Hyperequivalence of programs and operators*
(Preliminary version)

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

Stefan Woltran
Institut für Informationssysteme 184/2
Technische Universität Wien
Favoritenstraße 9-11
A-1040 Vienna, Austria
woltran@dbai.tuwien.ac.at

Dedicated to Victor Marek
on his 65th birthday

Abstract

Recent research in nonmonotonic logic programming has focused on different notions of program equivalence relevant for program optimization and modular programming. So far, most results concern the stable-model semantics. However, other semantics for logic programs are also of interest, especially the semantics of supported models which, when properly generalized, is closely related to a prominent nonmonotonic formalism, the autoepistemic logic of Moore. In this paper, we consider a framework of equivalence notions for logic programs under the supported (minimal) model-semantics and provide characterizations for this framework in both model-theoretic and algebraic terms. We then use some of our results in order to obtain characterizations of the equivalence of theories in Moore's autoepistemic logic.

1 Introduction

The problem of the equivalence of logic programs with respect to the stable-model semantics has received substantial attention in the answer-set programming research community in the past several years (Lifschitz, Pearce, & Valverde 2001; Lin 2002; Turner 2003; Inoue & Sakama 2004; Eiter, Tompits, & Woltran 2005; Eiter, Fink, & Woltran 2007; Oikarinen & Janhunen 2006; Oetsch, Tompits, & Woltran 2007; Woltran 2007). In most general terms, the problem can be stated as follows. Given a class $\mathcal{C}$ of logic programs, we say that programs P and Q are *equivalent with respect to* $\mathcal{C}$ if for every program $R \in \mathcal{C}$, $P \cup R$ and $Q \cup R$ have the same *stable models*. We will sometimes refer to programs in $\mathcal{C}$ as *contexts*. Clearly, for every class $\mathcal{C}$, the equivalence with respect to $\mathcal{C}$ implies the standard nonmonotonic equivalence of programs, where two programs P and Q are *nonmonotonically* equivalent if they have the same stable models. Therefore, we will refer to these stronger versions of equivalence collectively as *hyperequivalence*.

Understanding hyperequivalence is fundamental for the development of modular answer-set programs and knowledge bases. The problem is non-trivial due to the nonmonotonic nature of the stable-model semantics. If S is a module within a larger program T, replacing S with S' results in the program $T' = (T \setminus S) \cup S'$, which must have the same meaning (the same stable models) as T. The nonmonotonic equivalence of S and S' does not guarantee it. The equivalence with respect to the class of all programs does. However, the latter may be a too restrictive approach in certain application scenarios, in particular if properties of possible realizations for T are known in advance.

Several notions of hyperequivalence can be obtained by different restrictions to the context class $\mathcal{C}$. In particular, if $\mathcal{C}$ is unrestricted, that is, any program is a possible context, we obtain *strong* equivalence (Lifschitz, Pearce, & Valverde 2001). If $\mathcal{C}$ is the collection of all set of facts, one obtains *uniform* equivalence (Eiter, Fink, & Woltran 2007). Another direction is to restrict the alphabet over which contexts are given. Such notions are called *relativized* (with respect to such an alphabet) and can be combined with strong or uniform equivalence (Eiter, Fink, & Woltran 2007). In an even more general setting, we can specify different alphabets for bodies and heads of rules in contexts. This gives rise to a common view on strong and uniform equivalence as discussed in (Woltran 2007). A different research direction recently applied to equivalence notions is to compare only some dedicated projected output atoms (see (Eiter, Tompits, & Woltran 2005; Oikarinen & Janhunen 2006; Oetsch, Tompits, & Woltran 2007), rather than entire stable models.

All of these results concern the case when the semantics of programs is given by stable models.[1] In this paper, we address the problem of the hyperequivalence of programs with respect to the other major semantics, that of supported models (Clark 1978; Apt, Blair, & Walker 1988). We define several concepts of hyperequivalence, depending on the class of programs allowed as contexts. For some of these concepts, we obtain characterizations in terms of semantic objects that can be attributed to a program, similar to se-models (Turner 2003) or ue-models (Eiter, Fink, & Woltran 2007). We extend our characterizations to an abstract algebraic setting of operators on lattices, following the approach proposed in

*This work was partially supported by the NSF grant IIS-0325063, the KSEF grant KSEF-1036-RDE-008, and by the Austrian Science Fund (FWF) under grant P18019.

[1] In fact, there is only little work on other semantics, e.g. (Cabalar *et al.* 2006).

(Truszczyński 2006) as the generalization of hyperequivalence based on the stable-model semantics. We use these results to derive characterizations of hyperequivalence for theories in the autoepistemic logic of Moore (Moore 1984; 1985).

While in many respects the problem of hyperequivalence for the semantics of supported models is simpler than its counterpart for the stable-model semantics, the situation changes if we impose additional requirements on supported models. In the paper, we study one such restriction: the semantics of supported *minimal* models (Zhang & Marek 1989). We obtain some characterizations of the resulting concept of hyperequivalence, but the picture here is far from complete.

2 The case of logic programs

We fix a countable infinite set At of atoms. All programs we consider here are built of atoms in At. Let $A, B \subseteq At$. By $\mathcal{LP}(A, B)$ we denote the class of all disjunctive logic programs P such that $hd(P) \subseteq A$ and $bd(P) \subseteq B$, that is, all atoms in the heads of rules in P come from A, and all atoms in the bodies of rules in P come from B.

If r is a logic program rule

$$a_1 | \ldots | a_k \leftarrow bd,$$

where $k \geq 1$, a *shift* of r is a normal logic program rule of the form

$$a_i \leftarrow bd, not\, a_1, \ldots, not\, a_{i-1}, not\, a_{i+1}, \ldots, not\, a_k,$$

where $i = 1, \ldots, k$. If r is a constraint, the only *shift* of r is r itself. A program consisting of all shifts of rules in a program P is the *shift* of P. We denote it by $sh(P)$. It is evident that a set Y of atoms is a model of P if and only if Y is a model of $sh(P)$. It is also known that Y is a supported model of P if and only if it is a supported model of $sh(P)$.

Two disjunctive logic programs (with constraints) P and Q are *supp-equivalent* relative to $\mathcal{LP}(A, B)$ if for every disjunctive logic program $R \in \mathcal{LP}(A, B)$, $P \cup R$ and $Q \cup R$ have the same supported models.

