Strong and uniform equivalence of nonmonotonic theories — an algebraic approach*

Mirosław Truszczyński
Department of Computer Science
University of Kentucky,
Lexington, KY 40506-0046, USA
mirekcs.uky.edu

Abstract

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

Introduction

In knowledge representation, as in programming in general, when building a knowledge base for a large application domain one of the key methodological principles is that of *modularity*. An application domain at hand is partitioned into smaller fragments and each of these fragments is represented as a separate module. Sometimes it becomes necessary to replace a module with another one, for instance, to optimize the performance of reasoning algorithms. However, it is paramount that the replacement leaves the overall meaning of the knowledge base unchanged. Thus, deciding when two modules are *equivalent for substitution* emerges as a fundamental problem in studies of knowledge representation formalisms.

In some cases, the answer is straightforward. If a knowledge base is represented as a theory in propositional logic, equivalence for substitution coincides with the standard logical equivalence. Indeed, if two propositional theories P and Q are logically equivalent then for every theory of the form $T = P \cup R$, the theory $T' = Q \cup R$, obtained by replacing P with Q in T, is logically equivalent to T. The converse statement holds, as well and so, theories P and Q are equivalent for substitution if and only if they are logically equivalent.

For knowledge representation formalisms based on nonmonotonic logics, the situation is more complex. In logic programming with the semantics of stable models (Gelfond & Lifschitz 1988), having the same stable models is too weak a requirement to guarantee equivalence for substitution. For instance, 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 $\{q\}$. Similarly, $P \cup \{q \leftarrow \mathbf{not}(p)\}$ has one stable model, $\{p\}$, and $Q \cup \{q \leftarrow \mathbf{not}(p)\}$ has two stable models $\{p\}$ and $\{q\}$.

Characterizing logic programs that are equivalent for substitution with respect to the stable-model semantics was identified as an important research topic in (Lifschitz, Pearce, & Valverde 2001). That paper used the term *strong equivalence* instead of *equivalence for substitution*. Since the former is prevalent, we use it in our paper, too.

(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). Nested logic programming 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 and obtained simple characterizations of strong equivalence without explicit references to the logic *here-and-there*. 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 (nested) default theories.

(Eiter & Fink 2003) introduced one more notion of equivalence, the *uniform equivalence* of disjunctive logic programs with answer-set semantics. Two disjunctive logic programs P and Q are *uniformly equivalent* if for every set R of *facts*, $P \cup R$ and $Q \cup R$ have the same answer sets. (Eiter & Fink 2003) presented a characterization of *uniform equivalence* in terms of se-models and, for finite programs, in terms of *ue-models*, which are se-models with some additional properties.

A comprehensive discussion of strong and uniform equivalence of disjunctive logic programs, including recent extensions of the two concepts to the setting relativized with re-

*This work was partially supported by the NSF grant IIS-0325063.

spect to a fixed set of atoms can be found in (Eiter, Fink, & Woltran 2006).

Results from (Lifschitz, Pearce, & Valverde 2001; Turner 2001; Lin 2002; Turner 2003; Eiter & Fink 2003) and their proofs exhibit common themes and similarities. To a large degree, it is due to the fact that all characterizations of strong and uniform equivalence developed there are rooted, if not directly then implicitly, in the logic *here-and-there*. In this paper we point out to an additional reason behind these similarities, related to the fact that semantics of many nonmonotonic logics can be introduced in abstract algebraic terms. Our main contribution is an algebraic account of strong and uniform equivalence in terms of operators on complete lattices. Specifically, in the paper we:

1. extend the definitions of strong and uniform equivalence of logic programs to the abstract case of operators on lattices.
2. establish characterizations of strong and uniform equivalence of operators in terms of *se-pairs* — objects that generalize se-models to the setting of lattices.
3. demonstrate that these characterizations yield, as corollaries, characterizations of strong and uniform equivalence for those nonmonotonic logics whose semantics can be defined in terms of fixpoints of operators on lattices[1].

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 logics in a uniform, general and abstract way (Denecker, Marek, & Truszczyński 2003). Recent 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).

Preliminaries

We start with an overview of elements of the approximation theory (Denecker, Marek, & Truszczyński 2000). We assume familiarity with the concepts of a lattice, lattice ordering $\leq$, and lattice operations $\wedge$ and $\vee$. 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 $\perp$, 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 O 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.*

The approximation theory (Denecker, Marek, & Truszczyński 2000) is concerned with operators on lattices and mappings from L^2 to L. We emphasize that we consistently use the term *mapping* for functions from L^2 to L, and reserve the term *operator* for functions whose domains and co-domains coincide.

Definition 1 *Let L be a complete lattice. A mapping $A\colon L^2 \to L$ is an* approximating mapping *if for every $x \in L$, the operator $A(\cdot, x)$ is monotone and the operator $A(x, \cdot)$ is antimonotone* [2].

If O is an operator on L such that $O(x) = A(x, x)$, then A is an approximating mapping for O.

If $x, y, z \in L$ satisfy $x \leq z \leq y$, then we say that the pair (x, y) is an *approximation* of z. If A is an approximating mapping for an operator O on L and (x, y) is an approximation to z then

$$A(x, z) \leq A(z, z) \leq A(y, z)$$

and

$$A(z, y) \leq A(z, z) \leq A(z, x).$$

The first group of inequalities follows by the monotonicity of $A(\cdot, z)$, the other one by the antimonotonicity of $A(z, \cdot)$. Consequently, we have

$$A(x, z) \leq O(z) \leq A(y, z)$$

and

$$A(z, y) \leq O(z) \leq A(z, x),$$

that is, pairs $(A(x, z), A(y, z))$ and $(A(z, y), A(z, x))$ *approximate* $O(z)$. This property motivates the name "approximating mapping" for A.

Every operator O on a lattice L has an approximating mapping. Indeed, let $A\colon L^2 \to L$ be a mapping defined by:

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

Clearly, for every $x \in L$, $A(x, x) = O(x)$. Next, let $x_1, x_2, y \in L$ and let $x_1 \leq x_2$. If $x_1 < y$, $A(x_1, y) = \perp$. If it is not the case that $x_2 \leq y$, $A(x_2, y) = \top$. If neither of these two cases holds, $x_1 = x_2 = y$. In all cases, $A(x_1, y) \leq A(x_2, y)$, that is, for every $y \in L$, $A(\cdot, y)$ is

[1] In this paper, we mention only applications to logic programming and default logic.

[2] The set L^2 with the *precision* ordering is a complete lattice (Ginsberg 1988; Fitting 2002). (Denecker, Marek, & Truszczyński 2000) developed the approximation theory in terms of the so-called approximating *operators* on the lattice L^2. Approximating *mappings* lead to a simpler notation and so, we chose to use them in this paper.

monotone. In a similar way we verify that for every $x \in L$, $A(x, \cdot)$ is antimonotone. Thus, A is an approximating mapping for O.

