Strong equivalence of nonmonotonic theories — an algebraic approach (abstract)

Mirosław Truszczyński
Department of Computer Science
University of Kentucky,
Lexington, KY 40506-0046, USA

Abstract

We show that the concept of strong equivalence of logic programs can be generalized to an abstract algebraic setting of operators on complete lattices. Our results imply characterizations of strong equivalence for several nonmonotonic logics including logic programming with aggregates, default logic and a version of autoepistemic logic.

Introduction

The concept of strong equivalence of logic programs was introduced in (Lifschitz, Pearce, & Valverde 2001). Two logic programs P and Q are *strongly equivalent* if for every logic program R, programs $P \cup R$ and $Q \cup R$ have the same stable models. It follows that if P is strongly equivalent to Q and S is an arbitrary program containing P, then P can be replaced in S with Q and the stable models of the resulting program will remain the same as those of S. Thus, strong equivalence is fundamental to program rewriting and optimization and so, it received much attention in the literature (Lifschitz, Pearce, & Valverde 2001; Lin 2002; Turner 2003).

Characterizing the property of the "equivalence for substitution" is straightforward in the case of classical (monotone) logics. For nonmonotonic logics the situation is more complex. For instance, in logic programming with the semantics of stable models (Gelfond & Lifschitz 1988), having the same models, stable models or both (these are natural generalizations of the condition of having the same models, which works in the classical logic case) is too weak to guarantee the equivalence for substitution. Indeed, the following two logic programs

$$P = \{p\} \text{ and } Q = \{p \leftarrow \mathbf{not}(q)\}$$

have the same stable models (each program has $\{p\}$ as its *only* stable model). However, $P \cup \{q\}$ and $Q \cup \{q\}$ have *different* stable models. The only stable model of $P \cup \{q\}$ is $\{p, q\}$ and the only stable model of $Q \cup \{q\}$ is $\{q\}$.

(Lifschitz, Pearce, & Valverde 2001) studied the problem of strong equivalence in the setting of *logic programs with nested expressions*, also referred to as *nested logic programs* (Lifschitz, Tang, & Turner 1999). To interpret nested logic programs (Lifschitz, Tang, & Turner 1999) introduced the notion of an *answer set*. Nested logic programming with the semantics of answer sets generalizes disjunctive logic programming with the semantics of answer sets (Gelfond & Lifschitz 1991) and, therefore, also normal logic programming with the semantics of stable models.

(Lifschitz, Pearce, & Valverde 2001) presented a characterization of strong equivalence of nested logic programs by exploiting properties of the logic *here-and-there* (Heyting 1930). (Turner 2001; Lin 2002; Turner 2003) continued these studies. In particular, (Turner 2001; 2003) introduced the notion of an *se-model*, defined as a certain pair of sets of literals, and proved that two nested logic programs are strongly equivalent if and only if they have the same se-models. In addition, (Turner 2001) demonstrated that the approach of se-models extends to the case of default theories.

Results and proofs in (Lifschitz, Pearce, & Valverde 2001; Turner 2001; Lin 2002; Turner 2003) show common themes and similarities. To a large degree, it is due to the fact that all characterizations of strong equivalence developed there, are rooted, if not directly then at least implicitly, in the logic *here-and-there*. In this paper we show that there are additional reasons behind these similarities, related to the fact that semantics of many nonmonotonic logics can be introduced in abstract algebraic terms. The study of an algebraic account of strong and uniform equivalence is the main objective of our paper.

Our tool is the approximation theory, which deals with properties of fixpoints of operators on complete lattices (Denecker, Marek, & Truszczyński 2000). It provides an algebraic account of several nonmonotonic logics including (normal) logic programming, default logic and autoepistemic logic, and allows one to state and prove properties of these logic in a uniform, general and abstract way (Denecker, Marek, & Truszczyński 2003). Other applications of the approximation theory include the development of semantics of logic programs with aggregates (Pelov. 2004; Pelov, Denecker, & Bruynooghe 2004) and an abstract account of splitting theorems (Vennekens, Gilis, & Denecker 2004b; 2004a).