Let P be a normal logic program (with constraints). By T_P we denote the one-step provability operator for P (van Emden & Kowalski 1976). We note that due to the presence of constraints, T_P may be partial (undefined for some interpretations). It is well known that Y is a model of P if and only if $T_P(Y) \subseteq Y$ (by that we mean that T_P is defined for Y and satisfies $T_P(Y) \subseteq Y$). Similarly, Y is a supported model of P if $T_P(Y) = Y$ (by that we mean that T_P is defined for Y and satisfies $T_P(Y) = Y$). In the paper, whenever we write $T_P(Y)$, it is implied by the context that $T_P(Y)$ is defined (for instance, $T_P(Y)$ is defined if Y is a model of P).

Given a normal logic program with constraints, say P, and a set $A \subseteq At$ of atoms, we define

$$Mod_A(P) = \{Y \subseteq At \mid Y \models P \text{ and } Y \setminus T_P(Y) \subseteq A\}.$$

We have the following characterization of the supp-equivalence relative to $\mathcal{LP}(A, B)$.

Theorem 2.1 *Let P and Q be normal logic programs (with constraints) and $A, B \subseteq At$. The following conditions are equivalent*

1. *P and Q are supp-equivalent relative to $\mathcal{LP}(A, At)$*
2. *P and Q are supp-equivalent relative to $\mathcal{LP}(A, B)$*
3. *P and Q are supp-equivalent relative to $\mathcal{LP}(A, \emptyset)$*
4. *$Mod_A(P) = Mod_A(Q)$ and for every $Y \in Mod_A(P)$, $T_P(Y) = T_Q(Y)$.*

Proof: Clearly, (1) implies (2), and (2) implies (3). We will now show that (3) implies (4). Let $Y \in Mod_A(P)$. It follows that $Y \models P$ and $Y \setminus T_P(Y) \subseteq A$. Let us consider $P \cup (Y \setminus T_P(Y))$. Then

$$T_{P \cup (Y \setminus T_P(Y))}(Y) = T_P(Y) \cup (Y \setminus T_P(Y)).$$

Since $Y \models P$, $T_P(Y) \subseteq Y$. Thus, $T_{P \cup (Y \setminus T_P(Y))}(Y) = Y$. It follows that Y is a supported model of $P \cup (Y \setminus T_P(Y))$. Since $Y \setminus T_P(Y) \subseteq A$, Y is a supported model of $Q \cup (Y \setminus T_P(Y))$ and, consequently,

$$Y = T_{Q \cup (Y \setminus T_P(Y))}(Y) = T_Q(Y) \cup (Y \setminus T_P(Y)).$$

It follows that $T_Q(Y) \subseteq Y$ and $T_P(Y) \subseteq T_Q(Y)$. Thus, $Y \setminus T_Q(Y) \subseteq Y \setminus T_P(Y) \subseteq A$ and so, $Y \in Mod_A(Q)$. The converse inclusion follows by the symmetry argument.

Next, let $Y \in Mod_A(P)$ (and so, $Y \in Mod_A(Q)$, too). We have seen that $T_P(Y) \subseteq T_Q(Y)$. By the symmetry, $T_Q(Y) \subseteq T_P(Y)$. Thus, $T_P(Y) = T_Q(Y)$.

To show that (4) implies (1), we reason as follows. Let R be a logic program from $\mathcal{LP}(A, At)$. Let $Y \in Supp(P \cup R)$. Then, $Y \in Supp(sh(P \cup R)) = Supp(P \cup sh(R))$ (we recall that P is normal). It follows that $Y = T_P(Y) \cup T_{sh(R)}(Y)$. Thus, $T_P(Y) \subseteq Y$ (that is, $Y \models P$) and $Y \setminus T_P(Y) \subseteq A$ (because $hd(R) \subseteq A$). Thus, $Y \in Mod_A(P)$ and, by the assumption, $T_Q(Y) = T_P(Y)$. Since $Y = T_P(Y) \cup T_{sh(R)}(Y)$, $Y = T_Q(Y) \cup T_{sh(R)}(Y)$. That is, $Y \in Supp(Q \cup sh(R))$ and, since $sh(Q) = Q$, $Y \in Supp(sh(Q \cup R)) = Supp(Q \cup R)$. □

We point out that the characterization of supp-equivalence relative to $\mathcal{LP}(A, B)$ does not refer to the second alphabet B. In other words, different equivalence notions coincide as long as the context allows for the same atoms used in the heads. Therefore, the same characterization applies to strong and uniform equivalence.

We proceed by extending our result to the case of disjunctive programs.

Theorem 2.2 *Let P and Q be disjunctive logic programs (with constraints) and $A, B \subseteq At$. Then, P and Q are supp-equivalent relative to $\mathcal{LP}(A, B)$ if and only if $sh(P)$ and $sh(Q)$ are supp-equivalent relative to $\mathcal{LP}(A, B)$.*

Proof: (1) Let us assume P and Q are supp-equivalent relative to $\mathcal{LP}(A, B)$. Let $R \subseteq A$. Then

$$\begin{aligned} Supp(sh(P) \cup R) &= Supp(sh(P \cup R)) \\ &= Supp(P \cup R) = Supp(Q \cup R) \\ &= Supp(sh(Q \cup R)) = Supp(sh(Q) \cup R). \end{aligned}$$

It follows that $sh(P)$ and $sh(Q)$ are supp-equivalent relative to $\mathcal{LP}(A, \emptyset)$. By Theorem 2.1, $sh(P)$ and $sh(Q)$ are supp-equivalent relative to $\mathcal{LP}(A, B)$.

Conversely, let us consider $R \in \mathcal{LP}(A, B)$. Then

$$\begin{aligned} Supp(P \cup R) &= Supp(sh(P \cup R)) \\ &= Supp(sh(P) \cup sh(R)) = Supp(sh(Q) \cup sh(R)) \\ &= Supp(sh(Q \cup R)) = Supp(Q \cup R). \end{aligned}$$

Thus, the assertion follows. □

Corollary 2.3 *Let P and Q be disjunctive logic programs (with constraints) and let $A, B \subseteq At$. The following conditions are equivalent*

1. *P and Q are supp equivalent relative to $\mathcal{LP}(A, At)$*
2. *P and Q are supp equivalent relative to $\mathcal{LP}(A, B)$*
3. *P and Q are supp equivalent relative to $\mathcal{LP}(A, \emptyset)$*
4. *$Mod_A(sh(P)) = Mod_A(sh(Q))$ and for every $Y \in Mod_A(sh(P))$, $T_{sh(P)}(Y) = T_{sh(Q)}(Y)$.*

3 Suppmin-equivalence

Next, we move on to the semantics of supported minimal models. A set M of atoms is a *supported minimal model* (*suppmin* model, for short) of a logic program P if it is a supported model of P and a minimal model of P.