In general, approximating mappings are not unique. For monotone and antimonotone operators we distinguish special approximating mappings. Namely, if O is monotone, we set $C_O(x, y) = O(x)$, for $x, y \in L$. If O is antimonotone, we set $C_O(x, y) = O(y)$, for $x, y \in L$. In each case, one can verify that C_O is an approximating mapping for O — we call it *canonical*.

If A is an approximating mapping for some operator O on a complete lattice, then Theorem 1 ensures that for every $y \in L$, $lfp(A(\cdot, y))$ is well defined. This property makes the following definition sound.

Definition 2 (Denecker, Marek, & Truszczyński 2000) *Let O be an operator on a complete lattice L and let A be an approximating mapping for O. An A-stable operator for O on L is an operator S_A on L such that for every $y \in L$:*

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

An element $x \in L$ is an A-stable fixpoint of O if $x = S_A(x)$. We denote the set of A-stable fixpoints of O by $St(O, A_O)$.

We will now discuss the relevance of the approximation theory to nonmonotonic logics. We focus on logic programming, consider the propositional case only, and assume that an underlying language is generated by a set At of propositional variables. We represent 2-valued interpretations of At as subsets of At. With the inclusion relation as an ordering relation, the set of 2-valued interpretations of At, denoted by L_{At}, forms a complete lattice $\langle L_{At}, \subseteq\rangle$. The set union operator $\cup$ is the join operator in this lattice.

Each logic program P determines a one-input one-step provability operator T_P on the lattice L_{At} (van Emden & Kowalski 1976). Let $I \subseteq At$. We recall that $T_P(I)$ is the set of the heads of all rules in P whose body holds in I. Another operator associated with P is a two-input one-step provability operator Ψ_P (Fitting 1985; 1991). If $I, J \subseteq At$ then $\Psi_P(I, J)$ consists of the heads of those rules whose positive body holds in I and negative body holds in J. One can check that for every $I \subseteq At$, $\Psi_P(\cdot, I)$ is monotone, $\Psi_P(I, \cdot)$ is antimonotone and $\Psi_P(I, I) = T_P(I)$. It follows that Ψ_P is an approximating mapping for T_P. Thus, as long as we view logic programming as a study of properties of T_P and Ψ_P, it is a special case of the approximation theory.

The operators T_P and Ψ_P are fundamental to the study of semantics of logic programs. Fixpoints of T_P are precisely supported models of P, and 4-valued supported models of P (including the Kripke-Kleene model of P) are determined by pairs (I, J) of interpretations such that $(I, J) = (\Psi_P(I, J), \Psi_P(J, I))$. Next, the Gelfond-Lifschitz operator GL_P (Gelfond & Lifschitz 1988), satisfies $GL_P(I) = lfp(\Psi_P(\cdot, I))$. Thus, GL_P is the Ψ_P-stable operator for T_P and so, stable models of P coincide with Ψ_P-stable fixpoints of T_P. Since, 4-valued stable models (including the well-founded model of P) can be characterized by pairs (I, J) of interpretations such that $(I, J) = (GL_P(J), GL_P(I))$, it follows that all major 2-valued and 4-valued semantics of logic programs can be expressed as fixpoints of operators related to T_P and Ψ_P. The key point is that semantics of logic programs are special cases of a general algebraic theory of operators and their fixpoints (Denecker, Marek, & Truszczyński 2000).

Equivalence of lattice operators

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

As in the case of logic programs, strong and uniform equivalence of operators concerns stable fixpoints of their extensions. However, the notion of a stable fixpoint depends on the choice of an approximating mapping. Thus, whenever we consider the equivalence of two operators P and Q, we select for each of them one of their approximating mappings, say A_P and A_Q respectively. In this way, we determine a specific notion of stability for the operators P and Q.

The equivalence of P and Q will depend on stable fixpoints of the operators $P \vee R$ and $Q \vee R$. Informally, we will require that $P \vee R$ and $Q \vee R$ have the same stable fixpoints. However, the concept of stability becomes unambiguous only if $P \vee R$ and $Q \vee R$ are assigned some approximating mappings. These approximating mappings should depend in some way on the approximating mappings of P (Q, respectively) and R, as otherwise there would be no connection between the concepts of stability for P and $P \vee R$ (Q and $Q \vee R$, respectively).

We will now consider this issue. Let P and R be operators on a lattice L, and let A_P and A_R be approximating mappings for P and R, respectively. It is straightforward to check that the operator $A_P \vee A_R$ is an approximating mapping for the operator $P \vee R$. Thus, when considering operators $P \vee R$ and $Q \vee R$, we will use $A_P \vee A_R$ and $A_Q \vee A_R$ as their approximating mappings. In particular, we will compare $(A_P \vee A_R)$-stable fixpoints of $P \vee R$ with $(A_Q \vee A_R)$-stable fixpoints of $Q \vee R$.

Another point concerns operators to use to extend P and Q with. As in logic programming, we impose no restrictions when defining strong equivalence. To properly generalize the concept of uniform equivalence, we note that logic programs consisting of facts (this class of programs was used to define the uniform equivalence in the case of logic programming), have constant one-step provability operators. Therefore, we define uniform equivalence of operators with respect to extensions by constant operators only. Moreover, we consider them only together with their canonical approximations (we recall that constant operators are monotone and have canonical approximating mappings). We formalize this discussion in 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 mappings, respectively.*

1. *P and Q are* strongly equivalent *with respect to (A_P, A_Q), written $P \equiv_s Q \mod (A_P, A_Q)$, if for every operator R and for every approximating mapping A_R of R,*
$$St(P \vee R, A_P \vee A_R) = St(Q \vee R, A_Q \vee A_R).$$
2. *P and Q are* uniformly equivalent *with respect to (A_P, A_Q), written $P \equiv_u Q \mod (A_P, A_Q)$, if for every* constant *operator R*
$$St(P \vee R, A_P \vee C_R) = St(Q \vee R, A_Q \vee C_R),$$
where C_R is the canonical approximating mapping for R (constant operators are monotone and have canonical approximating mappings).

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

Let us consider these definitions 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 mapping Ψ_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, $St(T_P \cup T_R, \Psi_P \cup \Psi_R) = St(T_Q \cup T_R, \Psi_Q \cup \Psi_R)$. 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*, an arbitrary approximating mapping A_R for R, while in the case of logic programming we only need to consider one approximating mapping — Ψ_P. Nevertheless, later in the paper we will show that our definition of strong equivalence, when applied to logic programs yields the same concept of the strong equivalence as the one defined in (Lifschitz, Pearce, & Valverde 2001).