In this paper, we show that the concept of strong equivalence can be extended to the case of operators on complete lattices and can be given a purely algebraic account. Our results yield as corollaries characterizations of strong and uniform equivalence for every nonmonotonic logic whose semantics can be defined in terms of fixpoints of operators

on lattices.

An overview of the approximation theory

We start with a brief overview of the approximation theory (Denecker, Marek, & Truszczyński 2000), which establishes properties of fixpoints of certain types of operators on complete lattices. The theory yields a uniform abstract treatment of semantics of major nonmonotonic logics.

A *lattice* is a partially ordered set $\langle L, \leq \rangle$ such that every two element set $\{x, y\} \subseteq L$ has a *least upper bound*, $x \vee y$, and a *greatest lower bound*, $x \wedge y$. It is common to omit the explicit mention of the lattice ordering relation from the notation and we follow this convention here.

A lattice L is *complete* if every subset of L has both least upper and greatest lower bounds. In particular, a complete lattice has a least element, denoted by $\bot$, and a greatest element, denoted by $\top$.

An *operator* on a lattice L is any function from L to L. An operator O on L is *monotone* if for every $x, y \in L$ such that $x \leq y$ we have $O(x) \leq O(y)$. Similarly, an operator O on L is *antimonotone* if for every $x, y \in L$ such that $x \leq y$ we have $O(y) \leq O(x)$. *Constant* operators are both monotone and antimonotone.

Let L be an operator on a lattice L. An element $x \in L$ is a *prefixpoint* (a *fixpoint*, respectively) of O if $O(x) \leq x$ ($O(x) = x$, respectively). If an operator O has a least fixpoint, we denote this fixpoint by $lfp(O)$. The following theorem by Tarski and Knaster establishes a fundamental property of monotone operators on complete lattices (Tarski 1955).

Theorem 1 *Let O be a monotone operator on a complete lattice L. Then, O has a least fixpoint and this least fixpoint is also the least prefixpoint of O.*

Let L be a lattice. The key intuition of the approximation theory from (Denecker, Marek, & Truszczyński 2000) is to view elements of the cartesian product L^2 of L as *approximations* and order them according to their "precision". Namely, if $x, y, z \in L$ and $x \leq z \leq y$, then $(x, y) \in L^2$ *approximates* $z \in L$. Moreover, if for some $x', y' \in L$, $x' \leq z \leq y'$, $x \leq x'$ and $y' \leq y$, then (x', y') is a "tighter" or "more precise" approximation to z.

An element $(x, y) \in L^2$ is a "proper" approximation only if $x \leq y$. Otherwise, there is nothing that the pair (x, y) approximates. Nevertheless, it is useful to extend the intuition of approximation to all elements in L^2 and define the *precision* ordering on L^2, $\leq_p$ in symbols, as follows:

$$(x, y) \leq_p (x', y') \quad \textit{if } x \leq x' \textit{ and } y' \leq y.$$

One can check that if L is a complete lattice then L^2 with the ordering $\leq_p$ is also a complete lattice. We call this lattice the *product lattice* of L. Product lattices, as well as closely related algebraic structures called *bilattices*, were thoroughly studied in (Ginsberg 1988; Fitting 2002).

Let A be an operator on the product lattice L^2. We denote by A^1 and A^2 the corresponding *projection* functions from L^2 to L, that is, the unique functions such that

$$A(x, y) = (A^1(x, y), A^2(x, y)).$$

We say that A is *symmetric* if $A^1(x, y) = A^2(y, x)$. It follows directly from the definition that if an operator $A : L^2 \to L^2$ is symmetric then for every $x \in L$, $A^1(x, x) = A^2(x, x)$.

Let A be a $\leq_p$-monotone operator on L^2 and $x, x', y \in L$. If $x < x'$ then $(x, y) \leq_p (x', y)$. By the $\leq_p$-monotonicity of A, $A(x, y) \leq_p A(x', y)$ and so, $A^1(x, y) \leq A^2(x', y)$. It follows that $A^1(\cdot, y)$, which is an operator on L, is monotone. Similarly, for every $x \in L$, the operator $A^2(y, \cdot)$ is monotone, too.

We are ready now to introduce the key concepts of the approximation theory from (Denecker, Marek, & Truszczyński 2000).