As before, let $A, B \subseteq At$. Two logic programs P and Q are *suppmin-equivalent* relative to $\mathcal{LP}(A, B)$ if for every program $R \in \mathcal{LP}(A, B)$, $P \cup R$ and $Q \cup R$ have the same suppmin models. A refinement of the method used in the previous section provides a characterization of suppmin-equivalence in the case when $A \subseteq B$. Compared to supp-equivalence the second alphabet, B, has to be taken into consideration now.

Let us define $Mod_A^B(P)$ to be the set of all pairs (X, Y) such that

1. $Y \in Mod_A(P)$
2. $X \subseteq Y|_B$
3. for every $Z \subset Y$, if $Z|_B = X$ or $Z|_B = Y|_B$, then $Z \not\models P$.

In this definition and in what follows, expressions such as $Z|_B$ stand for $Z \cap B$. We have the following result.

Theorem 3.1 *Let $A, B \subseteq At$ satisfy $A \subseteq B$, and let P, Q be normal programs (with constraints). The following conditions are equivalent*

1. *P and Q are suppmin-equivalent relative to $\mathcal{LP}(A, B)$*
2. *$Mod_A^B(P) = Mod_A^B(Q)$ and for every $(X, Y) \in Mod_A^B(P)$, $T_P(Y) = T_Q(Y)$.*

[(1) implies (2)]: Let us assume that $(X, Y) \in Mod_A^B(P)$. If $X = Y|_B$, we set $R' = Y \setminus T_P(Y)$. Otherwise, we fix an element, say $t \in Y|_B \setminus X$, and define

$$R' = \{y \leftarrow t \mid y \in Y \setminus T_P(Y)\}.$$

Finally, we set

$$\begin{aligned} R = R' \cup \{ &\leftarrow not\, x \mid x \in X\} \\ &\cup \{\leftarrow u, not\, z \mid u, z \in Y|_B \setminus X\}. \end{aligned}$$

We note that $R \in \mathcal{LP}(A, B)$. Indeed, since $Y \in Mod_A(P)$, we have $Y \setminus T_P(Y) \subseteq A$. Moreover, $Y|_B \subseteq B$ (trivially) and $X \subseteq Y|_B \subseteq B$ (by the definition of $Mod_A^B(P)$). Finally, t, which forms the bodies of rules in R' if $X \subset Y|_B$, is also in B. Thus, every atom in the body of a rule in R is in B. We also note that $Y \models R$ and $X \models R$.

By a similar argument as that in the proof of Theorem 2.1, Y is a supported model of $P \cup R$. We will show that Y is a minimal model of $P \cup R$. To this end, let us consider $Z \subseteq Y$ such that $Z \models P \cup R$. Since $Z \models R$, $X \subseteq Z$. We recall that $X \subseteq B$. Thus, $X \subseteq Z|_B \subseteq Y|_B$. If there are u, z such that $u \in Z|_B \setminus X$ and $z \in Y|_B \setminus Z|_B$, then the rule $\leftarrow u, not\, z$ belongs to R, which contradicts the assumption that $Z \models R$. Thus, either $X = Z|_B$ or $Z|_B = Y|_B$. Since $(X, Y) \in Mod_A^B(P)$, and $Z \models P$, we obtain $Z = Y$. It follows that Y is a minimal model of $P \cup R$ and so, a supported minimal model of $P \cup R$.

We have that P and Q are suppmin-equivalent relative to $\mathcal{LP}(A, B)$. Hence, Y is a supported minimal model of $Q \cup R$. As in the proof of Theorem 2.1, we can now show that $T_P(Y) \subseteq T_Q(Y)$, and $Y \in Mod_A(Q)$. Let $Z \subset Y$ be such that $X = Z|_B$ or $Z|_B = Y|_B$. In the first case, since $X \models R$, $Z|_B \models R$ and, as R contains only atoms from B, $Z \models R$. In the second case, since $Y \models R$, $Y|_B \models R$ (again, since R contains only atoms from B). Thus, $Z|_B \models R$. Consequently, $Z \models R$ in that case, too. Since Y is a minimal model of $Q \cup R$, it follows that $Z \not\models Q$. Since $Y \in Mod_A(Q)$ and $X \subseteq Y|_B$, we obtain that $(X, Y) \in Mod_A^B(Q)$. Hence, $Mod_A^B(P) \subseteq Mod_A^B(Q)$ and, to recall a property proved earlier, for every $(X, Y) \in Mod_A^B(P)$, $T_P(Y) \subseteq T_Q(Y)$.

By the symmetry argument, $Mod_A^B(Q) \subseteq Mod_A^B(P)$ and for every $(X, Y) \in Mod_A^B(Q)$, $T_Q(Y) \subseteq T_P(Y)$. It follows that $Mod_A^B(Q) = Mod_A^B(P)$ and for every $(X, Y) \in Mod_A^B(P)$, $T_P(Y) = T_Q(Y)$.

[(2) implies (1)]: Let R be a logic program from $\mathcal{LP}(A, B)$, and let Y be a supported minimal model of $P \cup R$. It follows that $Y \models P$ and $Y \models R$. It also follows that Y is a supported model of $P \cup sh(R)$, that is, $Y = T_P(Y) \cup T_{sh(R)}(Y)$. This latter identity shows (in a similar way as in the proof of Theorem 2.1) that $Y \setminus T_P(Y) \subseteq A$. Since $Y \models P$, $Y \in Mod_A(P)$.

Let us consider $Z \subset Y$ such that $Z|_B = Y|_B$. Since $Y \models R$, $Z \models R$. Since Y is a minimal model of $P \cup R$, $Z \not\models P$. Thus, $(Y|_B, Y) \in Mod_A^B(P)$. By the assumption, $(Y|_B, Y) \in Mod_A^B(Q)$ and $T_P(Y) = T_Q(Y)$. Since $Y = T_P(Y) \cup T_{sh(R)}(Y)$, it follows that $Y = T_Q(Y) \cup T_{sh(R)}(Y)$, that is, Y is a supported model of $Q \cup R$.