As concerns the concept of uniform equivalence, the situation is simpler. Uniform equivalence of two programs P and Q, as introduced by (Eiter & Fink 2003), requires that for every set R of atoms, stable models of $P \cup R$ coincide with stable models of $Q \cup R$. In the language of operators, this defining condition can be expressed as follows: for every set of facts R, $St(T_P \cup T_R, \Psi_P \cup \Psi_R) = St(T_Q \cup T_R, \Psi_Q \cup \Psi_R)$. We now note that if R is a set of facts, T_R is a constant operator and $\Psi_R(X, Y) = T_R(X)$. Thus, $\Psi_R = C_R$. Consequently, our definition of uniform equivalence is a direct generalization of the definition in (Eiter & Fink 2003).

Se-pairs

In this section, we generalize the notion of an se-model (Turner 2001; 2003) to the case of operators.

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

(SE1) $x \leq y$

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

(SE3) $A_P(x, y) \leq x$

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

Let us consider this definition from the logic programming perspective. Let P be a logic program. We observed earlier that semantics of P are captured by the operators T_P and Ψ_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 $\Psi_P(X, Y) \subseteq X$.

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 Ψ_P. Consequently, se-pairs generalize se-models.

In the next two sections we will develop characterizations of strong and uniform equivalence in terms of se-pairs and we will show that our characterizations generalize the results from (Turner 2001) and (Eiter & Fink 2003).

Strong equivalence

In this section we study the case of strong equivalence, obtain a characterization of this concept, and show that one can substantially weaken the defining condition of strong equivalence.

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

To prove Theorem 2 we will first state and prove some auxiliary results.

Lemma 1 *Let P be an operator on a lattice L and let A_P be an approximating mapping for P. If $P(y) \leq y$ then $(y, y) \in SE(P, A_P)$ and $(lfp(A_P(\cdot, y)), y) \in SE(P, A_P)$.*

Proof: The pair (y, y) satisfies the conditions (SE1) and (SE2). Since A_P is an approximating mapping for P, $A_P(y, y) = P(y)$. Thus, the pair (y, y) satisfies the condition (3), as well. It follows that $(y, y) \in SE(P, A_P)$.

Let us denote $y' = lfp(A_P(\cdot, y))$ (we recall that $A_P(\cdot, y)$ is monotone and so, it has a least fixpoint). Since $A_P(y, y) = P(y) \leq y$, y is a prefixpoint of the operator $A_P(\cdot, y)$. By Theorem 1, y' is also the least prefixpoint of $A_P(\cdot, y)$. Thus, $y' \leq y$ and the pair (y', y) satisfies the condition (SE1). The condition (SE2) holds by the assumption. Finally, since y' is a fixpoint of $A_P(\cdot, y)$, we have $A_P(y', y) = y'$. Thus, the condition (SE3) holds for (y', y), as well. Consequently, $(y', y) \in SE(P, A_P)$. $\square$

Lemma 2 *Let P and Q be operators on a lattice L and let A_P and A_Q be approximating mapping for P and Q, respectively. If $SE(P, A_P) = SE(Q, A_Q)$, then $St(P, A_P) = St(Q, A_Q)$.*

Proof: Let $y \in St(P, A_P)$. By the definition, we have $y = lfp(A_P(\cdot, y))$. It follows that $A_P(y, y) = y$ and so, $P(y) = A_P(y, y) = y$. Thus, by Lemma 1, $(y, y) \in SE(P, A_P)$ and so, $(y, y) \in SE(Q, A_Q)$. In particular, it follows that $Q(y) \leq y$.

Let $y' = lfp(A_Q(\cdot, y))$. Since $Q(y) \leq y$, Lemma 1 implies that $(y', y) \in SE(Q, A_Q)$. Thus, $(y', y) \in SE(P, A_P)$ and, by (SE3), y' is a prefixpoint of the operator $A_P(\cdot, y)$. Consequently, $y \leq y'$ (by Theorem 1, being the least fixpoint of $A_P(\cdot, y)$, y is also the least prefixpoint of $A_P(\cdot, y)$).

Since $(y', y) \in SE(Q, A_Q)$, $y' \leq y$. Consequently, $y = y'$ and so, $y = lfp(A_Q(\cdot, y))$. Therefore, $y \in St(Q, A_Q)$. It follows that $St(P, A_P) \subseteq St(Q, A_Q)$. The converse inclusion follows by the symmetry. $\square$

Lemma 3 *Let P be an operator on a lattice L and let A_P be an approximating mapping for P. For every operator R on a complete lattice L and for every approximating mapping A_R for R,*

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

Proof: If $(x, y) \in SE(P \vee R, A_P \vee A_R)$ or $(x, y) \in SE(P, A_P) \cap SE(R, A_R)$ then $x \leq y$. Moreover, $(P \vee R)(y) \leq y$ if and only if $P(y) \leq y$ and $R(y) \leq y$. Finally, $(A_P \vee A_R)(x, y) \leq x$ if and only if $A_P(x, y) \leq x$ and $A_R(x, y) \leq x$. Thus, $(x, y) \in SE(P \vee R, A_P \vee A_R)$ if and only if $(x, y) \in SE(P, A_P) \cap SE(R, A_R)$. $\square$

Proof of Theorem 2. Let R be an operator on L and let A_R be an approximating mapping for R. Since $SE(P, A_P) = SE(Q, A_Q)$, by Lemma 3 it follows that $SE(P \vee R, A_P \vee A_R) = SE(Q \vee R, A_Q \vee A_R)$. Thus, by Lemma 2, $St(P \vee R, A_P \vee A_R) = St(Q \vee R, A_Q \vee A_R)$, and the assertion follows. $\square$

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

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

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 $R(z_2) = y$, then $R(z_1) \leq R(z_2)$ (as $R(z_1) = x$ or y, and $x \leq y$). If, on the other hand, $R(z_2) = x$ then $z_2 \leq x$. Thus, $z_1 \leq x$, too, and $R(z_1) = x$. In each case, $R(z_1) \leq R(z_2)$.

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

Theorem 3 *Let P and Q be operators on a complete lattice L and let A_P and A_Q be approximating mappings 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 will first state and prove an auxiliary result.

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: Let $y \in L$ and let us assume that $P(y) \leq y$. We define R by setting $R(z) = y$, for every $z \in L$. Thus, R is a constant operator on L.

We note that $A_P(y, y) = P(y) \leq y$. Moreover, $C_R(y, y) = R(y) = y$. Thus, $y = A_P(y, y) \vee C_R(y, y)$ or, in other words, y is a fixpoint of $A_P(\cdot, y) \vee C_R(\cdot, y)$.

