From Answer Set Logic Programming to Circumscription via Logic of GK

Fangzhen Lin and **Yi Zhou**
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

We first provide a mapping from Pearce's equilibrium logic and Ferraris's general logic programs to Lin and Shoham's logic of knowledge and justified assumptions, a nonmonotonic modal logic that has been shown to include as special cases both Reiter's default logic in the propositional case and Moore's autoepistemic logic. From this mapping, we obtain a mapping from general logic programs to circumscription, both in the propositional and first-order case. Furthermore, we show that this mapping can be used to check the strong equivalence between two propositional logic programs in classical logic.

1 Introduction

Answer Set Programming (ASP) is a new paradigm of constraint-based programming based on logic programming with answer set semantics [Niemelä, 1999; Lifschitz, 1999; Marek and Truszczynski, 1999]. It started out with normal logic programs, which are programs that can have negation but not disjunction. Driven by the need of applications, various extensions have been proposed. These include disjunctive logic programs [Gelfond and Lifschitz, 1991; Leone *et al.*, 2006], nested expressions [Lifschitz *et al.*, 1999], cardinality and weight constraints [Niemelä; and Simons, 2000], and others. Recently, Ferraris [2005] proposed to view formulas in propositional logic as logic programs and showed that they include as special cases all the above mentioned classes of logic programs. In particular, Ferraris [2005] provided a stable model semantics for these formulas using a transformation similar to the original Gelfond-Lifschitz transformation, and showed that this semantics coincides with Pearce's equilibrium logic [Pearce, 1997].

In this paper, we show that this general stable model semantics can be embedded in Lin and Shoham's logic of knowledge and justified assumptions [Lin and Shoham, 1992], aka logic of GK. Besides showing the generality of Lin and Shoham's logic, which was proposed as a general logic for nonmonotonic reasoning, this embedding allows us to obtain a way to check in classical propositional logic whether any given two logic programs are strongly equivalent in almost the same way as in [Lin, 2002]. It also allows us to obtain a mapping from general logic programs to propositional circumscription in the same way as Lin and Shoham [1992] did for mapping normal logic programs to circumscription. As it turned out, this mapping, when extended to first-order case, yields a semantics to first-order general logic programs that is similar to the one proposed recently by Ferraris *et al.* [2007].

We first briefly review Lin and Shoham's logic of GK, Ferraris's general logic programs, and Pearce's equilibrium logic.

2 Logic of GK

The language of the logic of GK is a modal propositional language with two modal operators, K, for knowledge, and A, for assumption. Given a set *Atom* of atoms (also called variables or primitive propositions), formulas in the logic of GK are defined inductively below in BNF notation:

$$F ::= \bot \mid p \mid K(F) \mid A(F) \mid \neg F \mid F \wedge F \mid F \vee F \mid F \rightarrow F,$$

where $p \in \mathit{Atom}$, and $\bot$ is a constant standing for falsity. Formulas without modal operators are called *base formulas*.

The semantics of the logic of GK is defined through *Kripke interpretations*. A Kripke interpretation M is a tuple $\langle W, \pi, R_K, R_A, s\rangle$, where W is a nonempty set whose elements are called *possible worlds*, π a function that maps a possible world to a truth assignment on *Atom*, R_K and R_A binary relations over W representing the accessibility relations for K and A, respectively, and $s \in W$, called the *actual world* of M. The *satisfaction relation* $\models$ between a Kripke interpretation $M = \langle W, \pi, R_K, R_A, s\rangle$ and a formula F is defined inductively as follows:

- $M \not\models \bot$;
- $M \models p$ if $\pi(s)(p) = 1$, where $p \in \mathit{Atom}$;
- $M \models \neg F$ iff $M \not\models F$;
- $M \models F \wedge G$ iff $M \models F$ and $M \models G$;

- $M \models F \vee G$ iff $M \models F$ or $M \models G$;
- $M \models F \to G$ iff $M \not\models F$ or $M \models G$;
- $M \models K(F)$ iff $\langle W, \pi, R_K, R_A, w\rangle \models F$ for any $w \in W$, such that $(s, w) \in R_K$;
- $M \models A(F)$ iff $\langle W, \pi, R_K, R_A, w\rangle \models F$ for any $w \in W$, such that $(s, w) \in R_A$.