Let $Y' \subset Y$ be a model of $Q \cup R$. Since $Y' \models Q$, $(Y'|_B, Y) \notin Mod_A^B(Q)$. By the assumption, $(Y'|_B, Y) \notin Mod_A^B(P)$. Since $Y'|_B \subseteq Y|_B$ and $Y \in Mod_A(P)$, there is $U \subset Y$ such that $Y'|_B = U|_B$ or $U|_B = Y|_B$, and $U \models P$. Since both $Y' \models R$ and $Y \models R$, it follows that $U \models R$. Thus, $U \models P \cup R$ and Y is not a minimal model of $P \cup R$, a contradiction. It follows that there is no $Y' \subset Y$ such that $Y' \models Q \cup R$. That is, Y is a supported minimal model of

$Q \cup R$. □

The proof of Theorem 3.1 implies the following corollary for disjunctive programs.

Corollary 3.2 *Let $A, B \subseteq At$ satisfy $A \subseteq B$, and let P, Q be disjunctive programs (with constraints). Then, P and Q are suppmin-equivalent relative to $\mathcal{LP}(A, B)$ if and only if $sh(P)$ and $sh(Q)$ are suppmin-equivalent relative to $\mathcal{LP}(A, B)$.*

Proof: The argument is similar to that we used to prove Theorem 2.2. It is based on two facts: the programs used to prove that (1) implies (2) in Theorem 3.1 are non-disjunctive (and so, invariant under shift), and the shift operation does not change supported nor minimal models of a disjunctive program. □

Theorem 3.1 implies a more direct characterization.

Corollary 3.3 *Let $A, B \subseteq At$ satisfy $A \subseteq B$, and let P, Q be disjunctive programs (with constraints). The following conditions are equivalent*

1. *P and Q are suppmin-equivalent relative to $\mathcal{LP}(A, B)$*
2. *$Mod_A^B(sh(P)) = Mod_A^B(sh(Q))$ and for every $(X, Y) \in Mod_A^B(sh(P))$, $T_{sh(P)}(Y) = T_{sh(Q)}(Y)$.*

Finally, we note that in the case when $A = B = At$, the concepts of suppmin-equivalence and supp-equivalence coincide.

Corollary 3.4 *Let P and Q be disjunctive logic programs (with constraints). The following conditions are equivalent*

1. *P and Q are suppmin-equivalent relative to $\mathcal{LP}(At, At)$*
2. *P and Q are supp-equivalent relative to $\mathcal{LP}(At, At)$.*

Proof: Since the operation of shift does not change models, minimal models and supported models, it is enough to prove the assertion under the assumption that both P and Q are normal.

[(1) implies (2)]: Let Y be a model of P. Then, Y is a supported minimal model of $P \cup Y$. By the assumption, Y is a supported minimal model of $Q \cup Y$. In particular, Y is a model of Q. By the symmetry argument, P and Q have the same models.

Let Y be a model of P (and so of Q). Clearly Y is a supported model of $P \cup (Y \setminus T_P(Y)) \cup \{ \leftarrow not\, y \mid y \notin Y\}$. Moreover, it is the only model and so, a minimal model of that program. Thus, Y is a supported minimal model of $Q \cup (Y \setminus T_P(Y)) \cup \{ \leftarrow not\, y \mid y \notin Y\}$. Since $T_Q(Y) \subseteq Y$, It follows that $T_P(Y) \subseteq T_Q(Y)$. By the symmetry argument, $T_Q(Y) \subseteq T_P(Y)$. Thus, $T_P(Y) = T_Q(Y)$. By Theorem 2.1, P and Q are supp-equivalent relative to $\mathcal{LP}(At, At)$.

[(2) implies (1)]: Let us consider a logic program R. Since P and Q are supp-equivalent, they have the same models. Thus, $P \cup R$ and $Q \cup R$ have the same models and, consequently, the same minimal models. By the assumption, $P \cup R$ and $Q \cup R$ have the same supported models. Thus, $P \cup R$ and $Q \cup R$ have the same supported minimal models. □

We observe that the results of this section do not address uniform suppmin-equivalence as they do not cover the case of $B = \emptyset$ (except for the trivial subcase $A = B = \emptyset$).

4 Algebraic setting

The approach presented in the previous section extends to an abstract algebraic setting. The role of programs is played by non-deterministic operators on a boolean algebra, and models and supported models of programs are represented by prefixpoints and fixpoints of the operators, respectively. We start by introducing formally the concepts we need.

First, we fix a boolean algebra and denote it by L. We write $\bot$ and $\top$ for the least and the greatest elements of L, and $\vee$, $\wedge$ and $-$ for the binary operations of the algebra. To simplify notation, we often write y_b for $y \wedge b$. Finally, we denote by $\leq$ the partial order relation of the algebra.

We call mappings from L to $\mathcal{P}(L)$ *non-deterministic operators* on L. We denote the set of all non-deterministic operators on L by $\mathcal{NO}_L$. We note that standard deterministic (partial) operators on L can be viewed as special non-deterministic operators on L, namely those that assign to each element of L a set with at most one element. We call this class of such operators in $\mathcal{NO}_L$ *deterministic* and denote it by $\mathcal{DO}_L$. Finally, we denote by $\mathcal{CO}_L$ the class of those constant operators in $\mathcal{DO}_L$ that do not use $\emptyset$ as their common value.

For $D, E \in \mathcal{NO}_L$, we define $D \vee E$ to be an operator in $\mathcal{NO}_L$ such that for every $y \in L$,

$$(D \vee E)(y) = \{u \vee v \mid u \in D(y), v \in E(y)\}.$$

We note that if y is a prefixpoint of $D \vee E$, then y is a prefixpoint of D and of E.

An element $y \in L$ is a *prefixpoint* of D if there is $z \in D(y)$ such that $z \leq y$. An element $y \in L$ is a *fixpoint* of D if $y \in D(y)$. If D is deterministic, these definitions coincide with the standard definitions of prefixpoints and fixpoints of deterministic operators. Finally, a fixpoint of D is a *super-fixpoint* of D if it is a minimal prefixpoint of D.

Let $\mathcal{O} \subseteq \mathcal{NO}_L$. Two operators $D, E \in \mathcal{NO}_L$ are *fp-equivalent relative to* $\mathcal{O}$ if for every operator $F \in \mathcal{O}$, the operators $D \vee F$ and $E \vee F$ have the same fixpoints. They are *sfp-equivalent relative to* $\mathcal{O}$ if for every operator $F \in \mathcal{O}$, the operators $D \vee F$ and $E \vee F$ have the same super-fixpoints.