Let $z \in L$ be an arbitrary fixpoint of $A_P(\cdot, y) \vee C_R(\cdot, y)$. Then

$$\begin{aligned} z &= A_P(z, y) \vee C_R(z, y) = A_P(z, y) \vee R(z) \\ &= A_P(z, y) \vee y \geq y. \end{aligned}$$

It follows that $y = lfp(A_P(\cdot, y) \vee C_R(\cdot, y))$ and so, $y \in St(P \vee R, A_P \vee C_R)$. By the assumption of Lemma 4, $y \in St(Q \vee R, A_Q \vee C_R)$, that is, $y = lfp(A_Q(\cdot, y) \vee C_R(\cdot, y))$. In particular, it follows that $y = A_Q(y, y) \vee C_R(y, y) = Q(y) \vee y$. Thus, $Q(y) \leq y$. The converse implication follows by the symmetry argument. $\square$

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

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

We observe that $A_Q(y, y) = Q(y) \leq y$ and, as $x < y$, that $C_R(y, y) = R(y) = y$. It follows that

$$y = A_Q(y, y) \vee C_R(y, y).$$

That is, y is a fixpoint of the operator $A_Q(\cdot, y) \vee C_R(\cdot, y)$.

Let z be an arbitrary fixpoint of $A_Q(\cdot, y) \vee C_R(\cdot, y)$, that is,

$$z = A_Q(z, y) \vee C_R(z, y).$$

It follows that $A_Q(z, y) \leq z$. In addition, $x \leq R(z) = C_R(z, y) \leq z$.

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

$$y = R(z) = C_R(z, y) \leq z.$$

Thus, $y = \mathit{lfp}(A_Q(\cdot, y) \vee C_R(\cdot, y))$ and so, $y \in St(Q \vee R, A_Q \vee C_R)$. By the assumption, $y \in St(P \vee R, A_P \vee C_R)$ and so, $y = \mathit{lfp}(A_P(\cdot, y) \vee C_R(\cdot, y))$.

Since $A_P(x, y) \leq x$ and $C_R(x, y) = R(x) = x$, we have

$$A_P(x, y) \vee C_R(x, y) = x.$$

Thus, we have that x is a fixpoint of $A_P(\cdot, y) \vee C_R(\cdot, y)$ and y is the least fixpoint of $A_P(\cdot, y) \vee C_R(\cdot, y)$. It follows that $y \leq x$, a contradiction.

As $x \leq z$, we have then that $x = z$. Thus, $A_Q(x, y) \leq x$ 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 mappings 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 also imply a result stating that when establishing strong equivalence it suffices to consider extensions by simple operators, and for each simple operator — to consider its canonical approximating mapping only. Thus, the defining condition of strong equivalence can be weakened significantly.

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

We will now show formally that 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 (Ψ_P, Ψ_Q) according to Definition 3.*

Proof: The lattice of interest here is $\langle L_{At}, \subseteq \rangle$, in which the join operator is $\cup$.
($\Leftarrow$) Let R be an arbitrary logic program. Since P and Q are strongly equivalent according to Definition 3,

$$St(T_P \cup T_R, \Psi_P \cup \Psi_R) = St(T_Q \cup T_R, \Psi_Q \cup \Psi_R).$$

As we noted earlier, the sets of stable models of $P \cup R$ and $Q \cup R$ are given by the left-hand side and the right-hand side, respectively, of the equality above. Thus, P and Q are strongly equivalent according to the definition in (Lifschitz, Pearce, & Valverde 2001).
($\Rightarrow$) Let S be an arbitrary simple operator on the lattice L_{At}. Then there are sets $X, Y \subseteq At$ such that $X \subseteq Y$ and, for every $Z \subseteq At$,

$$S(Z) = \begin{cases} X & \text{if } Z \subseteq X \\ Y & \text{otherwise.} \end{cases}$$

Let R be a logic program defined as follows:

$$R = X \cup \{a \leftarrow b\colon a \in Y,\ b \in At \setminus X\}.$$

It is easy to check that $S = T_R$.

Since P and Q are strongly equivalent in the sense of (Lifschitz, Pearce, & Valverde 2001), $P \cup R$ and $Q \cup R$ have the same stable models. In the language of operators, it means that $St(T_P \cup T_R, \Psi_P \cup \Psi_R) = St(T_Q \cup T_R, \Psi_Q \cup \Psi_R)$. The program R is a Horn program. Thus, $\Psi_R(V, W) = \Psi_R(V, V) = T_R(V) = S(V) = C_S(V, W)$. It follows that $St(T_P \cup S, \Psi_P \cup C_S) = St(T_Q \cup S, \Psi_Q \cup C_S)$. By Theorem 5, P and Q are strongly equivalent according to Definition 3. □

Uniform equivalence

Se-pairs can also be used to characterize uniform equivalence. We have the following theorem.

Theorem 7 *Let P and Q be operators on a complete lattice L and let A_P and A_Q be approximating mappings for P and Q respectively. Then $P \equiv_u Q \mod (A_P, A_Q)$ if and only if*

1. *for every $y \in L$, $P(y) \leq y$ if and only if $Q(y) \leq y$*
2. *for every $x, y \in L$ such that $x < y$ and $(x, y) \in SE(P, A_P)$, there is $u \in L$ such that $x \leq u < y$ and $(u, y) \in SE(Q, A_Q)$*
3. *for every $x, y \in L$ such that $x < y$ and $(x, y) \in SE(Q, A_Q)$, there is $u \in L$ such that $x \leq u < y$ and $(u, y) \in SE(P, A_P)$*

Proof: ($\Leftarrow$) Let R be a constant operator, say $R(z) = x$ for every $z \in L$. Let $y \in St(P \vee R, A_P \vee C_R)$. Then $y = \mathit{lfp}(A_P(\cdot, y) \vee C_R(\cdot, y))$. It follows that

$$C_R(y, y) \leq y \ \text{ and } \ P(y) = A_P(y, y) \leq y.$$

Since $P(y) \leq y$, the condition (1) implies that $Q(y) \leq y$. Thus, $A_Q(y, y) = Q(y) \leq y$. Since $C_R(y, y) \leq y$, we obtain that y is a prefixpoint of $A_Q(\cdot, y) \vee C_R(\cdot, y)$.

Let $y' = \mathit{lfp}(A_Q(\cdot, y) \vee C_R(\cdot, y))$. Therefore, we have

$$y' \leq y$$

and

$$x = R(y') = C_R(y', y) \leq y'.$$

Let us assume that $y' < y$. Since $y' = A_Q(y', y) \vee C_R(y', y)$, $A_Q(y', y) \leq y'$. Thus, $(y', y) \in SE(Q, A_Q)$ (we already proved that $y' \leq y$ and $Q(y) \leq y$). By the condition (3), there is y'' such that $y' \leq y'' < y$ and $(y'', y) \in SE(P, A_P)$. In particular, $A_P(y'', y) \leq y''$. In addition, we have

$$C_R(y'', y) = R(y'') = x \leq y' \leq y''.$$

It follows that y'' is a prefixpoint of the operator $A_P(\cdot, y) \vee C_R(\cdot, y)$ and so, $y \leq y''$, a contradiction (we recall that y is the least fixpoint of $A_P(\cdot, y) \vee C_R(\cdot, y)$).