We say that a Kripke interpretation M is a *model* of a formula F if M satisfies F. In the following, given a Kripke interpretation M, we let

$$K(M) = \{F \mid F \text{ is a base formula and } M \models K(F)\}$$
$$A(M) = \{F \mid F \text{ is a base formula and } M \models A(F)\}.$$

Notice that $K(M)$ and $A(M)$ are always closed under classical logical entailment. In the following, for any set X of formulas, we let $Th(X)$ be the logical closure of X under classical logic.

Definition 2.1 (*GK* Models) *Given a formula F, a Kripke interpretation M is a* minimal model *of F if M is a model of F and there does not exist another model M_1 of F such that $A(M_1) = A(M)$ and $K(M_1) \subset K(M)$. We say that M is a* GK model[1] *if M is a minimal model of F and $K(M) = A(M)$.*

Lin and Shoham showed that the logic of GK can be used to capture Reiter's default logic [Reiter, 1980] and Moore's auto-epistemic logic [Moore, 1985]. As a consequence, normal logic programs under stable model semantics can be captured in the logic of GK as well. Specifically, they showed that a normal rule

$$r \leftarrow p_1, ..., p_n, \mathsf{not}\, q_1, ..., \mathsf{not}\, q_m$$

can be translated into the following sentence in the logic of GK:

$$Kp_1 \wedge \cdots \wedge Kp_n \wedge \neg Aq_1 \wedge \cdots \wedge \neg Aq_m \to Kr. \qquad (1)$$

They also showed that this translation extends to disjunction logic programs.

In this paper, we shall show that general logic programs proposed by Ferraris [2005] can be captured in the logic of GK as well.

3 General logic programs

Given a set *Atom* of atoms, general logic programs [Ferraris and Lifschitz, 2005] are formulas defined inductively below in BNF notation:

$$F ::= \bot \mid p \mid F \wedge F \mid F \vee F \mid F \to F,$$

where $p \in Atom$. Notice that there is no negation in the language. Instead, for any formula F, $\neg F$ is considered to be a shorthand for $F \to \bot$.

A set $X \subseteq Atom$ of atoms can be considered as a truth assignment in the straightforward way:

$$X \not\models \bot,\ X \models p \text{ iff } p \in X,$$

[1]In [Lin and Shoham, 1992], GK models are called preferred models.

and the usual definition for the logical connectives.

The stable models of a formula (general logic program) are defined by a modified extended Gelfond-Lifschitz transformation. Given a general logic program F, and a set X of atoms, the *reduct of F under X* [Ferraris, 2005], written F^X, is the formula obtained from F by replacing each maximal subformula that is not classically satisfied by X with $\bot$. Thus for example, $(\neg F)^X = \begin{cases} \top & X \models \neg F \\ \bot & \text{otherwise} \end{cases}$

Now a set X of atoms is a stable model of a general logic program F if:

(i) $X \models F^X$;

(ii) there is no proper subset X_1 of X, such that $X_1 \models F^X$.

Example 3.1 Consider the following three general logic program.

$$\begin{aligned} P &= \neg p \to q, \\ Q &= \neg p \vee p, \\ R &= p \to \neg\neg p, \end{aligned}$$

where p, q are atoms. The maximal subformula in P that is false under $\{q\}$ is p, thus $P^{\{q\}}$ is $\neg\bot \to q$, which is satisfied by $\{q\}$, but not by $\emptyset$. Therefore, $\{q\}$ is a stable model of P. On the other hand, $P^{\{p\}}$ is $\bot \to \bot$, which is satisfied by $\{p\}$ as well as its subset $\emptyset$. Therefore, $\{p\}$ is not a stable model of P. It can be seen that $\{q\}$ is the only stable model of P. Similarly, it can be shown that Q has two stable models, $\{p\}$ and $\emptyset$, and R has exactly one stable model $\emptyset$.