Definition 1 *An operator A of L^2 is* approximating *if it is symmetric and $\leq_p$-monotone. If A is an approximating operator on L^2, then the corresponding A-*stable *operator S_A is defined by*

$$S_A(x, y) = (lfp(A^1(\cdot, y), lfp(A^1(\cdot, x)).$$

(*By the symmetry of A, an equivalent definition of S_A is $S_A(x, y) = (lfp(A^2(y, \cdot), lfp(A^2(x, \cdot))$.*)

Let O be an operator on a lattice L. An operator A of L^2 is an *approximating* operator for O if A is approximating and if for every $x \in L$, $A(x, x) = (O(x), O(x))$. In this case, we also say that S_A is an *A-stable operator* for O.

Every operator O on a lattice L has an approximating operator. Let A be an operator on the product lattice L^2 defined by:

$$A^1(x, y) = \begin{cases} \bot & \text{if } x < y \\ O(x) & \text{if } x = y \\ \top & \text{otherwise} \end{cases}$$

and $A^2(x, y) = A^1(y, x)$. The symmetry of A is guaranteed by the second part of the definition. Due to the symmetry of A, to prove that A is $\leq_p$ monotone, it suffices to show that if $(x, y) \leq_p (x', y')$ then $A^1(x, y) \leq A^1(x', y')$. It is, however, straightforward. If $x < y$, $A^1(x, y) = \bot$. If it is not the case that $x' \leq y'$, then $A^2(x', y') = \top$. Otherwise, $x = y = x' = y'$ and $A^1(x, y) = O(x) = O(x') = A^1(x', y')$.

In general, approximating operators are not unique. For monotone and antimonotone operators we distinguish particular approximating operators. Namely, if O is monotone, we define the operator C_O by setting $C_O(x, y) = (O(x), O(y))$. If O is antimonotone, we define C_O by setting $C_O(x, y) = (O(y), O(x))$. In each case, one can verify that C_O is an approximating operator for O — we call it *canonical*.

Definition 2 ((Denecker, Marek, & Truszczyński 2000)) *Let O be an operator on L and let A be an approximating operator for O. An element $x \in L$ is an A-stable fixpoint of O if $x = S^1_A(x, x)$. We denote the set of A-stable fixpoints of O by $St(O, A_O)$.*

It follows directly from the definition that x is an A-stable fixpoint of O if and only if

$$x = lfp(A^1(\cdot, x)).$$

We note that the concept of a stable fixpoint of an operator on a complete lattice is parameterized with an approximating operator. Thus, later in the paper, when generalizing the concept of strong equivalence, we will need to make it clear which "version" of stability we have in mind, that is, which approximating operators we use to define it.

We will now discuss the relevance of the approximation theory to studies of nonmonotonic logics, specifically on the case of logic programming. We will discuss other formalisms in a full version of the paper. We focus on the propositional case and assume an underlying language generated by a set At of propositional variables.

Each logic program P determines two operators: the one-step provability operator T_P of van Emden and Kowalski (van Emden & Kowalski 1976) and the *stable* operator GL_P of Gelfond and Lifschitz (Gelfond & Lifschitz 1988). Both operators are defined on the lattice L_{At} of all 2-valued interpretations of the set At. They are fundamental to logic programming since:

1. prefixpoints and fixpoints of the operator T_P are precisely models and supported models of P, respectively; and
2. fixpoints of the operator GL_P are precisely stable models of P.

These operators extend to the lattice of 4-valued interpretations of the Herbrand base of a logic program P, which use elements of the Belnap algebra as truth values. The resulting operators are the 4-valued one-step provability operator $\mathcal{T}_P$ (Fitting 1985; 1991) and the 4-valued stable operator Ψ'_P (Przymusinski 1990). Prefixpoints and fixpoints of $\mathcal{T}_P$ are precisely 4-valued models and 4-valued supported models of P. Moreover, $\mathcal{T}_P$ has the least fixpoint, which corresponds to the Kripke-Kleene (4-valued) model of P. Similarly, fixpoints of Ψ'_P are precisely 4-valued stable models of P and the least fixpoint of Ψ'_P, which is guaranteed to exist, corresponds to the well-founded model of P. In other words, all major semantics of logic programs can be described by means of fixpoints and prefixpoints of operators on lattices of interpretations.