Thus, $y' = y$ and so, $y = \mathit{lfp}(A_Q(\cdot, y) \vee C_R(\cdot, y))$. It follows that $y \in St(Q \vee R, A_Q \vee C_R)$. We conclude that $St(P \vee R, A_P \vee C_R) \subseteq St(Q \vee R, A_Q \vee A_R)$. The converse inclusion follows by the symmetry argument. Thus, $P \equiv_u Q \mod (A_P, A_Q)$.

($\Rightarrow$) The condition (1) follows from Lemma 4. We will now show that the condition (2) holds. Let $x, y \in L$ be such that $x < y$ and $(x, y) \in SE(P, A_P)$. The latter assumption implies that $P(y) \leq y$. By the condition (1), $Q(y) \leq y$.

Let R be an operator on L such that for every $z \in L$, $R(z) = x$. Let $y' = lfp(A_Q(\cdot, y) \vee C_R(\cdot, y))$. Since $A_Q(y, y) = Q(y) \leq y$ and $C_R(y, y) = R(y) = x < y$, $A_Q(y, y) \vee C_R(y, y) \leq y$. Thus, y is a prefixpoint of $A_Q(\cdot, y) \vee C_R(\cdot, y)$ and so, $y' \leq y$.

If $y' = y$ then, $y = lfp(A_Q(\cdot, y) \vee C_R(\cdot, y))$ and, consequently, $y = lfp(A_P(\cdot, y) \vee C_R(\cdot, y))$. Since $(x, y) \in SE(P, A_P)$, $A_P(x, y) \leq x$ and so, $A_P(x, y) \vee C_R(x, y) \leq x$ (as $C_R(x, y) = R(x) = x$). Thus, x is a prefixpoint of $A_P(\cdot, y) \vee C_R(\cdot, y)$. Consequently, $y \leq x$, a contradiction. Thus, $y' < y$.

By the definition of y', $A_Q(y', y) \leq y'$. Thus, $(y', y) \in SE(Q, A_Q)$. The definition of y' also implies that $x = R(y') = C_R(y', y) \leq y'$. Thus, the condition (2) holds (for $u = y'$). The condition (3) follows by the symmetry argument. □

In the case, when a lattice L has the property that its every nonempty subset has maximal elements (in particular, every finite lattice has this property) we have a more elegant characterization of uniform equivalence.

An se-pair $(x, y) \in SE(P, A_P)$ is a *ue-pair* for P with respect to A_P if for every $(x', y) \in SE(P, A_P)$ such that $x < x'$, we have $x' = y$. We write $UE(P, A_P)$ for the set of all ue-pairs for P with respect to A_P.

Theorem 8 *Let L be a complete lattice with the property that its every subset has a maximal element. Let P and Q be operators on L and let A_P and A_Q be approximating mappings for P and Q respectively. Then $P \equiv_u Q \mod (A_P, A_Q)$ if and only if $UE(P, A_P) = UE(Q, A_Q)$.*

Proof: First, it is easy to show that $(y, y) \in UE(P, A_P)$ if and only if $(y, y) \in UE(Q, A_Q)$.

($\Rightarrow$) Let us assume that $UE(P, A_P) \neq UE(Q, A_Q)$, that is,

$$U = (UE(P, A_P) \setminus UE(Q, A_Q)) \cup (UE(P, A_P) \setminus UE(Q, A_Q)) \neq \emptyset.$$

Let X consist of all elements $x \in L$ such that for some $y \in L$, $(x, y) \in U$. Since $X \neq \emptyset$, X has a maximal element, say x_0. Let y_0 be an element of L such that $(x_0, y_0) \in U$. Without the loss of generality, we may assume that $(x_0, y_0) \in UE(P, A_P) \setminus UE(Q, A_Q)$. By our observation above, $x_0 \neq y_0$ and so, $x_0 < y_0$.

Since $P \equiv_u Q \mod (A_P, A_Q)$ and since $(x_0, y_0) \in UE(P, A_P) \subseteq SE(P, AP)$, by Theorem 7 there is $u \in L$ such that $x_0 \leq u < y_0$ and $(u, y_0) \in SE(Q, A_Q)$. Let u' be a maximal such element u (its existence follows from our assumption about the lattice L). Then $(u', y_0) \in UE(Q, A_Q)$. Since $(x_0, y_0) \notin UE(Q, A_Q)$, $u' \neq x_0$. Thus, $x_0 < u'$. From the way we chose x_0 it follows that $(u', y_0) \in UE(P, A_P)$ and so $(u', y_0) \in SE(P, A_P)$. Since $x_0 < u' < y_0$, this is a contradiction with the property that $(x_0, y_0) \in UE(P, A_P)$.

($\Leftarrow$) We first show that the condition (1) of Theorem 7 holds. If $P(y) \leq y$ then, by Lemma 1, $(y, y) \in SE(P, A_P)$. It follows that $(y, y) \in UE(P, A_P)$ and so, $(y, y) \in UE(Q, A_Q)$. In particular, we have that $Q(y) \leq y$. The proof of the converse implication is symmetric.

To prove the condition (2) of Theorem 7, let us consider $(x, y) \in SE(P, A_P)$ and such that $x < y$. Let y' be a maximal element such that $(y', y) \in SE(P, A_P)$ and $x \leq y' < y$. It follows that $(y', y) \in UE(P, A_P)$. Consequently, $(y', y) \in UE(Q, A_Q) \subseteq SE(Q, A_Q)$.

The condition (3) of Theorem 7 follows by symmetry. Thus, by Theorem 7, $P \equiv_u Q \mod (A_P, A_Q)$. □

We conclude this section by a result showing that in the case of normal logic programs, our notion of uniform equivalence generalizes that of (Eiter & Fink 2003). The result follows directly from the two corresponding definitions, when we (1) connect programs with their one-step provability operators, (2) take into account that every constant operator S on the lattice L_{At} is of the form T_R, where R is a set of atoms (facts) from At, and (3) observe that $\Psi_R = C_S$.

Theorem 9 *Normal logic programs P and Q are uniformly equivalent in the sense of (Eiter & Fink 2003) if and only if the operators T_P and T_Q are uniformly equivalent with respect to (Ψ_P, Ψ_Q) according to Definition 3.*

Other results

In this section, we present results on strong and uniform equivalence of monotone and antimonotone operators. We start with a lemma that characterizes se-pairs of a monotone operator with respect to its canonical approximating mapping.