4 Pearce's equilibrium logic

Pearce's equilibrium logic [Pearce, 1997] is based on the logic of here-and-there, a non-classical logic. Given a set *Atom* of atoms, formulas of *Atom* are exactly the same as in the case of general logic programs. Thus, negation in equilibrium logic is considered a shorthand as well.

The semantics of the logic of here-and-there is defined in terms of *HT-interpretations*, which are pairs $\langle X, Y\rangle$ of sets atoms such that $X \subseteq Y$. The *HT satisfaction relation*[2] $\models$ between an HT-interpretation $\langle X, Y\rangle$ and a formula F is defined recursively as follows:

- For $p \in Atom$, $\langle X, Y\rangle \models p$ if $p \in X$;
- $\langle X, Y\rangle \not\models \bot$;
- $\langle X, Y\rangle \models F \wedge G$ if $\langle X, Y\rangle \models F$ and $\langle X, Y\rangle \models G$;
- $\langle X, Y\rangle \models F \vee G$ if $\langle X, Y\rangle \models F$ or $\langle X, Y\rangle \models G$;
- $\langle X, Y\rangle \models F \to G$ if
  (i) $\langle X, Y\rangle \not\models F$ or $\langle X, Y\rangle \models G$, and
  (ii) $Y \models F \to G$.

[2]We overload $\models$, and use it to stand for satisfaction relations for modal logic, classical logic, and logic of here-and-there. Which one it stands for should be clear from the context.

An HT interpretation $\langle X, Y\rangle$ is an *equilibrium model* of a formula F if $X = Y$, $\langle X, Y\rangle \models F$, and there is no proper subset X_1 of X, such that $\langle X_1, Y\rangle \models F$. Ferraris [2005] showed that the stable models of a formula are essentially the same as its equilibrium models.

Theorem 1 (Ferraris) *Let X be a set of atoms and F a general logic program, X is a stable model of F iff $\langle X, X\rangle$ is an equilibrium model of F.*

5 From general logic program and equilibrium logic to the logic of GK

In this section, we present a translation from a general logic program (also a formula in equilibrium logic) to a formula in the logic of GK, and show that under the translation, stable models (thus equilibrium models) coincide with GK models in the logic of GK.

Given a general logic program F, we define two formulas F_A and F_{GK} in the logic of GK as follows:

1. F_A is obtained from F by simultaneously replacing each atom p by Ap.
2. F_{GK} is defined inductively as follows:
   - $\bot_{GK} = \bot$;
   - For $p \in Atom$, $p_{GK} = Kp$;
   - $(F \odot G)_{GK} = F_{GK} \odot G_{GK}$ ($\odot$ is $\wedge$ or $\vee$).
   - $(F \rightarrow G)_{GK} = (F_{GK} \rightarrow G_{GK}) \wedge (F \rightarrow G)_A$.

It can be shown that for a normal logic program F, F_{GK} is equivalent to the translation by Lin and Shoahm [1992] given in Section 2 under

$$\bigwedge_{p \in Atom} Kp \rightarrow Ap, \qquad (2)$$

and that for any formula W in the logic of GK, M is a GK model of W iff M is a GK model of $W \wedge (2)$.

This translation is also similar to the mapping from formulas in equilibrium logic to quantified boolean formulas given in [Pearce *et al.*, 2001]. We shall discuss this in more detail in a later section.

To illustrate, consider the three programs in Example 3.1. P_{GK} is

$$((\neg p)_{GK} \rightarrow q_{GK}) \wedge (\neg p \rightarrow q)_A,$$

which is equivalent (in classical logic) to

$$((\neg p_{GK} \wedge \neg p_A) \rightarrow Kq) \wedge (\neg p_A \rightarrow q_A),$$

which is

$$(\neg Kp \wedge \neg Ap) \rightarrow Kq) \wedge (\neg Ap \rightarrow Aq). \qquad (3)$$

Now let M be a model of the above sentence. If $p \in A(M)$, then (3) holds no matter what $K(M)$ is, thus its minimal model is $K(M) = Th(\emptyset)$, so cannot be a GK model. Now if $p \notin A(M)$, then (3) is equivalent to $(\neg Kp \rightarrow Kq) \wedge Aq$. Thus $q \in A(M)$. Thus if M is a minimal model, then $K(M) = Th(\{q\})$. And if $A(M) = K(M)$, then M is a GK model. What we have shown here is that in any GK model M of (3), $K(M) = A(M) = Th(\{q\})$. The existence of such a model is apparent.