The connection to the approximation theory becomes apparent when we note that the lattice L_{At} of 2-valued interpretations is complete and that the lattice of 4-valued interpretations is isomorphic to the product lattice L^2_{At} of the lattice L_{At}. Under this isomorphism, the 4-valued operator $\mathcal{T}_P$ can be viewed as an operator on the product lattice L^2_{At} and turns out to be an approximating operator for the operator T_P. Moreover, the operator Ψ'_P turns out to be the stable operator for $\mathcal{T}_P$. The key point is that properties of semantics of logic programs become special cases of general algebraic results of the approximation theory (Denecker, Marek, & Truszczyński 2000), concerning with operators on complete lattices, their approximating operators and the corresponding stable operators.

Equivalence of lattice operators

Our goal in this paper is to show that the concept of strong equivalence can be cast in the abstract algebraic setting of the approximation theory.

We start by defining the concept of an *extension* of an operator. Let P and Q be operators on a lattice L. An *extension* of P with R is an operator $P \vee R$ defined on L by setting

$$(P \vee R)(x) = P(x) \vee R(x),$$

for every $x \in L$. If we consider programs in terms of their one-step provability operators, this definition is a direct generalization of the concept of the union of two logic programs. Indeed, if P and R are logic programs, then one can show that $T_{P \cup R} = T_P \vee T_R$. We call R an *extending* operator and $P \vee R$ an *extension* of P with R.

Next, we note that if A_P and A_R are approximating operators for P and R respectively, then the operator $A_P \vee A_R$ is an approximating operator for $P \vee R$. That leads us to the following definition.

Definition 3 *Let P and Q be operators on a lattice L and let A_P and A_Q be their approximating operators, respectively. Programs P and Q are* strongly equivalent *with respect to (A_P, A_Q) if for every operator R and for every approximating operator A_R of R,*

$$St(P \vee R, A_P \vee A_R) = St(Q \vee R, A_Q \vee A_R).$$

In such case, we write $P \equiv_s Q \mod (A_P, A_Q)$.

Thus, given P and Q and their approximating operators A_P and A_Q, P and Q are strongly equivalent with respect to (A_P, A_Q) if extensions of P and Q with an operator R have the same stable fixpoints ($(A_P \vee A_R)$-fixpoints on the one side and $(A_Q \vee A_R)$-fixpoints on the other) for an arbitrary operator R and for an arbitrary approximating operator for R.

Let us consider this definition from the perspective of normal logic programs. Let P be a program. As we noted, P can be represented in algebraic terms by means of the operator T_P and its approximating operator $\mathcal{T}_P$. Strong equivalence of programs P and Q as defined in (Lifschitz, Pearce, & Valverde 2001) requires that for every program R stable models of $P \cup R$ and $Q \cup R$ be the same. In the language of operators, that condition can be expressed as follows: for every program R, $St(T_P \vee T_R, \mathcal{T}_P \vee \mathcal{T}_R) = St(T_Q \vee T_R, \mathcal{T}_Q \vee \mathcal{T}_R)$. It is now clear that our definition of strong equivalence requires more, namely it requires that we consider an arbitrary operator R as an extending operator and, *in addition*, that we consider an arbitrary approximating operator A_R for R (in the case of logic programming we only need to consider one approximating operator — $\mathcal{T}_P$). Nevertheless, later in the paper we will show that the defining condition for the strong equivalence can be weakened, and that when applied to logic programs it yield the same concept of the strong equivalence as the one defined in (Lifschitz, Pearce, & Valverde 2001).

A characterization of strong equivalence

In this section, we extend the characterization of strong equivalence of logic programs in terms of se-models (Turner 2001; 2003) to the case of operators.

A pair $(x, y) \in L^2$ is an *se-pair* for P with respect to A_P if

(SE1) $x \leq y$

(SE2) $P(y) \leq y$

(SE3) $A^1_P(x, y) \leq x$

We will denote the set of *se-pair* for P with respect to A_P by $SE(P, A_P)$.