Lemma 5 *Let P be a monotone operator on a complete lattice L. Then $SE(P, C_P) = \{(x, y) \in L^2 \colon x \leq y,\ P(y) \leq y, \text{ and } P(x) \leq x\}$.*

Proof. By the definition,

$$SE(P, C_P) = \{(x, y) \in L^2 \colon x \leq y,\ P(y) \leq y, \text{ and } C_P(x, y) \leq x\}.$$

We have $C_P(x, y) = P(x)$. Thus, the assertion follows. □

Theorem 10 *Let P and Q be monotone operators on lattice L. Then $P \equiv_s Q \mod (C_P, C_Q)$ if and only if P and Q have the same prefixpoints.*

Proof: From Lemma 5 it follows that if P and Q have the same prefixpoints then $SE(P, C_P) = SE(Q, C_Q)$ and so, $P \equiv_s Q \mod (C_P, C_Q)$.

For the converse implication, let us assume that $P \equiv_s Q \mod (C_P, C_Q)$. It follows that $SE(P, C_P) = SE(Q, C_Q)$. Since $(y, y) \in SE(P, C_P)$ ($(y, y) \in SE(Q, C_Q)$, respectively) if and only if $P(y) \leq y$ ($Q(y) \leq y$, respectively), the assertion follows. □

Corollary 11 *Let P and Q be monotone operators on a complete lattice L. Then $P \equiv_u Q \mod (C_P, C_Q)$ if and only if $P \equiv_s Q \mod (C_P, C_Q)$.*

Proof: Strong equivalence implies uniform equivalence. Thus, let us assume that $P \equiv_u Q \mod (C_P, C_Q)$. By Theorem 7, for every $z \in L$, $P(z) \leq z$ if and only if $Q(z) \leq z$.

That is, P and Q have the same prefixpoints. By Theorem 10, $P \equiv_s Q \mod (C_P, C_Q)$. □

If P is a Horn program then T_P is monotone and $\Psi_P = C_P$. Moreover, prefixpoints if T_P are precisely models of P. Thus, Theorem 10 and Corollary 11 imply results on strong and uniform equivalence of Horn programs (cf. (Eiter, Fink, & Woltran 2006)).

Corollary 12 *Let P and Q be Horn programs. Then the following conditions are equivalent:*

1. *P and Q are strongly equivalent*
2. *P and Q are uniformly equivalent*
3. *P and Q have the same models.*

For antimonotone operators we only have a simple characterization of strong equivalence.

Theorem 13 *Let P and Q be antimonotone operators on a complete lattice L. Then $P \equiv_s Q \mod (C_P, C_Q)$ if and only if P and Q have the same prefixpoints and for every prefixpoint y of both P and Q, $P(y) = Q(y)$.*

Proof: $(\Rightarrow)$ Let $P(y) \leq y$. Since $C_P(P(y), y) = P(y)$, $(P(y), y) \in SE(P, A_P)$. Thus, $(P(y), y) \in SE(Q, C_Q)$. That is, $Q(y) = C_Q(P(y), y) \leq P(y) \leq y$. It follows that y is a prefixpoint of Q and that $Q(y) \leq P(y)$. By the symmetry argument, if $Q(y) \leq y$, then $P(y) \leq y$ and $P(y) \leq Q(y)$. Thus, the assertion follows.

$(\Leftarrow)$ We have that $(x, y) \in SE(P, C_P)$ if and only if $P(y) \leq x \leq y$. This is equivalent to $Q(y) \leq x \leq y$ and, further, to $(x, y) \in SE(Q, C_Q)$. Thus, $SE(P, C_P) = SE(Q, C_Q)$ and so, P and Q are strongly equivalent. □

This result implies a corollary for logic programs that are purely negative (no rule has a positive literal in the body).

Corollary 14 *Let P and Q be purely negative logic programs. Then P and Q are strongly equivalent if and only if P and Q have the same models and for every model M of both P and Q, the sets of heads of M-applicable rules in P and Q are the same.*

Default logic

We will now apply the results of this paper to default logic (Reiter 1980). Let At be a set of propositional variables. By $\mathcal{F}_{At}$ we denote the set of all propositional formulas over At and by $\mathcal{P}(\mathcal{F}_{At})$ — the family of all subsets of $\mathcal{F}_{At}$. Together with the inclusion relation, $\mathcal{P}(\mathcal{F}_{At})$ forms a complete lattice. The operator $\cup$ is the join in this lattice.

In our presentation, we will assume familiarity with basic concepts of default logic and refer to (Marek & Truszczyński 1993) for details. We recall that a *default* is an expression d of the form

$$d = \frac{\alpha\colon \beta_1, \ldots, \beta_n}{\gamma},$$

where α, β_i, $1 \leq i \leq n$, and γ are formulas from $\mathcal{F}_{At}$ called the *prerequisite*, the *justifications* and the *consequent* of d, respectively. We set $pre(d) = \alpha$, $just(d) = \{\beta_1, \ldots, \beta_n\}$ and $cons(d) = \gamma$.

A *default theory* is a pair (D, W), where D is a set of defaults and $W \subseteq \mathcal{F}_{At}$. A key notion associated with default theories is that of an *extension* (Reiter 1980). We will now present a definition of an extension. It is a reformulation of the original definition to make it better aligned with the abstract theory of equivalence.

Let $U, V \subseteq \mathcal{F}_{At}$ and let d be a default. We say that (U, V) *enables* d, written $(U, V) \rhd d$, if $U \models pre(d)$ and, for every $\beta \in just(d)$, $V \not\models \neg\beta$. Let $\Delta = (D, W)$ be a default theory. We now define a *2-input one-step provability* mapping

$$\Psi_\Delta\colon \mathcal{P}(\mathcal{F}_{At}) \times \mathcal{P}(\mathcal{F}_{At}) \to \mathcal{P}(\mathcal{F}_{At})$$

by setting for every pair of sets $U, V \in \mathcal{P}(\mathcal{F}_{At})$

$$\Psi_\Delta(U, V) = W \cup \{cons(d)\colon d \in D, (U, V) \rhd d\}.$$

It is easy to check that the operator $\Psi_\Delta(\cdot, V)$ is monotone. Thus, it has a least fixpoint and we define

$$\Gamma_\Delta(V) = Cn(lfp(\Psi_\Delta(\cdot, V))).$$

The choice of the notation is not accidental. The operator Γ_Δ is indeed the operator Γ introduced in (Reiter 1980). We call a set $E \in \mathcal{P}(\mathcal{F}_{At})$ an *extension* of Δ if

$$E = \Gamma_\Delta(E).$$

Given extensions as basic semantic objects, we now define the concepts of strong and uniform equivalence of default theories (the notion of strong equivalence was introduced in (Turner 2001), in a slightly more general setting of nested default theories). We will use the following notation: for default theories $\Delta' = (D', W')$ and $\Delta'' = (D'', W'')$, we will write $\Delta' \cup \Delta''$ for the default theory $(D' \cup D'', W' \cup W'')$.