We will now investigate fp- and sfp-equivalence relative to some special classes of operators on L. Let $a \in L$. We define $\mathcal{NO}_L(a)$ to be the class of all operators $D \in \mathcal{NO}_L$ such that for every $y \in L$ and every $z \in D(y)$, $z \leq a$. We define $\mathcal{DO}_L(a) = \mathcal{DO}_L \cap \mathcal{NO}_L(a)$, and $\mathcal{CO}_L(a) = \mathcal{CO}_L \cap \mathcal{NO}_L(a)$. Finally, for $a, b \in L$, we define $\mathcal{NO}_L(a, b)$ to be the class of all operators $D \in \mathcal{NO}_L(a)$ such that for every $x \in L$, $D(x) = D(x_b)$.

We will now characterize operators that are fp-equivalent relative to $\mathcal{NO}_L(a)$. This result generalizes Theorem 2.1 (with one-step provability operators representing programs, and fixpoints of the one-step provability operators representing supported models of programs).

To this end, we introduce still more notation. We say that $A \leq_{a,y} B$ if for every $z \in A$ such that $z \leq y$ and $y - z \leq a$, there is $z' \in B$ such that $z \leq z'$ and $z' \leq y$. We define $A =_{a,y} B$ if $A \leq_{a,y} B$ and $B \leq_{a,y} A$. Further, for every $D \in \mathcal{NO}_L$, we define $PF_a(D)$ to consist of all elements $y \in L$, for which there is $z \in D(y)$ such that $z \leq y$ and $y - z \leq a$.

Theorem 4.1 *Let $D, E \in \mathcal{NO}_L$ be operators on L, $a \in L$, and let $\mathcal{O}$ be a class of operators such that $\mathcal{CO}_L(a) \subseteq \mathcal{O} \subseteq \mathcal{NO}_L(a)$. Then, the following conditions are equivalent*

1. *D and E are fp-equivalent relative to $\mathcal{NO}_L(a)$*
2. *D and E are fp-equivalent relative to $\mathcal{O}$*
3. *D and E are fp-equivalent relative to $\mathcal{CO}_L(a)$*
4. *$PF_a(D) = PF_a(E)$, and for every $y \in PF_a(D)$, $D(y) =_{a,y} E(y)$.*

Proof: Clearly, (1) implies (2), and (2) implies (3). We will now show that (3) implies (4).

Let $y \in PF_a(D)$. It follows that there is $z \in L$ such that $z \in D(y)$, $z \leq y$ and $y - z \leq a$. Let us consider the operator C such that for every $u \in L$, $C(u) = y - z$. Since $y - z \leq a$, $C \in \mathcal{CO}_L(a)$.

We have $y = z \vee (y - z)$. Thus, $y \in (D \vee C)(y)$. By the assumption, $y \in (E \vee C)(y)$ and, consequently, there is $z' \in E(y)$ such that $y = z' \vee (y - z)$. It follows that $z' \leq y$ and $z \leq z'$. Thus, $y - z' \leq y - z \leq a$ and, consequently, $y \in PF_a(E)$. The converse inclusion follows by the symmetry argument.

Let $y \in PF_a(D)$, and let $z \in D(y)$ be such that $z \leq y$ and $y - z \leq a$. For such z, the argument above demonstrates the existence of $z' \in E(y)$ such that $z \leq z'$ and $z' \leq y$. Thus, $D(y) \leq_{a,y} E(y)$. The converse inequality follows by the symmetry argument. Thus, $D(y) =_{a,y} E(y)$.

To show that (4) implies (1), we reason as follows. Let $F \in \mathcal{NO}_L(a)$. Let us assume that y is a fixpoint of $(D \vee F)$. Then, there are $z \in D(y)$ and $z' \in F(y)$ such that $y = z \vee z'$. It follows that $z, z' \leq y$ and $y - z \leq z'$. Since $z' \leq a$ (we recall that $F \in \mathcal{NO}_L(a)$), $y - z \leq a$. Thus, $y \in PF_a(D)$ and so, $D(y) =_{a,y} E(y)$. Since $z \leq y$ and $y - z \leq a$, there is $z'' \in E(y)$ such that $z \leq z''$ and $z'' \leq y$. It follows that $y = z'' \vee z'$ and, consequently, $y \in (E \vee F)(y)$. Thus, every fixpoint of $D \vee F$ is a fixpoint of $E \vee F$. By the symmetry argument, $D \vee F$ and $E \vee F$ have the same fixpoints. Consequently, D and F are fp-equivalent relative to $\mathcal{NO}_L(a)$. □

Assuming $a = \top$, we obtain the following corollary concerning the fp-equivalence relative to the class of all operators and some of its subclasses. In the statement, we use the notation $=_y$ as an abbreviation for $=_{\top,y}$. It is easy to see that $A =_y B$ if for every $z \in A$ such that $z \leq y$, there is $z' \in B$ such that $z \leq z'$ and $z' \leq y$, and for every $z \in B$ such that $z \leq y$, there is $z' \in A$ such that $z \leq z'$ and $z' \leq y$.

Corollary 4.2 *Let $D, E \in \mathcal{NO}_L$ be operators on L, and $\mathcal{O}$ a class of operators such that $\mathcal{CO}_L \subseteq \mathcal{O} \subseteq \mathcal{NO}_L$. Then, the following conditions are equivalent*

1. *D and E are fp-equivalent relative to $\mathcal{NO}_L$*
2. *D and E are fp-equivalent relative to $\mathcal{O}$*
3. *D and E are fp-equivalent relative to $\mathcal{CO}_L$*
4. *D and E have the same prefixpoints, and for every prefixpoint of D, $D(y) =_y E(y)$.*

Next, we will develop an algebraic counterpart of our characterization of suppmin-equivalence. To this end, for $D \in \mathcal{NO}_L$, and for $a, b \in L$, with $a \leq b$, we define $PF^b_a(D)$ to be the set of all pairs (x, y), $x, y \in L$, such that (we recall that u_v stands for $u \wedge v$):

1. $y \in PF_a(D)$
2. $x \leq y_b$
3. for every $z < y$, if $z_b = x$ or $z_b = y_b$, then z is not a prefixpoint of D.

Theorem 4.3 *Let $a, b \in L$ be such that $a \leq b$, and let $D, E \in \mathcal{NO}_L$. The following conditions are equivalent*

1. *D and E are sfp-equivalent relative to $\mathcal{NO}_L(a, b)$*
2. *$PF^b_a(D) = PF^b_a(E)$, and for every $(x, y) \in PF^b_a(D)$, $D(y) =_{a,y} E(y)$.*

[(1) implies (2)]: Let us assume that $(x, y) \in PF^b_a(D)$. In particular, it follows that $y \in PF_a(D)$, that is, there is $u \in D(y)$ such that $u \leq y$ and $y - u \leq a$.