Let us look at this definition from the logic programming perspective. Let P be a logic program. We mentioned earlier that semantics properties of P are captured by two operators on the complete lattice of subsets of the Herbrand base (lattice of 2-valued interpretations), with the inclusion as the lattice ordering relation. The two operators are the one-step provability operator T_P and its approximating operator $\mathcal{T}_P$. The following two properties are well known: a set of atoms Y is a model of a program P if and only if $T_P(Y) \subseteq Y$; and a set of atoms X is a model of the program P^Y if and only if $\mathcal{T}^1_P(X, Y) \subseteq Y$.

We now recall that an se-model of a program P is a pair (X, Y) of sets of atoms (interpretations) such that $X \subseteq Y$, Y is a model of P and X is a model of P^Y (Turner 2001). Thus, our comments above imply that a pair (X, Y) is an se-model according to (Turner 2001) if and only if (X, Y) is an se-pair for T_P with respect to $\mathcal{T}_P$. Consequently, se-pairs generalize se-models.

Theorem 2 *Let P and Q be operators on a lattice L and let A_P and A_Q be approximating operators for P and Q respectively. If $SE(P, A_P) = SE(Q, A_Q)$ then $P \equiv_s Q \mod (A_P, A_Q)$.*

The proof of this result depends on two lemmas.

Lemma 1 *If $SE(P, A_P) = SE(Q, A_Q)$, then $St(P, A_P) = St(Q, A_Q)$.*

Lemma 2 *For every operator R on a complete lattice L and for every approximating operator A_R for R,*

$$SE(P \vee R, A_P \vee A_R) = SE(P, A_P) \cap SE(R, A_R)$$

Proof of Theorem 2. Let R be an operator on L and let A_R be an approximating operator for R. Since $SE(P, A_P) = SE(Q, A_Q)$, by Lemma 2 it follows that $SE(P \vee R, A_P \vee A_R) = SE(Q \vee R, A_Q \vee A_R)$. Thus, by Lemma 1, $St(P \vee R, A_P \vee A_R) = St(Q \vee R, A_Q \vee A_R)$. □

We will now prove the converse statement to Theorem 2. In fact, we will prove a stronger statement by restricting the class of operators one needs to consider as expanding operators.

An operator R on a complete lattice L is *simple* if for some $x, y \in L$ such that $x \leq y$, we have

$$R(z) = \begin{cases} y & \text{if } x < z \\ x & \text{otherwise} \end{cases}$$

for every $z \in L$.

We note that constant operators are simple. Indeed, if w is the only value taken by an operator R, R is simple with $x = y = w$.

Moreover, every simple operator R is monotone. Indeed, let $x \leq y$ be two elements in L that define R (according to the formula given above). If $z_1 \leq z_2$ and $R(z_1) = x$, then $R(z_1) \leq R(z_2)$. If, on the other hand, $R(z_1) = y$ then $x < z_1$. Thus, $x < z_2$ and so, $R(z_1) = R(z_2)$. In each case, $R(z_1) \leq R(z_2)$.

In particular, R has the *canonical* approximating operator C_R which, we recall, satisfies $C_R(x, y) = (R(x), R(y))$.

Theorem 3 *Let P and Q be operators on a complete lattice L and let A_P and A_Q be approximating operators for P and Q, respectively. If for every simple operator R on L we have $St(P \vee R, A_P \vee C_R) = St(Q \vee R, A_Q \vee C_R)$, then $SE(P, A_P) = SE(Q, A_Q)$.*

As before, we need auxiliary results.

Lemma 3 *If $P(y) \leq y$ then $(y, y) \in SE(P, A_P)$ and $(lfp(A^1_P(\cdot, y)), y) \in SE(P, A_P)$.*

Lemma 4 *If for every constant operator R on a complete lattice L we have $St(P \vee R, A_P \vee C_R) = St(Q \vee R, A_Q \vee C_R)$, then for every $y \in L$, $P(y) \leq y$ if and only if $Q(y) \leq y$.*

Proof of Theorem 3. Let $(x, y) \in SE(P, A_P)$. It follows that $x \leq y$ and $P(y) \leq y$. By Lemma 4, $Q(y) \leq y$.

