Proceedings, Eleventh International Conference on Principles of Knowledge Representation and Reasoning (2008)

Belief Revision of Logic Programs under Answer Set Semantics*

James Delgrande
School of Computing Science
Simon Fraser University
Burnaby, B.C.
Canada V5A 1S6

Torsten Schaub†
Institut für Informatik
Universität Potsdam
August-Bebel-Str. 89
D–14482 Potsdam, Germany

Hans Tompits and **Stefan Woltran**
Institut für Informationssysteme
Technische Universität Wien,
Favoritenstraße 9–11
A–1040 Vienna, Austria

Abstract

We address the problem of belief revision in (nonmonotonic) logic programming under answer set semantics: given logic programs P and Q, the goal is to determine a program R that corresponds to the revision of P by Q, denoted $P * Q$. Unlike previous approaches in logic programming, our formal techniques are analogous to those of distance-based belief revision in propositional logic. In developing our results, we build upon the model theory of logic programs furnished by SE models. Since SE models provide a formal, monotonic characterisation of logic programs, we can adapt well-known techniques from the area of belief revision to revision in logic programs. We investigate two specific operators: (logic program) expansion and a revision operator based on the distance between the SE models of logic programs. It proves to be the case that expansion is an interesting operator in its own right, unlike in classical AGM-style belief revision where it is relatively uninteresting. Expansion and revision are shown to satisfy a suite of interesting properties; in particular, our revision operators satisfy the majority of the AGM postulates for revision. A complexity analysis reveals that our revision operators do not increase the complexity of the base formalism. As a consequence, we present an encoding for computing the revision of a logic program by another, within the same logic programming framework.

Introduction

Answer set programming (ASP) (Baral 2003) has emerged as a major area of research in knowledge representation and reasoning (KRR). On the one hand, ASP has an elegant and conceptually simple theoretical foundation, while on the other hand efficient implementations of ASP solvers exist which have been finding application to practical problems. At its heart, ASP exploits negation as failure with respect to a fixed-point semantics; this enables the specification of a wide variety of problems. Consequently, ASP provides an appealing approach for representing problems in KRR.

Given that knowledge is continually evolving and always subject to change, there is also a need to be able to revise logic programs as new information is received. In KRR, the area of *belief revision* (Alchourrón, Gärdenfors, and Makinson 1985; Gärdenfors 1988) addresses just such change to a knowledge base. In *AGM* belief revision (named after the aforecited developers of the approach) one has a knowledge base K and a formula α, and the issue is how to consistently incorporate α in K to obtain a new knowledge base K'. The interesting case is when $K \cup \{\alpha\}$ is inconsistent, since beliefs have to be dropped from K before α can be consistently added. Hence a fundamental issue concerns how such change should be managed.

In classical propositional logic, specific belief revision operators have been proposed based on the distance between *models* of a knowledge base and a formula for revision. That is, a characterisation of the revision of a knowledge base K by formula α is to set the models of the revised knowledge base K' to be the models of α that are "closest" to those of K. Of course the notion of "closest" needs to be pinned down, but natural definitions based on the Hamming distance (Dalal 1988; Satoh 1988) are well known. Clearly, also the set of models of a knowledge base gives an abstract characterisation of the knowledge base, suppressing irrelevant syntactic details.

It is natural then to consider belief change in the context of logic programs. Indeed, there has been substantial effort in developing approaches to so-called logic program updating under answer set semantics (we discuss previous work in the next section). Unfortunately, given the nonmonotonic nature of answer set programs, the problem of change in logic programs has appeared to be intrinsically more difficult than in a monotonic setting.

In this paper, our goal is to reformulate belief change in logic programs in a manner analogous to belief revision in classical propositional logic, and to investigate specific belief revision operators for extended logic programs. Central for our approach are *SE models* (Turner 2003), which are semantic structures characterising *strong equivalence* between programs (Lifschitz, Pearce, and Valverde 2001). This particular kind of equivalence plays a major role for different problems in logic programming—in particular, in program simplifications and modularisation. This is due to the fact that strong equivalence gives rise to a *substitution principle* in the sense that, for all programs P, Q, $P \cup R$ and $Q \cup R$ have the same answer sets, for *any* program R. As is well known,

*This work was partially supported by the Austrian Science Fund (FWF) under grant P18019.