In the proof, we will use three auxiliary operators, A_1, A_2 and A_3. They are defined as follows. If $y_b - x \leq z$, we define $A_1(z) = \{y - u\}$; otherwise, we set $A_1(z) = \{\bot\}$. For A_2, we define $A_2(z) = \{\bot\}$, if $x \leq z$, and $A_2(z) = \emptyset$, otherwise. Finally, for A_3, we set $A_3(z) = \{\bot\}$, if $(y_b - x) \wedge z = \bot$ or $(y_b - x) - z = \bot$. Otherwise, we set $A_3(z) = \emptyset$.

We now define

$$F = A_1 \vee A_2 \vee A_3.$$

We note that $F \in \mathcal{NO}_L(a, b)$. Indeed, we observe that for every $z \in L$, $F(z)$ equals $y - u$, $\bot$ or $\emptyset$. Since $y - u \leq a$, for every $v \in F(z)$, $v \leq a$. Thus, $F \in \mathcal{NO}_L(a)$. Next, if $F(z) = \emptyset$, then $A_2(z) = \emptyset$ or $A_3(z) = \emptyset$. In the first case, $x \not\leq z$ and so, $x \not\leq z_b$. Thus, $A_2(z_b) = \emptyset$. In the second case, $(y_b - x) \wedge z \neq \bot$ and $(y_b - x) - z \neq \bot$. Since $y_b - x \leq b$, it follows that $(y_b - x) \wedge z_b \neq \bot$ and $(y_b - x) - z_b \neq \bot$. Thus, $A_3(z_b) = \emptyset$. In each case, $F(z_b) = \emptyset = F(z)$.

If $F(z) \neq \emptyset$, then $x \leq z$, and $(y_b - x) \wedge z = \bot$ or $(y_b - x) - z = \bot$. Since $x \leq y_b \leq b$, $x \leq z_b$. Moreover, since $y_b - x \leq b$, $(y_b - x) \wedge z_b = \bot$ or $(y_b - x) - z_b = \bot$. Thus, $F(z_b) \neq \emptyset$. It follows that $F(z) = A_1(z)$ and $F(z_b) = A_1(z_b)$. Since $y_b - x \leq b$, $y_b - x \leq z$ if and only $y_b - x \leq z_b$. Thus, $A_1(z) = A_1(z_b)$ and, consequently, $F(z) = F(z_b)$.

By a similar argument as that in the proof of Theorem 4.1, y is a fixpoint of $D \cup F$. Let us consider a prefixpoint z of $D \vee F$ such that $z \leq y$. In particular, it follows that $F(z) \neq \emptyset$. Since $F \in \mathcal{NO}_L(a, b)$, $F(z) = F(z_b)$. Thus, $A_2(z_b) = A_3(z_b) = \bot$. Consequently, $x \leq z_b$, and $(y_b - x) \wedge z_b = \bot$ or $(y_b - x) - z_b = \bot$. Since $x \leq z_b \leq y_b$, the first condition implies $z_b = x$, and the other one, $y_b = z_b$. We recall that $(x, y) \in PF^B_A(D)$. If $z < y$, then z is not a prefixpoint of D, a contradiction. Thus, $z = y$. It follows that y is a minimal prefixpoint of $D \vee F$ and so, a super-fixpoint of $D \vee F$.

From the assumption, it follows that y is super-fixpoint of $E \vee F$. In particular, we have that $y \in (E \vee F)(y)$. Since $y_b - x \leq y_b \leq y$, it follows that $F(y) = A_1(y) = \{y - u\}$. Thus, $y = v \vee (y - u)$, for some $v \in E(y)$. It follows that $v \leq y$ and $y - v \leq y - u \leq a$. Thus, $y \in PF_a(E)$. Moreover, u is an arbitrary element from $D(y)$ such that $u \leq y$ and $y - u \leq a$. Since $u \leq v$ (as $y - v \leq y - u$ and $u, v \leq y$), it follows that $D(y) \leq_{a,y} E(y)$.

Let $z < y$ be such that $x = z_b$ or $z_b = y_b$. In the first case, it is easy to see that $x \leq z$ and $(y_b - x) \wedge z = \bot$. Thus, $F(z) = A_1(z)$. If $y_b - x \not\leq z$, then $F(z) = \{\bot\}$ and so, z is a prefixpoint of F. So, let us assume that $y_b - x \leq z$. Since $x = z_b$, it follows that $y_b = x$ and $F(z) = \{y - u\}$. We recall that $y - u \leq a \leq b$. Thus, $y - u \leq y_b = x = z_b \leq z$. Hence, z is a prefixpoint of F in this case, too.

Let us now assume the second possibility, $z_b = y_b$. It is easy to check that $A_2(y_b) = \{\bot\}$ and $A_3(y_b) = \{\bot\}$. Since $x \leq y_b$, $F(z_b) = F(y_b) = A_1(y_b) = \{y - u\}$. We recall that $y - u \leq a \leq b$. Thus, $y - u \leq y_b = z_b \leq z$. Since $F \in \mathcal{NO}_L(a, b)$, $F(z) = F(z_b)$. It follows that z s a prefixpoint of F, also in this case.

Since y is a minimal prefixpoint of $E \cup F$ and $z < y$, it follows that z is not a prefixpoint of E. Since $y \in PF_a(E)$ and $x \leq y_b$, $(x, y) \in PF^b_a(E)$. Hence, $PF^b_a(D) \subseteq PF^b_a(E)$ and, to recall a property proved earlier, $D(y) \leq_{a,y} E(y)$.

By the symmetry argument, $PF^b_a(E) \subseteq PF^b_a(D)$ and for every $(x, y) \in PF^b_a(E)$, $E(y) \leq_{a,y} D(y)$. It follows that $PF^b_a(E) = PF^b_a(D)$ and for every $(x, y) \in PF^b_a(D)$, $D(y) =_{a,y} E(y)$.

[(2) implies (1)]: Let $F \in \mathcal{NO}_L(a, b)$, and let y be a super-fixpoint of $D \vee F$. It follows that there are $w \in D(y)$ and $v \in F(y)$ such that $w \vee v = y$. Since $F \in \mathcal{NO}_L(a, b)$, $v \leq a$. Thus, $w \leq y$ and $y - w \leq a$. Consequently, $y \in PF_a(D)$.