If $x = y$ then, by Lemma 3, $(x, y) \in SE(Q, A_Q)$. So, let us assume that $x < y$. Let R be an simple operator on L given by

$$R(z) = \begin{cases} y & \text{if } x < z \\ x & \text{otherwise.} \end{cases}$$

We observe that $A^1_Q(y, y) = Q(y) \leq y$ and, as $x < y$, that $C^1_R(y, y) = R(y) = y$. It follows that

$$y = A^1_Q(y, y) \vee C^1_R(y, y).$$

That is, y is a fixpoint of the operator $A^1_Q(\cdot, y) \vee C^1_R(\cdot, y)$.

Let z be an arbitrary fixpoint of $A^1_Q(\cdot, y) \vee C^1_R(\cdot, y)$, that is,

$$z = A^1_Q(z, y) \vee C^1_R(z, y).$$

It follows that $A^1_Q(z, y) \leq z$. In addition, $x \leq R(z) = C^1_R(z, y) \leq z$.

Let us assume that $x < z$. By the definition of R, $R(z) = y$ and so,

$$y = C^1_R(z, y) \leq z.$$

Thus, $y = lfp(A^1_Q(\cdot, y) \vee C^1_R(\cdot, y)$ and, consequently, $y = lfp(A^1_P(\cdot, y) \vee C^1_R(\cdot, y))$.

Since $A^1_P(x, y) \leq x$ and $C^1_R(x, y) = R(x) = x$, we have

$$A^1_P(x, y) \vee C^1_R(x, y) = x.$$

Thus, $y \leq x$, a contradiction.

As $x \leq z$, it follows that $x = z$. Thus, $A^1_Q(x, y) \leq x$ and so, $(x, y) \in SE(Q, A_Q)$. Consequently, $SE(P, A_P) \subseteq SE(Q, A_Q)$. The converse inclusion follows by the symmetry argument. □

Theorems 2 and 3 yield a complete characterization of the strong equivalence of operators.

Corollary 4 *Let P and Q be operators on a lattice L and let A_P and A_Q be approximating operators for P and Q respectively. Then $P \equiv_s Q \mod (A_P, A_Q)$ if and only if $SE(P, A_P) = SE(Q, A_Q)$.*

Theorems 2 and 3 imply also the following result, which shows that we can substantially weaken the defining condition for the strong equivalence without changing the concept.

Theorem 5 *Let P and Q be operators on a lattice L and let A_P and A_Q be approximating operators for P and Q respectively. Then $P \equiv_s Q \mod (A_P, A_Q)$ if and only if for every simple operator R, $St(P \vee R, A_P \vee C_R) = St(Q \vee R, A_Q \vee C_R)$.*

In other words, to determine strong equivalence of operators it suffices to consider simple operators as extending operators, and for each simple operator — to consider its canonical approximating operator only.

In the case of normal logic programs our approach to strong equivalence generalizes the one developed in (Lifschitz, Pearce, & Valverde 2001).

Theorem 6 *Normal logic programs P and Q are strongly equivalent in the sense of (Lifschitz, Pearce, & Valverde 2001) if and only if the operators T_P and T_Q are strongly equivalent with respect to $(\mathcal{T}_P, \mathcal{T}_Q)$ according to Definition 3.*

Discussion

In several places in this paper, we demonstrated that our approach yields as corollaries results concerning strong equivalence of logic programs. In the same way, it can also be applied to other nonmonotonic logics whose semantics can be described in algebraic terms of (Denecker, Marek, & Truszczyński 2000). Examples of such logics include some extensions of logic programming with aggregates (Pelov. 2004; Pelov, Denecker, & Bruynooghe 2004), as well as default and autoepistemic logic (the latter understood as presented in (Denecker, Marek, & Truszczyński 2003)). In a full version of the paper we will present details of some of these applications.

On the other hand, there are at present limits to the applicability of our approach. In particular, it does not apply to nested logic programs and default theories. The reason is that these formalisms do not have a satisfactory algebraic presentation such us their counterparts without nested expressions. Two key problems are minimality and nondeterminism. Some work on ways to develop an algebraic treatment of these two issues in the context of disjunctive logic programs can be found in (Pelov & Truszczyński 2004). However, more work is needed.

