heidelberg New York (2005)
32. Gelfond, M.: Logic programming and reasoning with incomplete information. Ann. Math. Artif. Intell. **12**, 98–116 (1994)
33. Gelfond, M. Lifschitz, V.: The stable model semantics for logic programs. In: Bowen, K., Kowalski, R. (eds.) Proc 5th Int Conf on Logic Programming 2, pp. 1070–1080. MIT, Cambridge, Massachusetts (1988)
34. Gelfond, M., Lifschitz, V.: Logic programs with classical negation. In: Warren, D., Szeredi, P. (eds.) Proc ICLP-90, pp. 579–597. MIT, Cambridge, Massachusetts (1990)
35. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Gener. Comput., 365–387 (1991)

36. Giordano, L., Olivetti, N.: Combining negation-as-failure and embedded implications in logic programs. J. Log. Program. **36**, 91–147 (1998)
37. Gödel, K.: Zum intuitionistischen aussagenkalkül. Anzeiger der Akademie der Wissenschaften Wien, mathematisch, naturwissenschaftliche Klasse. **69**, 65–66 (1932)
38. Greco, S., Leone, N., Scarcello.: Datalog with nested rules. In: Dix, J., et al (eds.) Proceedings on Logic Programming and Knowledge Representation 1997, Port Jefferson, New York, LNCS 1471, pp. 52–65. Springer, Berlin Heidelberg New York (1998)
39. Gurevich, Y.: Intuitionistic logic with strong negation. Stud. Log. **36**(1–2), 49–59 (1977)
40. Hähnle, R.: Towards an efficient tableau proof procedure for multiple-valued logics. In: Börger, Egon, Kleine Büning, Hans, Richter, Michael M., Schönfeld, Wolfgang (eds.) Selected papers from Computer Science Logic, CSL'90, Heidelberg, Germany. Lect. Notes Comput. Sci., **533**, 248–260. Springer, Berlin Heidelberg New York (1991)
41. Hähnle, R.: Automated Deduction in Multiple-valued Logics. Oxford University Press, London, UK (1993)
42. Heyting, A.: Die formalen regeln der intuitionistischen logik. Sitz.ber. Preuss. Akad. Wiss., Phys. Math. Kl., 42–56 (1930)
43. Hosoi, T.: The axiomatization of the intermediate propositional systems S_n of Gödel. J. Coll. Sci., Imp. Univ. Tokyo, **13**, 183–187 (1966)
44. Inoue, K., Sakama, C.: Equivalence in abductive logic. In: Proceedings IJCAI 2005. Edinburg, Scotland (2005)
45. De Jongh, D., Hendricks, L.: Characterization of strongly equivalent logic programs in intermediate logics. Theor. Pract. Log. Prog. **3**(3), 259–270 (2003)
46. Janhunen, T., Niemelä, I., Simons, P., You, J.-H.: Unfolding partiality and disjunctions in stable model semantics. ACM Trans. Comput. Log. (TOCL) **7**(1), 1–37, (2006)
47. Kleine-B§ning, H., Karpinski, M., FlŽgel, A.: Resolution for Quantified Boolean Formulas. Inf. Comput. **117**, 12–18 (1995)
48. Kracht, M.: On extensions of intermediate logics by strong negation. J. Philos. Logic **27**(1), 49–73, (1998)
49. Kraus, S., Lehmann, D., Magidor, M.: Nonmonotonic reasoning, preferential models and cumulative logics. Artif. Intell. **44**, 167–207 (1990)
50. Leone, N., et al.: Data integration: a challenging ASP application. In: Baral, C., et al (eds.) Proc. LPNMR 2005, Diamante, Italy. LNAI, pp. 379–383, Springer, Berlin Heidelberg New York (2005)
51. Leone, N., Pfeifer, G.W., Faber, T., Eiter, G., Gottlob, Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. CoRR: cs.AI/0211004, September, 2003
52. Lierler, Y. CMODELS – SAT-based disjunctive answer set solver. In: Baral, C., et al (eds.) Proc. LPNMR 2005, Diamante, Italy. LNAI, pp. 447–451 Springer, Berlin Heidelberg New York (2005)
53. Lifschitz, V.: Foundations of logic programming. In: Brewka, G. (ed.) Principles of Knowledge Representation, pp. 69–128. CSLI, Stanford, California (1996)
54. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM Trans. Comput. Log. **2**(4), 526–541 (2001)
55. Lifschitz, V., Tang, L., Turner, H.: Nested expressions in logic programs. Ann. Math. Artif. Intell. **25**(3-4), 369–389 (1999)
56. Lin, F.: Reducing strong equivalence of logic programs to entailment in classical propositional logic. In: Proc. KR'02. pp. 170–176, Toulouse, France.