It can be similarly shown that Q_{GK} is equivalent to $Ap \rightarrow Kp$, and that M is a GK model of Q_{GK} iff $K(M) = A(M) = Th(\{p\})$ or $K(M) = A(M) = Th(\{\top\})$. And R_{GK} is equivalent to $Kp \rightarrow Ap$, and that M is a GK model of R_{GK} iff $K(M) = A(M) = Th(\{\top\})$. Thus for these three programs, their GK models correspond one-to-one with their stable models. In general, we have the following result.

Theorem 2 *Let X be a set of atoms and F a general logic program. The following three statements are equivalent.*

1. *X is a stable model of F.*
2. *$\langle X, X\rangle$ is an equilibrium model of F.*
3. *There is a GK model M of F_{GK} such that $K(M) = A(M) = Th(X)$.*

Proof sketch: Given a general logic program F, two set of atoms X and Y such that $X \subseteq Y$ and a Kripke structure M such that $K(M) = Th(X)$, $A(M) = Th(Y)$, by induction on the structure of F, we have $X \models F^Y$ iff $M \models F_{GK}$. From this, 2 $\Leftrightarrow$ 3 follows. By Theorem 1, 1 $\Leftrightarrow$ 2. ■

6 From general logic programs and equilibrium logic to circumscription: propositional case

Given their mapping (1) from normal logic program to the logic of GK, Lin and Shoham [1992] showed that stable model semantics for normal logic programs can be captured in circumscription [McCarthy, 1986] as follows. Given a set $Atom = \{p, q, ...\}$ of atoms, let $Atom' = \{p', q', ...\}$ be a new set of atoms. Given a normal logic program F, let $C(F)$ be the conjunction of the sentences:

$$p_1 \wedge \cdots \wedge p_n \wedge \neg q_1' \wedge \cdots \wedge \neg q_m' \rightarrow r,$$

for each rule

$$r \leftarrow p_1, ..., p_n, \mathsf{not}\, q_1, ..., \mathsf{not}\, q_m$$

in F. Lin and Shoham [1992] showed that X is a stable model of F iff $X \cup X'$ is a model of

$$\bigwedge_{p \in Atom} (p \leftrightarrow p') \wedge Circum(C(F); Atom),$$

where $Circum(W; Q)$ is the circumscription of the atoms in Q in W (with all other atoms fixed). Lin and Shoham also showed that this result can be extended to disjunctive logic programs. Using the same idea, we can capture the stable model semantics of general logic program and equilibrium logic in circumscription as well.

Let $Atom$ be a set of atoms. Again let $Atom' = \{p' | p \in Atom\}$ be a set of new atoms. Given any general logic program F in the language $Atom$, let $C(F)$ be the result obtained from F_{GK} by replacing every Kp in it by p and every Ap in it by p', for every $p \in Atom$.

Theorem 3 *For any general logic program F in the language $Atom$, any set $X \subseteq Atom$, the following four statements are equivalent*

1. *X is a stable model of F.*
2. *$\langle X, X\rangle$ is an equilibrium model of F.*
3. *There is a GK model M of F_{GK} such that $K(M) = A(M) = Th(X)$.*
4. *$X \cup X'$ is a model of*

$$\bigwedge_{p\in Atom} (p \leftrightarrow p') \wedge Circum(C(F); Atom). \qquad (4)$$

Proof sketch: 3 $\Leftrightarrow$ 4 follows from the definitions of GK models and circumscription. ■