Definition 4 *Let Δ' and Δ'' be default theories.*

1. *Δ' and Δ'' are* strongly equivalent *if for every default theory Δ, the default theories $\Delta' \cup \Delta$ and $\Delta'' \cup \Delta$ have the same extensions*
2. *Δ' and Δ'' are* uniformly equivalent *if for every default theory $\Delta = (\emptyset, W)$, the default theories $\Delta' \cup \Delta$ and $\Delta'' \cup \Delta$ have the same extensions.*

We will now show that these two concepts fall into the general algebraic scheme discussed in the paper.

We observed earlier that for every $V \in \mathcal{P}(\mathcal{F}_{At})$, the operator $\Psi_\Delta(\cdot, V)$ is monotone. It is also easy to see that for every $U \in \mathcal{P}(\mathcal{F}_{At})$, the operator $\Psi_\Delta(U, \cdot)$ is antimonotone. It follows that Ψ_Δ is an approximating mapping for the operator G_Δ on $\mathcal{P}(\mathcal{F}_{At})$ such that for every $U \in \mathcal{P}(\mathcal{F}_{At})$,

$$G_\Delta(U) = \Psi_\Delta(U, U).$$

The following property of extensions is a direct consequence of the corresponding definitions.

Theorem 15 *Let $\Delta = (D, W)$ be a default theory. Then a set $E \in \mathcal{P}(\mathcal{F}_{At})$ is an extension of Δ if and only if there is $V \in \mathcal{P}(\mathcal{F}_{At})$ such that V is a Ψ_Δ-stable fixpoint of G_Δ (that is, $V = lfp(\Psi_\Delta(\cdot, V))$) and $E = Cn(V)$.*

Proof: $(\Rightarrow)$ Let E be an extension of Δ, that is,

$$E = \Gamma_\Delta(E) = Cn(lfp(\Psi_\Delta(\cdot, E))).$$

Let us define $V = \mathit{lfp}(\Psi_\Delta(\cdot, E))$. It follows that $E = Cn(V)$. Moreover, since $E = Cn(V)$, for every $U \in \mathcal{P}(\mathcal{F}_{At})$, we have $\Psi_\Delta(U, V) = \Psi_\Delta(U, E)$. Consequently, $V = \mathit{lfp}(\Psi_\Delta(\cdot, V))$.
($\Leftarrow$) If $E = Cn(V)$ then for every $U \in \mathcal{P}(\mathcal{F}_{At})$ we have $\Psi_\Delta(U, V) = \Psi_\Delta(U, E)$. Thus,

$$\begin{aligned} E &= Cn(V) = Cn(\mathit{lfp}(\Psi_\Delta(\cdot, V))) \\ &= Cn(\mathit{lfp}(\Psi_\Delta(\cdot, E))) = \Gamma_\Delta(E). \end{aligned}$$

Thus, E is an extension of Δ. □

Theorem 15 implies that extensions of a default theory Δ are precisely the closures under propositional consequence of Ψ_Δ-stable fixpoints of G_Δ. Consequently, we have the following result establishing a connection between strong (uniform) equivalence of default theories Δ' and Δ'', and strong (uniform) equivalence of operators $G_{\Delta'}$ and $G_{\Delta''}$.

Theorem 16 *Let Δ' and Δ'' be default theories. Then Δ' and Δ'' are strongly (respectively, uniformly) equivalent if and only if the operators $G_{\Delta'}$ and $G_{\Delta''}$ are strongly (respectively, uniformly) equivalent with respect to $(\Psi_{\Delta'}, \Psi_{\Delta''})$.*

Proof: We recall that the lattice of interest here is the lattice $\langle \mathcal{P}(\mathcal{F}_{At}), \subseteq \rangle$, and that the corresponding join operator is $\cup$. We also note that for every two default theories Δ' and Δ'', we have

$$G_{\Delta' \cup \Delta''} = G_{\Delta'} \cup G_{\Delta''},$$

and

$$\Psi_{\Delta' \cup \Delta''} = \Psi_{\Delta'} \cup \Psi_{\Delta''}.$$

We will now deal with the case of strong equivalence.
($\Leftarrow$) Let Δ be an arbitrary default theory. Since $G_{\Delta'}$ and $G_{\Delta''}$ are strongly equivalent with respect to $(\Psi_{\Delta'}, \Psi_{\Delta''})$, $\Psi_{\Delta'} \cup \Psi_\Delta$-stable fixpoints of $G_{\Delta'} \cup G_\Delta$ are the same as $\Psi_{\Delta''} \cup \Psi_\Delta$-stable fixpoints of $G_{\Delta''} \cup G_\Delta$. By our observations above, $\Psi_{\Delta' \cup \Delta}$-stable fixpoints of $G_{\Delta' \cup \Delta}$ are the same as $\Psi_{\Delta'' \cup \Delta}$-stable fixpoints of $G_{\Delta'' \cup \Delta}$. By Theorem 15, $\Delta' \cup \Delta$ and $\Delta'' \cup \Delta$ have the same extensions and so, Δ' and Δ'' are strongly equivalent.
($\Rightarrow$) Let S be an arbitrary simple operator on the lattice $\mathcal{P}(\mathcal{F}_{At})$ (with the inclusion as the ordering relation). Then, there are sets $X, Y \in \mathcal{P}(\mathcal{F}_{At})$ such that $X \subseteq Y$ and

$$S(Z) = \begin{cases} X & \text{if } Z \subseteq X \\ Y & \text{otherwise} \end{cases}$$

for every $Z \in \mathcal{P}(\mathcal{F}_{At})$. Let us define

$$D = \{\frac{\alpha :}{\beta} : \alpha \in Y, \beta \in \mathcal{F}_{At} \setminus X\}$$

and set $\Delta = (D, X)$. Clearly, $G_\Delta = S$

Since Δ' and Δ'' are strongly equivalent, $\Delta' \cup \Delta$ and $\Delta'' \cup \Delta$ have the same extensions. In the language of operators, it means that $St(G_{\Delta'} \cup G_\Delta, \Psi_{\Delta'} \cup \Psi_\Delta) = St(G_{\Delta''} \cup G_\Delta, \Psi_{\Delta''} \cup \Psi_\Delta)$. As all defaults of Δ are justification-free, $\Psi_\Delta(U, V) = \Psi_\Delta(U, U) = G_\Delta(U) = S(U) = C_S(U, V)$. It follows that $St(G_{\Delta'} \cup S, \Psi_{\Delta'} \cup C_S) = St(G_{\Delta''} \cup S, \Psi_{\Delta''} \cup C_S)$. By Theorem 5, $G_{\Delta'}$ and $G_{\Delta''}$ are strongly equivalent with respect to $(\Psi_{\Delta'}, \Psi_{\Delta''})$.