57. Lin, F., Zhao, Y.: ASSAT: Computing answer sets of a logic program by SAT solvers. Artif. Intell. **157**(1–2), 115–137 (2004)
58. Łukasiewicz, J.: Die Logik und das Grundlagenproblem. In: Les Entreties de Zürich sur les Fondaments et la Méthode des Sciences Mathématiques **12**, 6–9 (1938), Zürich, 82–100 (1941)
59. McCarty, L.T.: Clausal intuitionistic logic I. Fixed-point semantics. J. Log. Program. **5** , 1–31 (1988)
60. Makinson, D.: General theory of cumulative inference. In: Reinfrank, M.,et al (eds.) Nonmonotonic reasoning. LNAI 346, Springer, Berlin Heidelberg New York (1989)
61. Makinson, D.: General patterns in nonmonotonic reasoning. In: Gabbay, D., et al (eds.) Handbook of Logic in Artificial Intelligence. Clarendon, Oxford (1994)
62. Maksimova, L.: Craig's interpolation theorem and amalgamable varieties. Doklady Akademii Nauk SSSR, 237, no.6, 1281–1284 (1977) (Translated as Soviet Math. Doklady.)
63. Miller. D.: Logical analysis of modules in logic programming. J. Log. Program. **6**, 79–108 (1989)

64. Mundici, D.: Satisfiability in many-valued sentential logic is NP-complete. Theor. Comp. Sci.. **52**(1-2), 145–153 (1987)
65. Nelson, D.: Constructible falsity. J. Symb. Log. **14**, 16–26 (1949)
66. Odintsov, S., Pearce, D.: A logic for reasoning about paraconsistent answer sets. Proc. LPNMR 2005, Diamante, Italy, LNAI, pp. 343–355 Springer, Berlin Heidelberg New York (2005)
67. Ojeda, M.P., de Guzmán, I., Aguilera, G., Valverde, A.: Reducing signed propositional formulas. Soft Comput. **2**(4), 157–166 (1998)
68. Ono, H.: Model extension theorem and Craig's interpolation theorem for intermediate predicate logics. Rep. Math. Log. **15**, 41–58 (1983)
69. Ortiz, M., Osorio, M.: Nelson's strong negation, safe beliefs and the answer set semantics. In: De Vos, M., Provetti, A. (eds.) ASP 2005 Workshop Proceedings, Bath, UK (2005)
70. Osorio, M., Navarro, J., Arrazola, J.: Equivalence in answer set programming. In: Proc. LOPSTR 2001, Paphos, Cyprus. LNCS 2372, pp. 57–75. Springer, Berlin Heidelberg New York (2001)
71. Osorio, M., Navarro Pérez, J.A., Arrazola, J.: Safe beliefs for propositional theories. Ann. Pure Appl. Logic. **134**(1), 63–82 (2005).
72. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley, Reading, Massachusetts (1994)
73. Pearce, D.: Answer Sets and Nonmonotonic S4. In: Dyckoff, R. (ed.) Extensions of Logic Programming. Proc 4th Int Workshop, LNAI 798, Springer, Berlin Heidelberg New York (1994)
74. Pearce, D.: Nonmonotonicity and Answer Set Inference. In: Proc. LPNMR 1995, LNAI 928, pp. 372–387, Springer, Berlin Heidelberg New York (1995)
75. Pearce, D.: A new logical characterisation of stable models. In: Proceedings Non-Monotonic Extensions of Logic Programming, NMELP 96, Bad Honnef, Germany (1996)
76. Pearce, D.: A new logical characterisation of stable models and answer sets. In: Non-Monotonic Extensions of Logic Programming, NMELP 96, Bad Honnef, Germany. LNCS 1216, pp. 57–70. Springer, Berlin Heidelberg New York (1997)
77. Pearce, D.: From here to there: stable negation in logic programming. In: Gabbay, D., Wansing, H. (eds.) What is negation? Kluwer, Norwell (1999)
78. Pearce, D.: Stable inference as intuitionistic validity. J. Log. Program. 38, 79–91 (1999)
79. Pearce, D.: Simplifying logic programs under answer set semantics. Demoen, B., Lifschtiz, V. (eds.) Proceedings of ICLP04, LNCS 3132, pp. 210–224. Springer, Berlin Heidelberg New York (2004)
80. Pearce, D., de Guzmán, I.P., Valverde, A.: A tableau calculus for equilibrium entailment. In: Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2000, LNAI 1847, pp. 352–367. Springer, Berlin Heidelberg New York (2000)
81. Pearce, D., de Guzmán, I.P., Valverde, A.: Computing equilibrium models using signed formulas. In: Lloyd, John W., et al (eds.) Computational Logic – CL 2000, First International Conference. Proceedings, LNAI 1861, pp. 688–702. Springer, Berlin Heidelberg New York (2000)
82. Pearce, D. Tompits, H., Woltran, S.: Encodings for equilibrium logic and logic programs with nested expressions. In: Proceedings EPIA '01, LNAI , pp. 306–320. Springer, Berlin Heidelberg New York (2001)