Interestingly, our translation $C(F)$ that embeds general logic programs and equilibrium logic in circumscription is exactly the same as the one by Pearce *et al.* [Pearce *et al.*, 2001] for embedding equilibrium logic in quantified boolean formulas. They showed that $\langle X, X\rangle$ is an equilibrium model of a formula F in equilibrium logic iff X' is a model of the following quantified boolean formula:

$$F' \wedge \neg\exists Atom((Atom < Atom') \wedge C(F)),$$

where F' is the formula obtained from F by replacing every atom p by p', and $Atom < Atom'$ stands for

$$\bigwedge_{p\in Atom} (p \rightarrow p') \wedge \neg \bigwedge_{p\in Atom} (p' \rightarrow p).$$

While propositional circumscription is also a quantified boolean formula, it is a well studied formalism. There are many known results about circumscription that we can use. Mapping logic programs to circumscription helps understanding both formalisms.

Notice that the formula $\bigwedge_{p\in Atom}(p \leftrightarrow p')$ is equivalent to

$$[\bigwedge_{p\in Atom} (p \rightarrow p')] \wedge [\bigwedge_{p\in Atom} (p' \rightarrow p)].$$

Thus (4) is equivalent to

$$[\bigwedge_{p\in Atom} (p \rightarrow p')] \wedge [\bigwedge_{p\in Atom} (p' \rightarrow p)] \wedge Circum(C(F); Atom),$$

which is equivalent to

$$[\bigwedge_{p\in Atom} (p' \rightarrow p)] \wedge Circum(C(F) \wedge \bigwedge_{p\in Atom} (p \rightarrow p'); Atom),$$

as the atoms (predicates) to be minimized occur only negatively in $\bigwedge_{p\in Atom}(p \rightarrow p')$. Putting the formula $\bigwedge_{p\in Atom}(p \rightarrow p')$ into the theory in the circumscription is good as it can be used to simplify $C(F)$.

Proposition 6.1 *If $\bigwedge_{p\in Atom}(p \rightarrow p') \models C(F) \leftrightarrow W$, then (4) is equivalent to*

$$\bigwedge_{p\in Atom} (p \leftrightarrow p') \wedge Circum(W; Atom).$$

Consider again the three programs in Example 3.1. For P, $C(P)$ is equivalent to $\neg p \wedge \neg p' \rightarrow q$, which is equivalent to $\neg p' \rightarrow q$ under $(p \rightarrow p') \wedge (q \rightarrow q')$. Thus for this program, (4) is equivalent to

$$(p \leftrightarrow p') \wedge (q \leftrightarrow q') \wedge Circum(\neg p' \rightarrow q; \{p, q\}),$$

which is equivalent to

$$(p \leftrightarrow p') \wedge (q \leftrightarrow q') \wedge \neg p \wedge (\neg p' \leftrightarrow q),$$

which has a unique model $\{q, q'\}$.

For $Q = \neg p \vee p$, $C(Q)$ is equivalent to $p' \rightarrow p$. Thus for this program, (4) is equivalent to

$$(p \leftrightarrow p') \wedge Circum(p' \rightarrow p; \{p\}),$$

which is equivalent to $p \leftrightarrow p'$, which has two models p and $\neg p$.

7 From general logic programs and equilibrium logic to circumscription: first order case

As in [Lin and Shoham, 1992], we can extend the above mapping to the first-order case. First of all, we extend logic programs to first-order case. Let L be a relational first-order language with equality, i.e. it has no proper functions. By an atom, we mean an atomic formula including equality atoms.

In the following, let Σ_{una} be the set of unique names assumptions on constants: $c_1 \neq c_2$ for any two distinct constants c_1 and c_2.

A first-order general logic program is a first-order sentence in the following set:

$$F ::= \bot \mid A \mid F \wedge F \mid F \vee F \mid F \rightarrow F \mid \forall x F \mid \exists x F,$$

where A is an atom, and x a variable. Again, for any general logic program F, $\neg F$ is considered to be a shorthand for $F \rightarrow \bot$.