For the case of uniform equivalence the argument is similar but it requires an observation that constant operators on $\mathcal{P}(\mathcal{F}_{At})$ are precisely the operators of the form G_Δ, for some default theory $\Delta = (\emptyset, W)$. □

Theorem 16 allows us to apply the results of this paper to characterize the strong and uniform equivalence of default theories.

A pair (U, V), where $U, V \in \mathcal{P}(\mathcal{F}_{At})$ is a *default se-pair* (or, *dse-pair*) for a default theory $\Delta = (D, W)$ if

(SE-DL1) $W \subseteq U \subseteq V$

(SE-DL2) for every default $d \in D$, if $(V, V) \rhd d$ then $\mathit{cons}(d) \in V$

(SE-DL3) for every default $d \in D$, if $(U, V) \rhd d$, then $\mathit{cons}(d) \in U$.

One can check that (U, V) is a dse-pair for a default theory Δ if and only if (U, V) is an *se-pair* for the operator G_Δ with respect to Ψ_Δ. Thus, Corollary 4 implies the following result.

Theorem 17 *Default theories Δ' and Δ'' are strongly equivalent if and only if they have the same dse-pairs.*

This result in turn has a corollary, which allows one to restrict the class of se-pairs that one needs to inspect when testing strong equivalence.

Corollary 18 *Default theories Δ' and Δ'' are strongly equivalent if and only if they have the same dse-pairs (U, V), where $U, V \subseteq W' \cup W'' \cup \{\mathit{cons}(d) \colon d \in D' \cup D''\}$.*

Our general results also imply characterizations of the uniform equivalence of default theories. We say that a set $V \subseteq \mathcal{F}_{At}$ is *closed* under a set D of defaults if for every $d \in \Delta$ such that $(V, V) \rhd d$, $\mathit{cons}(d) \in V$.

Theorem 19 *Default theories Δ' and Δ'' are uniformly equivalent if and only if*

1. *for every $V \subseteq \mathcal{F}_{At}$, V is closed under D' if and only if V is closed under D'', where D' and D'' are the sets of defaults of Δ' and Δ'', respectively*
2. *for every dse-pair (U, V) for Δ', if $U \subsetneq V$ then there is U' such that $U \subseteq U' \subsetneq V$ and (U', V) is a dse-pair for Δ''*
3. *for every dse-pair (U, V) for Δ'', if $U \subsetneq V$ then there is U' such that $U \subseteq U' \subsetneq V$ and (U', V) is a dse-pair for Δ'.*

In the case of finite default theories, the characterization can be restated in terms of default ue-pairs. A default se-pair for a default theory Δ, say (U, V), is a *default ue-pair* (or, *due-pair*) for Δ if for every default se-pair (U', V) for Δ such that $U \subsetneq U'$, we have $U' = V$.

Theorem 20 *Let Δ' and Δ'' be finite default theories. Then Δ' and Δ'' are uniformly equivalent if and only if they have the same due-pairs.*

Discussion

We showed in the paper that our approach yields as corollaries results on strong and uniform equivalence of logic programs and default theories. In a similar way, we can characterize strong and uniform equivalence of logic programs with aggregates as studied in (Pelov. 2004; Pelov, Denecker, & Bruynooghe 2004), and of modal theories with the semantics of extensions (Denecker, Marek, & Truszczyński 2000), which yields a version of autoepistemic logic forming a precise modal match to the default logic. The reason is that in each case the semantics (stable models, extensions) is given in terms of an operator on a complete lattice and its approximating mapping.

Our approach, as presented here, it does not apply to nested logic programs and nested default theories. We conjecture that it can be extended to cover these formalisms by building on the algebraic approach to disjunctive logic programming proposed in (Pelov & Truszczyński 2004). This is a topic of our ongoing research.

A fundamental research question is whether there are other versions of equivalence of operators on complete lattices. (Pearce & Valverde 2004) argued that in the context of answer-set programming strong and uniform equivalence are the only two concepts of this type. Our results suggest that the two concepts are close to each other also in a more general algebraic setting we considered here. Namely, as long as we define equivalence in terms of extending operators defined non-trivially on the *entire* lattice L, they essentially exhaust all possibilities. Considering constant operators (with their canonical approximations) as extending operators characterizes uniform equivalence. Considering just a slightly larger class of simple operators (moreover, also with their canonical approximations only) already yields the notion of strong equivalence.

To get a new notion of equivalence, we would need a class of operators containing constant operators but not simple ones. One candidate is the class of antimonotone operators. This class, however, does not seem to correspond to any situations of practical relevance. Another possibility is to consider constant operators only, as in uniform equivalence, but allow arbitrary approximating mappings. We note however, that in the context of logic programming (and most likely also other nonmonotonic logics) this is not a promising direction. The reason is that if a program P is a set of facts, no natural approximating mappings emerged for T_P other than the two-input operator Ψ_P.

On the other hand, an interesting and important extension of strong and uniform equivalence of programs can be obtained by restricting the class of extending programs to those built only of atoms from some fixed set $A \subseteq At$ (Eiter, Fink, & Woltran 2006). This approach results in strong and uniform equivalence of programs *relativized* with respect to A. We observe that the relativized equivalence can be considered in our algebraic setting. Let L be a complete lattice and let $y \in L$. An operator R on L is a *y-operator* if (1) for every $z \in L$, $R(z) \leq y$, and (2) for every $z_1, z_2 \in L$, $R(z_1 \wedge y) = R(z_2 \wedge y)$; that is, if R is determined by an operator on the complete lattice $\{x \in L\colon x \leq y\}$. By allowing only y-operators as extending operators, we obtain strong and uniform *y-equivalence*, which generalizes the corresponding notions from (Eiter, Fink, & Woltran 2006) proposed there for programs. We are presently studying algebraic properties of strong and uniform y-equivalence.

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.

Eiter, T.; Fink, M.; and Woltran, S. 2006. Semantical characterizations and complexity of equivalences in answer set programming. *ACM Transactions on Computational Logic*. To appear.

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.

Marek, W., and Truszczyński, M. 1993. *Nonmonotonic Logic; Context-Dependent Reasoning*. Berlin: Springer.

Pearce, D., and Valverde, A. 2004. Uniform equivalence for equilibrium logic and logic programs. In *Proceedings of the 7th International Conference on Logic Programming* *and Nonmonotonic Reasoning*, volume 2923 of *Lecture Notes in Artificial Intelligence*, 194–206. Springer.

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.*

Reiter, R. 1980. A logic for default reasoning. *Artificial Intelligence* 13(1-2):81–132.

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.