Let us consider $z < y$ such that $z_b = y_b$. Since $F \in \mathcal{NO}_L(a, b)$, $F(y_b) = F(y)$. Thus, $v \in F(y_b) = F(z_b)$. Since $v \leq a \leq b$ and $v \leq y$, $v \leq y_b = z_b$. By the assumption that $F \in \mathcal{NO}_L(a, b)$, $v \in F(z)$. Since $v \leq z_b \leq z$, z is a prefixpoint of F. We recall that y is a minimal prefixpoint of $D \vee F$. It follows that z is not a prefixpoint of D. Thus, $(y_b, y) \in PF^b_a(D)$. By the assumption, $(y_b, y) \in PF^b_a(E)$ and $D(y) =_{a,y} E(y)$. Since $w \in D(y)$, $w \leq y$ and $y - w \leq a$, there is $w \in E(y)$ such that $w \leq w'$ and $w' \leq y$. It follows that $y = w' \vee v$ and so, $y \in E(y) \vee F(y)$. In other words, y is a fixpoint of $E \vee F$.

Let $y' < y$ be a prefixpoint of $E \vee F$. Since y' is a prefixpoint of E, $(y'_b, y) \notin PF^b_a(E)$. By the assumption, $(y'_b, y) \notin PF^b_a(D)$. Since $y'_b \leq y_b$ and $y \in PF_a(D)$, there is $u < y$ such that $y'_b = u_b$ or $u_b = y_b$, and u is a prefixpoint of D. Since $a \leq b$, $F \in \mathcal{NO}_L(a, b)$, and both y' and y are prefixpoints of F, it follows that y'_b and y_b are prefixpoints of F. Since $y'_b = u_b$ or $u_b = y_b$, u_b is a prefixpoint of F. Thus, there is $u' \in F(u_b)$ such that $u' \leq u_b$. By the assumption that $F \in \mathcal{NO}_L(a, b)$, $F(u) = F(u_b)$. Thus, $u' \in F(u)$ and $u' \leq u_b \leq u$. That is, u is a prefixpoint of F and so, of $D \vee F$, contrary to the minimality of y.

It follows that there is no $y' < y$ such that y' is a prefixpoint of $E \vee F$. That is, y is a super-fixpoint of $E \vee F$. □

As in the case of logic programs, we have the following corollary.

Corollary 4.4 *Let $D, E \in \mathcal{NO}_L$. Then D and E are sfp-equivalent relative to $\mathcal{NO}_L$ if and only if D and E are fp-equivalent relative to $\mathcal{NO}_L$.*

Specializing these results to the case when L is the boolean algebra of all subsets of At (interpretations) and D and E are one-step provability operators for normal logic programs P and Q (considered as non-deterministic operators), we obtain as corollaries Theorems 2.1 and 3.1. Indeed, models of a normal logic program with constraints, say P, are precisely prefixpoints of T_P, and supported models of P are precisely fixpoints of T_P. Moreover, the class of constant operators on that algebra with values in a set A is contained in the class $\mathcal{LP}(A, At)$. Similarly, operators A_1, A_2 and A_3 used in the proof of Theorem 4.3, in the case of the algebra of subsets of At, belong to the class $\mathcal{LP}(A, B)$.

5 Autoepistemic logic

Autoepistemic logic is one of the classical nonmonotonic systems. It is a modal nonmonotonic logic introduced in (Moore 1984; 1985). The main motivation was to develop a nonmonotonic logic of belief sets of agents with perfect introspection capabilities. The goal was to correct serious shortcomings of an earlier attempt at the development of a modal nonmonotonic system (McDermott & Doyle 1980), that was ultimately established to be flawed.

The semantics of autoepistemic theories is given by the concept of an expansion (Moore 1984). As with other nonmonotonic systems, the question of hyperequivalence of modal theories in autoepistemic logic arises naturally. In this section we will address this question, exploiting our algebraic characterization of hyperequivalence of operators, and a well-known fact that expansions of a modal theory are in one-to-one correspondence with fixpoints of an operator on a boolean algebra.

We start with some preliminaries. We will consider the language of a propositional modal logic generated by a set of atoms At (possibly infinite), standard boolean connectives (including 0-ary connectives $\top$ and $\bot$) and one modal operator K. We will denote this language by $\mathcal{L}^K_{At}$ and its modal-free part by $\mathcal{L}_{At}$. As At is fixed, typically, we will omit an explicit references to At.

We denote by Cn the operator of propositional consequence, which we apply to subsets of $\mathcal{L}_{At}$. By $\mathcal{B}$ we denote the well-known boolean algebra consisting of all theories contained in $\mathcal{L}$ that are closed under propositional consequence. The $Cn(\emptyset)$ (the set of all tautologies) and $\mathcal{L}$ play the roles of $\bot$ and $\top$ in this algebra. Given $U, V \in \mathcal{B}$, the join $\sqcup$ is defined by $U \sqcup V = Cn(U \cup V)$, and the meet $\sqcap$ is defined by $U \sqcap V = U \cap V$. Finally, the maximal theory $W \subseteq \mathcal{L}$ closed under consequence and such that $U \sqcap W = Cn(\emptyset)$, is the complement of U.

Let $F \in \mathcal{B}$ and let $\varphi \in \mathcal{L}^K_{At}$. We define the formula φ_F by induction as follows:

1. $\varphi_F = \varphi$, if $\varphi \in \mathcal{L}_{At}$
2. $(\neg\varphi)_F = \neg\varphi_F$, other boolean connectives are handled in a similar standard way
3. $(K\varphi)_F = \top$, if $\varphi \in F$; $(K\varphi)_F = \bot$, otherwise.

It is evident that $\varphi_F \in \mathcal{L}$.

Let $T \subseteq \mathcal{L}_{At}^K$ and let $F \subseteq \mathcal{L}_{At}^K$ be closed under propositional consequence. We define

$$A_T(F) = Cn(\{\varphi_F \mid \varphi \in T\}).$$

Clearly, A_T is a deterministic operator on $\mathcal{B}$. It is straightforward to represent it as a nondeterministic operator on $\mathcal{B}$ and view it as an element of $\mathcal{NO}_{\mathcal{B}}$.

A theory $F \subseteq \mathcal{L}_{At}$ is an *expansion* of T if $F = A_T(F)$, that is, a fixpoint of A_T. To be precise, more commonly expansions are defined as stable expansions of such propositional theories F (cf. (Marek & Truszczyński 1993; Denecker, Marek, & Truszczyński 2003)). However, as there is a one-to-one correspondence between the concepts, in this paper we regard as an expansion its propositional "generator".