†Affiliated with Simon Fraser University, Canada, and Griffith University, Australia.

ordinary equivalence between programs (which holds if two programs have the same answer sets) does not yield a substitution principle. Hence, strong equivalence can be seen as the logic programming analogue of ordinary equivalence in classical logic. The important aspect of strong equivalence is that it coincides with equivalence in a specific *monotonic logic*, the logic of *here and there* (HT), which is intermediate between intuitionistic logic and classical logic. As shown by Turner (2003), equivalence between programs in HT corresponds in turn to equality between sets of SE models. Details on these concepts are given in the next section; the key point is that logic programs can be expressed in terms of a non-classical but monotonic logic, and it is this point that we exploit here.

More specifically, given this monotonic characterisation (via sets of SE models) of strong equivalence, we adapt techniques for revision in propositional logic to revision in logic programs. Hence we come up with specific operators for belief change in ASP analogous to operators in propositional logic. We first consider an *expansion* operator. In classical logic, the expansion of knowledge base K by formula α amounts to the deductive closure of $K \cup \{\alpha\}$. Hence it is not a very interesting operator, serving mainly as a tool for expressing concepts in belief revision and its dual, contraction. In logic programs however, expansion appears to be a more useful operator, perhaps due to the apparent "looser" notion of satisfiability provided by SE models. As well, it has appealing properties. We also develop a revision operator based on a notion of distance between SE models and show that it satisfies the majority of the corresponding AGM postulates. Curiously, in our approaches there is effectively no mention of answer sets; rather definitions of expansion and revision are given entirely with respect to logic programs. Notably too, our operators are syntax independent, which is to say, they are independent of how a logic program is expressed; hence, our operators deal with the *logical content* of a logic program.

Following an introductory background section, we show that there is a ready mapping between concepts in belief revision in classical logic and in ASP; this serves to place belief revision in ASP firmly in the "standard" belief revision camp. After this we describe our approaches to belief expansion and revision in ASP. The next section covers complexity issues and shows how we can in fact express the process of belief change in ASP. We conclude with a discussion. Proofs of our results are relegated to an appendix.

Background and Formal Preliminaries

Answer Set Programming

A (*generalised*) *logic program*[1] (GLP) over an alphabet $\mathcal{A}$ is a finite set of rules of the form

$$a_1; \dots; a_m; \sim b_{m+1}; \dots; \sim b_n \leftarrow c_{n+1}, \dots, c_o, \sim d_{o+1}, \dots, \sim d_p, \qquad (1)$$

where $a_i, b_j, c_k, d_l \in \mathcal{A}$ are *atoms*, for $1 \leq i \leq m \leq j \leq n \leq k \leq o \leq l \leq p$. Operators ';' and ',' express disjunctive and conjunctive connectives. A *default literal* is an atom a or its (default) negation $\sim a$. A rule r as in (1) is called a *fact* if $p = 1$, *normal* if $n = 1$, *positive* if $m = n$ and $o = p$, *disjunctive* if $m = n$, and an *integrity constraint* if $n = 0$, yielding an empty disjunction denoted by $\perp$. Accordingly, a program is called *disjunctive* (or a DLP), etc., if it consists of disjunctive, etc., rules only. We furthermore define $H(r) = \{a_1, \dots, a_m, \sim b_{m+1}, \dots, \sim b_n\}$ as the *head* of r and $B(r) = \{c_{n+1}, \dots, c_o, \sim d_{o+1}, \dots, \sim d_p\}$ as the *body* of r. Moreover, given a set X of literals, $X^+ = \{a \in \mathcal{A} \mid a \in X\}$, $X^- = \{a \in \mathcal{A} \mid \sim a \in X\}$, and $\sim X = \{\sim a \mid a \in X \cap \mathcal{A}\}$. For simplicity, we sometimes use a set-based notation, expressing a rule as in (1) as $H(r)^+; \sim H(r)^- \leftarrow B(r)^+, \sim B(r)^-$.