Finally, we note that the algebraic approach to strong equivalence developed here extends also to the case of uniform equivalence (Eiter & Fink 2003), a topic that we will discuss in detail in a full version of the paper.

Acknowledgments

We acknowledge the support of NSF grant IIS-0325063.

References

Denecker, M.; Marek, V.; and Truszczyński, M. 2000. Approximations, stable operators, well-founded fixpoints and applications in nonmonotonic reasoning. In Minker, J., ed., *Logic-Based Artificial Intelligence*. Kluwer Academic Publishers. 127–144.

Denecker, M.; Marek, V.; and Truszczyński, M. 2003. Uniform semantic treatment of default and autoepistemic logics. *Artificial Intelligence Journal* 143:79–122.

Eiter, T., and Fink, M. 2003. Uniform equivalence of logic programs under the stable model semantics. In *Proceedings of the 2003 International Conference on Logic Programming*, volume 2916 of *Lecture Notes in Computer Science*, 224–238. Berlin: Springer.

Fitting, M. C. 1985. A Kripke-Kleene semantics for logic programs. *Journal of Logic Programming* 2(4):295–312.

Fitting, M. 1991. Bilattices and the semantics of logic programming. *Journal of Logic Programming* 11:91–116.

Fitting, M. C. 2002. Fixpoint semantics for logic programming – a survey. *Theoretical Computer Science* 278:25–51.

Gelfond, M., and Lifschitz, V. 1988. The stable semantics for logic programs. In *Proceedings of the 5th International Conference on Logic Programming*, 1070–1080. MIT Press.

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

Ginsberg, M. 1988. Multivalued logics: a uniform approach to reasoning in artificial intelligence. *Computational Intelligence* 4:265–316.

Heyting, A. 1930. Die formalen regeln der intuitionistischen logik. *Sitzungsberichte der Preussischen Akademie von Wissenschaften. Physikalisch-mathematische KLasse* 42–56.

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

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

Lin, F. 2002. Reducing strong equivalence of logic programs to entailment in classical propositional logic. In *Principles of Knowledge Representation and Reasoning, Proceedings of the 8th International Conference (KR2002)*. Morgan Kaufmann Publishers.

Pelov, N., and Truszczyński, M. 2004. Semantics of disjunctive programs with monotone aggregates — an operator-based approach. In Delgrande, J., and Schaub, T., eds., *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning, NMR-04*, 327–334.

Pelov, N.; Denecker, M.; and Bruynooghe, M. 2004. Partial stable models for logic programs with aggregates. In Lifschitz, V., and Niemelä, I., eds., *Logic programming and Nonmonotonic Reasoning, Proceedings of the 7^{th} International Conference*, volume 2923, 207–219. Springer.

Pelov., N. 2004. Semantics of logic programs with aggregates. *PhD Thesis. Department of Computer Science, K.U.Leuven, Leuven, Belgium*.

Przymusinski, T. 1990. The well-founded semantics coincides with the three-valued stable semantics. *Fundamenta Informaticae* 13(4):445–464.

Tarski, A. 1955. Lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics* 5:285–309.

Turner, H. 2001. Strong equivalence for logic programs and default theories (made easy). In *Proceedings of Logic Programming and Nonmonotonic Reasoning Conference, LPNMR 2001*, volume 2173, 81–92. Lecture Notes in Artificial Intelligence, Springer.

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

van Emden, M., and Kowalski, R. 1976. The semantics of predicate logic as a programming language. *Journal of the ACM* 23(4):733–742.

Vennekens, J.; Gilis, D.; and Denecker, M. 2004a. Splitting an operator: An algebraic modularity result and its application to auto-epistemic logic. In Delgrande, J., and Schaub, T., eds., *Proceedings of the 10th International Workshop on Non-Monotonic Reasoning*, 400–408.

Vennekens, J.; Gilis, D.; and Denecker, M. 2004b. Splitting an operator: an algebraic modularity result and its applications to logic programming. In Lifschitz, V., and Demoen, B., eds., *Logic programming, Proceedings of the 20th International Conference on Logic Programming, ICLP-04*, 195–209.