Now let M be a finite model of Σ_{una} with domain D. Let σ be the mapping from constants to D under M. Clearly, for any distinct constants c_1 and c_2, $\sigma(c_1) \neq \sigma(c_2)$. We say that M is a stable model of a first-order general logic program F if $T(M)$, the set of ground facts true in M:

$$T(M) = \{P(\vec{u}) \mid P \text{ a predicate}, \vec{u} \in P^M\}$$

is a stable model of the general logic program F_M, which, called the *grounding* of F on M, is obtained from F and M in two steps:

1. First, replace every constant c in F by $\sigma(c)$, every subformula of the form $\forall x W$ in it by $\bigwedge_{u\in D} W(x/u)$, and every subformula of the form $\exists x W$ in it by $\bigvee_{u\in D} W(x/u)$, where $W(x/u)$ is the result of replacing every free occurrence of x in W by u. The order by which these subformulas are replaced does not matter.
2. In the expression obtained by the first step, replace every equality atom $u = u$ by $\top$, and every $u = u'$ for distinct u and u' by $\bot$.

Example 7.1 Consider the following four programs:

$$F_1 = \exists x p(x) \land \exists x(\neg p(x) \to q),$$
$$F_2 = \exists x p(x) \land [(\exists x \neg p(x)) \to q],$$
$$F_3 = \exists x p(x) \land [\neg(\exists x p(x)) \to q],$$
$$F_4 = \exists x p(x) \land \forall x(\neg p(x) \to q).$$

Now consider a structure with two elements $\{a, b\}$. The grounding of the four programs on this domain are

$$(p(a) \lor p(b)) \land ((\neg p(a) \to q) \lor (\neg p(b) \to q)),$$
$$(p(a) \lor p(b)) \land ((\neg p(a) \lor \neg p(b)) \to q),$$
$$(p(a) \lor p(b)) \land (\neg(p(a) \lor p(b)) \to q),$$
$$(p(a) \lor p(b)) \land (\neg p(a) \to q) \land (\neg p(b) \to q),$$

respectively. So for this domain, F_1 and F_3 have the same stable models, $\{p(a)\}$ and $\{p(b)\}$, and F_2 and F_4 have the same stable models, $\{p(a), q\}$ and $\{p(b), q\}$. It is easy to see that this is the case for any given domain: F_1 and F_3 have the same stable models, and F_2 and F_4 have the same stable models.

We now show that the stable models of first-order general logic programs can be captured in circumscription as well.

Let Δ be the set of predicates in the language. Let Δ' be a set of new predicates, one for each P in Δ with the same arity and denoted by P'. Now given a first-order general logic program F, let $C(F)$ be the first-order sentence defined inductively as follows.

- $C(\bot)$ is $\bot$.
- If W is an atomic formula, then $C(W)$ is W.
- $C(W_1 \odot W_2)$ is $C(W_1) \odot C(W_2)$, where $\odot \in \{\land, \lor\}$.
- $C(\forall x W)$ is $\forall x C(W)$, and $C(\exists x W)$ is $\exists x C(W)$.
- $C(W_1 \to W_2)$ is $(C(W_1) \to C(W_2)) \land (W_1' \to W_2')$, where W' is the result of replacing every predicate P in W by P'.

The stable models of F is then the circumscription of all the predicates in Δ in $C(F)$, together with the following axiom

$$\bigwedge_{P \in \Delta} \forall \vec{x}(P(\vec{x}) \leftrightarrow P'(\vec{x})). \tag{5}$$

Theorem 4 *Let M be a finite model of Σ_{una}. M is a stable model of F iff M' is a model of*

$$Circum(C(F); \Delta) \land (5), \tag{6}$$

where M' is the conservative extension of M under (5).

Similar to Proposition 6.1, we have

Proposition 7.1 *If $\bigwedge_{P \in \Delta} \forall \vec{x}(P(\vec{x}) \to P'(\vec{x})) \models C(F) \leftrightarrow W$, then (6) is equivalent to*

$$Circum(W; \Delta) \land (5).$$

Interestingly, Ferraris *et al.* [2007] also proposed a semantics for general first-order logic programs and showed that their semantics is equivalent to (6) when restricted on the predicates in the original program, that is, when all new predicates P' in (6) are existentially quantified.