83. Pearce, D., Sarsakov, V., Schaub, T., Tompits, H., Woltran, S.: A polynomial translation of logic programs with nested expressions into disjunctive logic programs: preliminary report. In: P. J. Stuckey (ed.) Logic Programming, 18th International Conference, ICLP 2002, Lect. Notes Comput. Sci. 2401, pp. 405–420. Springer, Berlin Heidelberg New York 200
84. Pearce, D., Valverde, A.: Uniform equivalence for equilibrium logic and logic programs. Lifschitz, V., NiemelŁ, I. (eds.) Proceedings of LPNMR'04 , LNAI 2923, pp. 194–206. Springer, Berlin Heidelberg New York (2004)
85. Pearce, D., Valverde, A.: Synonymous theories in answer set programming and equilibrium logic. López de Mántaras, R., Saitta, L. (eds.) Proceedings ECAI 2004, Nieuwe Hemweg, Amsterdam. pp. 388–392. IOS Press, Nieuwe Hemweg, Amsterdam (2004)
86. Pearce, D., Valverde, A.: A first-order nonmonotonic extension of constructive logic. Stud. Log. **80**, 321–246 (2005)
87. Pereira, L.M., Alferes, J.J.: Well founded semantics for logic programs with explicit negation. In: Neumann, B. (ed.) European Conference on Artificial Intelligence, pp.102–106. Wiley, New York (1992)
88. Przymusinski, T.: Stable semantics for disjunctive programs. New Gener. Comput. **9**, 401–424 (1991)

89. Sakama, C.: Inverse entailment in nonmonotonic logic programs. In: Proceedings of the 10th International Conference on Inductive Logic Programming (ILP-2000), LNAI 1866, pp. 209–224, Springer, Berlin Heidelberg New York (2000)
90. Sakama, C., Inoue, K.: Paraconsistent stable semantics for extended disjunctive programs. J. Log. Comput. **5**, 265–285 (1995)
91. Sakama, C., Inoue, K.: Inductive equivalence of logic programs Proceedings ILP 05 Springer, Berlin Heidelberg New York (2005)
92. Sarsakov, V., Schaub, T., Tompits, H., Woltran, S.: nlp: A Compiler for Nested Logic Programming. In: Proceedings of LPNMR 2004, LNAI 2923, pp. 361–364. Springer, Berlin Heidelberg New York (2004)
93. Seipel, D.: Using clausal deductive databases for defining semantics in disjunctive deductive databases. Ann. Math. Artif. Intell. **33**, 347–378 (2001)
94. Simons, P., Niemelä, I., Soininen, T.: Extending and implementing the stable model semantics. Artif. Intell. **138**(1–2), 181–234 (2002)
95. Truszczyński, M.: Strong and uniform equivalence of nonmonotonic theories: an algebraic approach. In: Proceedings KR 2006, AAAI, Menlo Park, California (2006)
96. Turner, H.: Strong equivalence for logic programs and default theories (made easy). In: Proc. of the Logic Programming and Nonmonotonic Reasoning, LPNMR'01, LNAI 2173, pp. 81–92. Springer, Berlin Heidelberg New York (2001)
97. Turner, H.: Strong equivalence made easy: nested expressions and weight constraints. Theor. Prac. Log. Prog. **3**, 609–622 (2003)
98. Turner, H.: Strong equivalence for causal theories. In: Lifschitz, V., Niemelä, I. (eds.) Proc. of the Logic Programming and Nonmonotonic Reasoning, LPNMR'04, LNAI 2923, pp. 289–301. Springer, Berlin Heidelberg New York (2004)
99. Vakarelov, D.: Notes on N-lattices and constructive logic with strong nxegation. Stud. Log. **36**(1–2), 109–125 (1977)
100. Valverde, A.: `tabeql`: A tableau based suite for equilibrium logic. Alferes, J.J., Leite, J. (eds.) Logics in Artificial Intelligence, Proc. JELIA 2004, LNAI 3229, pp. 734–737, Springer, Berlin Heidelberg New York (2004)
101. Vorob'ev, N. N.: A constructive propositional calculus with strong negation (in Russian). Dokl. Akad. Nauk SSSR, **85**, 465–468 (1952)
102. Vorob'ev, N. N.: The problem of deducibility in constructive propositional calculus with strong negation (in Russian). Dokl. Akad. Nauk SSSR, **85**, 689–692 (1952)
103. Wang, K., Zang.: Nested epistemic logic programs. In: Baral, C., et al (eds.), Proc. LPNMR 2005, LNAI 3229, pp. 279–290, Springer, Berlin Heidelberg New York (2005)
104. Woltran, S.: Quantified boolean formulas – from theory to practice. Dissertation, Technische Universität Wien, Institut für Informationssysteme (2003)
105. Woltran, S.: Characterizations for relativized notions of equivalence in answer set programming. In: Alferes, J.J., Leite, J. (eds.) Logics in Artificial Intelligence, Proc. JELIA 2004, LNAI 3229, Springer, Berlin Heidelberg New York (2004)