Two theories $U, V \subseteq \mathcal{L}^K$ are *expansion-equivalent* relative to a class $\mathcal{T}$ of theories if for every theory $W \in \mathcal{T}$, $U \cup W$ and $V \cup W$ have the same expansions. Since expansions of a theory are defined as fixpoints of an operator on a boolean algebra, we will now use the results from the previous section to characterize the concept of expansion-equivalence.

We denote by $\mathcal{T}_{mf}$ the class of modal-free theories T that are closed under propositional consequence. We observe that the class of operators A_T given by theories $T \in \mathcal{T}_{mf}$ contains all constant operators on $\mathcal{B}$. Next, we note that $U \cup W$ and $V \cup W$ have the same expansions if and only if $A_{U \cup W}$ and $A_{V \cup W}$ have the same expansions. Moreover, $A_{U \cup W} = A_U \sqcup A_V$ and $A_{V \cup W} = A_V \sqcup A_V$. Finally, since A_U and A_V are deterministic, we note that $A_U(Y) =_Y A_V(Y)$ if and only if $A_U(Y) = A_V(Y)$. Thus, applying Corollary 4.2, we get the following result.

Theorem 5.1 *Let $U, V \subseteq \mathcal{L}^K$ and let $\mathcal{T}$ be a class of theories such that $\mathcal{T}_{mf} \subseteq \mathcal{T}$. Then, the following conditions are equivalent:*

1. *U and V are fp-equivalent relative to the class of all theories*
2. *U and V are fp-equivalent relative to $\mathcal{T}$*
3. *U and V are fp-equivalent relative to $\mathcal{T}_{mf}$*
4. *A_U and A_V have the same prefixpoints, and for every prefixpoint Y of A_U, $A_U(Y) = A_V(Y)$.*

6 Conclusions

Our results extend the concept of hyperequivalence to two other major semantics of logic programs: the supported-model semantics and the suppmin-model semantics. Our results for the semantics of suppmin models are only partial. The case when the restriction $a \leq b$ is weakened or dropped altogether remains open.

There are other variants of Theorem 4.3 concerned with sfp-equivalence relative to other classes of operators than $\mathcal{NO}_L(a, b)$. However, the results we have obtained so far are too weak to imply characterizations for the relativized hyperequivalence of autoepistemic theories under the semantics of expansions that are in the same time minimal models. This is one of the topics we are currently pursuing. It will receive a more complete treatment in the final version of the paper.

Our results can be phrased in an algebraic setting of nondeterministic operators on boolean algebras. In the case, when the operators are deterministic (but possibly partial) we obtain in this way a generalization of our results for normal logic programs with constraints. The generalization exploits the property that there is a direct match between fixpoints of deterministic (partial) operators and supported models of normal programs with constraints, based on the one-step provability operator of a program. However, our results are insufficient to be viewed as a generalization of the case of disjunctive logic programs with constraints. The reason is that there is no similar direct match between fixpoints of non-deterministic operators and supported models of disjunctive logic programs with constraints. Finding an algebraic characterization of hyperequivalence for the disjunctive case is another open problem.

The results we obtained for logic programs can be used to establish the complexity of problems to decide hyperequivalence with respect to particular program classes. This topic is the subject of an ongoing work and the results will be presented in another paper.

References

Apt, K.; Blair, H.; and Walker, A. 1988. Towards a theory of declarative knowledge. In Minker, J., ed., *Foundations of deductive databases and logic programming*, 89–142. Morgan Kaufmann.

Cabalar, P.; Odintsov, S. P.; Pearce, D.; and Valverde, A. 2006. Analysing and extending well-founded and partial stable semantics using partial equilibrium logic. In Etalle, S., and Truszczynski, M., eds., *Proceedings of the 22nd International Conference (ICLP 2006)*, volume 4079 of *LNCS*, 346–360. Springer.

Clark, K. 1978. Negation as failure. In Gallaire, H., and Minker, J., eds., *Logic and data bases*. New York-London: Plenum Press. 293–322.

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

Eiter, T.; Fink, M.; and Woltran, S. 2007. Semantical characterizations and complexity of equivalences in answer set programming. *ACM Transactions on Computational Logic* 8(3). 53 pages.

Eiter, T.; Tompits, H.; and Woltran, S. 2005. On solution correspondences in answer-set programming. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, 97–102. Morgan Kaufmann.

Inoue, K., and Sakama, C. 2004. Equivalence of logic programs under updates. In *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA-04)*, volume 3229 of *Lecture Notes in Computer Science*, 174–186. Springer.

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

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

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

McDermott, D., and Doyle, J. 1980. Nonmonotonic logic I. *Artificial Intelligence* 13(1-2):41–72.

Moore, R. 1984. Possible-world semantics for autoepistemic logic. In *Proceedings of the Workshop on Non-Monotonic Reasoning*, 344–354. Reprinted in: M. Ginsberg, ed., *Readings on Nonmonotonic Reasoning*, pages 137–142, Morgan Kaufmann, 1990.

Moore, R. 1985. Semantical considerations on nonmonotonic logic. *Artificial Intelligence* 25(1):75–94.

Oetsch, J.; Tompits, H.; and Woltran, S. 2007. Facts do not Cease to Exist Because They are Ignored: Relativised Uniform Equivalence with Answer-Set Projection. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI-2007)*, 458–464. AAAI Press.

Oikarinen, E., and Janhunen, T. 2006. Modular Equivalence for Normal Logic Programs. In *Proceedings of the 17th European Conference on Artificial Intelligence* (*ECAI 2006*), 412–416. IOS Press.

Truszczyński, M. 2006. Strong and uniform equivalence of nonmonotonic theories — an algebraic approach. In Doherty, P.; Mylopoulos, J.; and Welty, C., eds., *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006)*, 389–399. AAAI Press.

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

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

Woltran, S. 2007. A Common View on Strong, Uniform, and Other Notions of Equivalence in Answer-Set Programming. In Pearce, D.; Polleres, A.; Valverde, A.; and Woltran, S., eds., *Proceedings of the 1st Workshop Correspondence and Equivalence for Nonmonotonic Theories (CENT'07)*, volume 265 of *CEUR Workshop Proceedings*, 13–24. CEUR-WS.org.

Zhang, A., and Marek, W. 1989. On the classification and existence of structures in default logic. In *EPIA 89 (Lisbon, 1989)*, volume 390 of *Lecture Notes in Computer Science*, 129–140. Berlin-New York: Springer.