Example 7.2 Consider the programs in Example 7.1

- $C(F_1)$ is

  $$\exists x p(x) \land \exists x[(\neg p(x) \land \neg p'(x) \to q) \land (\neg p'(x) \to q')].$$

  which is equivalent to $\exists x p(x) \land (\neg \exists x p'(x) \to q)$ under $\forall x.p(x) \to p'(x)$. Therefore, under (5), $Circum(C(F_1), \{p, q\})$ is equivalent to

  $$\exists! x p(x) \land ((\neg \exists x p'(x)) \leftrightarrow q),$$

  thus equivalent to

  $$\exists! x p(x) \land \neg q,$$

  which can be considered to be the first-order semantics of F_1. If $D = \{a, b\}$, then there are exactly two models of this sentence, $\{p(a)\}$ and $\{p(b)\}$.
- $C(F_2)$ is

  $$\exists x p(x) \land (\exists x(\neg p(x) \land \neg p'(x)) \to q) \land (\exists x \neg p'(x) \to q'),$$

  which is equivalent to

  $$\exists x p(x) \land (\exists x \neg p'(x) \to q)$$

  under $\forall x.p(x) \to p'(x)$. Therefore, $Circum(C(F_2), \{p, q\})$ is equivalent to

  $$\exists! x p(x) \land (\exists x \neg p'(x) \leftrightarrow q),$$

  under (5), thus equivalent to

  $$\exists! x p(x) \land ((\exists x \neg p(x)) \leftrightarrow q),$$

  which can be considered to be the first-order semantics of F_2. If $D = \{a, b\}$, then there are exactly two models of this sentence, $\{p(a), q\}$ and $\{p(b), q\}$.
- $C(F_3)$ is

  $$\exists x p(x) \land (\neg \exists x p(x) \land \neg \exists x p'(x) \to q) \land (\neg \exists x p'(x) \to q'),$$

  which is equivalent to $C(F_1)$ under (5). Thus F_1 and F_3 are equivalent.
- $C(F_4)$ is

  $$\exists x p(x) \land \forall x[(\neg p(x) \land \neg p'(x) \to q) \land (\neg p'(x) \to q')],$$

  which is equivalent to $C(F_2)$. Thus F_2 and F_4 are equivalent.

8 Strong equivalence

The notion of strong equivalence [Lifschitz *et al.*, 2001] is important in logic programming. For disjunctive logic programs, research by Lin and Chen [2005] and Eiter *et al.* [2006] show that interesting programs transformation rules can be designed based on the notion.

According to Ferraris and Lifschitz [2005], two general logic programs F and G are said to be *strongly equivalent* if for any formula F_1 that contains an occurrence of F, F_1 has the same stable models as the formula obtained from it by replacing an occurrence of F by G. They showed that for any F and G, they are strongly equivalent iff F and G are equivalent in the logic here-and-there.

As it turns out, our mapping from equilibrium logic to logic of GK also embeds logic of here-and-there to modal logic. Thus the problem of deciding whether two programs are strongly equivalent can be reduced to checking whether certain modal logic formulas are valid, and that, because of the special format of these modal formulas, can in turn be reduced to checking whether certain propositional formulas are tautologies.

Theorem 5 *Let F be a formula in equilibrium logic, X and Y two sets of atoms such that $X \subseteq Y$, and M a Kripke interpretation such that $K(M) = Th(X)$ and $A(M) = Th(Y)$. We have that $\langle X, Y \rangle \models F$ iff $M \models F_{GK}$.*

Theorem 6 *Let F and G be two general logic programs. The following conditions are equivalent.*

1. *F and G are strong equivalent.*

2. *F is equivalent to G in the logic here-and-there.*

3. $\bigwedge_{p \in Atom}(Kp \rightarrow Ap) \models (F \leftrightarrow G)_{GK}$.

4. $\bigwedge_{p \in Atom}(Kp \rightarrow Ap) \models F_{GK} \leftrightarrow G_{GK}$.

5. $\bigwedge_{p \in Atom}(p \rightarrow p') \models C(F \leftrightarrow G)$.

6. $\bigwedge_{p \in Atom}(p \rightarrow p') \models C(F) \leftrightarrow C(G)$.

Corollary 7 *The problem of deciding whether two general logic programs are strongly equivalent is co-NP complete.*

9 Conclusion

We showed that the logic of GK proposed by Lin and Shoham is flexible enough to handle stable model semantics of general logic programs. Because of this, the stable model semantics of general logic programs can also be formulated in circumscription, in both propositional and first-order case. For future work, we plan to make use of the expressive power of GK in other applications.

Acknowledgments

We thank Vladimir Lifschitz for stimulating discussions on topics related to this paper, and for his helpful comments on earlier versions of this paper. This work was supported in part by HK RGC under grant HKUST6170/04E, and by China NSFC under grant 60496322.

References

[Eiter *et al.*, 2006] T. Eiter, M. Fink, H. Tompits, P. Traxler, and S. Woltran. Replacements in non-ground answer-set programming. In *KR'2006*, pages 340–351, 2006.

[Ferraris, 2005] P. Ferraris. Answer sets for propositional theories. In *LPNMR*, pages 119–131, 2005.

[Ferraris and Lifschitz, 2005] P. Ferraris and V. Lifschitz. Mathematical foundations of answer set programming. In *We Will Show Them! (1)*, pages 615–664. 2005.

[Ferraris *et al.*, 2007] P. Ferraris, J. Lee, and V. Lifschitz. A new perspective on stable models. In *Proceedings of IJCAI'07* (this volume), 2007.

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

[Leone *et al.*, 2006] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. *ACM Transactions on Computational Logic*, 2006. To appear.

[Lifschitz *et al.*, 1999] V. Lifschitz, L. R. Tang, and H. Turner. Nested expressions in logic programs. *Annals of Mathematics and Artificial Intelligence*, 25:369-389, 1999.

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

[Lifschitz, 1999] V. Lifschitz. Action languages, answer sets and planning. In *The Logic Programming Paradigm: A 25-Year Perspective.* K.R. Apt, V.W. Marek, M. Truszczynski, D.S. Warren, eds, Springer-Verlag, 1999.

[Lin and Chen, 2005] F. Lin and Y. Chen. Discovering classes of strongly equivalent logic programs. In *Proc. of IJCAI'95*, pages 516–521, 2005.

[Lin and Shoham, 1992] F. Lin and Y. Shoham. A logic of knowledge and justified assumptions. *Artificial Intelligence*, 57:271–289, 1992.

[Lin, 2002] F. Lin. Reducing strong equivalence of logic programs to entailment in classical propositional logic. In *Proc. of KR'02*, pages 170–176, 2002.

[Marek and Truszczynski, 1999] V. W. Marek and M. Truszczynski. Stable logic programming - an alternative logic programming paradigm. In *The Logic Programming Paradigm: A 25-Year Perspective.* K.R. Apt, V.W. Marek, M. Truszczynski, D.S. Warren, eds, Springer-Verlag, 1999.

[McCarthy, 1986] J. McCarthy. Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence*, 28:89–118, 1986.

[Moore, 1985] R. Moore. Semantical considerations on non-monotonic logic. *Artificial Intelligence*, 25(1):75–94, 1985.

[Niemelä; and Simons, 2000] I. Niemelä; and P. Simons. Extending the smodels system with cardinality and weight constraints. pages 491–521, 2000.

[Niemelä, 1999] I. Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Ann. Math. and AI*, 25(3-4):241–273, 1999.

[Pearce *et al.*, 2001] D. Pearce, H. Tompits, and S. Woltran. Encodings for Equilibrium Logic and Logic Programs with Nested Expressions. In *Proc. EPIA-01*: 306–320, 2001.

[Pearce, 1997] D. Pearce. A new logical characterisation of stable models and answer sets. In *Non-Monotonic Extensions of Logic Programming*: 57–70, 1997.

[Reiter, 1980] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.