In what follows, we restrict ourselves to a finite alphabet $\mathcal{A}$. An interpretation is represented by the subset of atoms in $\mathcal{A}$ that are true in the interpretation. A (*classical*) *model* of a program P is an interpretation in which all of the rules in P are true according to the standard definition of truth in propositional logic, and where default negation is treated as classical negation. By $Mod(P)$ we denote the set of all classical models of P. An *answer set* Y of a program P is a subset-minimal model of

$$\{H(r)^+ \leftarrow B(r)^+ \mid r \in P, H(r)^- \subseteq Y, B(r)^- \cap Y = \emptyset\}.$$

The set of all answer sets of a program P is denoted by $AS(P)$. For example, the program $P = \{a \leftarrow, \quad c; d \leftarrow a, \sim b\}$ has answer sets $AS(P) = \{\{a, c\}, \{a, d\}\}$.

As defined by Turner (2003), an *SE interpretation* is a pair (X, Y) of interpretations such that $X \subseteq Y \subseteq \mathcal{A}$. An SE interpretation is an *SE model* of a program P if $Y \models P$ and $X \models P^Y$. The set of all SE models of a program P is denoted by $SE(P)$. Note that Y is an answer set of P iff $(Y, Y) \in SE(P)$ and no $(X, Y) \in SE(P)$ with $X \subset Y$ exists. Also, we have $(Y, Y) \in SE(P)$ iff $Y \in Mod(P)$.

A program P is *satisfiable* just if $SE(P) \neq \emptyset$. Two programs P and Q are *strongly equivalent*, symbolically $P \equiv_s Q$, iff $SE(P) = SE(Q)$. Alternatively, $P \equiv_s Q$ holds iff $AS(P \cup R) = AS(Q \cup R)$, for every program R (Lifschitz, Pearce, and Valverde 2001). We also write $P \subseteq_s Q$ iff $SE(P) \subseteq SE(Q)$. For simplicity, we often drop set-notation within SE interpretations and simply write, e.g., (a, ab) instead of $(\{a\}, \{a, b\})$.

A set S of SE interpretations is *well-defined* if, for each $(X, Y) \in S$, also $(Y, Y) \in S$. A well-defined set S of SE interpretations is *complete* if, for each $(X, Y) \in S$, also $(X, Z) \in S$, for any $Y \subseteq Z$ with $(Z, Z) \in S$. We have the following properties:

- For each GLP P, $SE(P)$ is well defined.
- For each DLP P, $SE(P)$ is complete.

Furthermore, for each well defined set S of SE interpretations, there exists a GLP P such that $SE(P) = S$, and for each complete set S of SE interpretations, there exists a DLP P such that $SE(P) = S$. Programs meeting these conditions can be constructed thus (Eiter, Tompits, and Woltran 2005; Cabalar and Ferraris 2007): In case S is a well-defined

[1]Such programs were first considered by Lifschitz and Woo (1992) and coined *generalised disjunctive logic programs* by Inoue and Sakama (1998).

set of SE interpretations over a (finite) alphabet $\mathcal{A}$, define P by adding

1. the rule $r_Y : \bot \leftarrow Y, \sim(\mathcal{A} \setminus Y)$, for each $(Y,Y) \notin S$, and
2. the rule $r_{X,Y} : (Y \setminus X); \sim Y \leftarrow X, \sim(\mathcal{A} \setminus Y)$, for each $X \subseteq Y$ such that $(X,Y) \notin S$ and $(Y,Y) \in S$.

In case S is complete, define P by adding

1. the rule r_Y, for each $(Y,Y) \notin S$, as above, and
2. the rule $r'_{X,Y} : (Y \setminus X) \leftarrow X, \sim(\mathcal{A} \setminus Y)$, for each $X \subseteq Y$ such that $(X,Y) \notin S$ and $(Y,Y) \in S$.

We call the resulting programs *canonical*.

For illustration, consider

$$S = \{(p,p), (q,q), (p,pq), (q,pq), (pq,pq), (\emptyset,p)\}$$

over $\mathcal{A} = \{p,q\}$. Note that S is not complete. The canonical GLP is as follows:

$$\begin{array}{rcrcl} r_\emptyset & : & \bot & \leftarrow & \sim p, \sim q; \\ r_{\emptyset,q} & : & q; \sim q & \leftarrow & \sim p; \\ r_{\emptyset,pq} & : & p; q; \sim p; \sim q & \leftarrow & . \end{array}$$

For obtaining a complete set, we have to add $(\emptyset, pq)$ to S. Then, the canonical DLP is as follows:

$$\begin{array}{rcrcl} r_\emptyset & : & \bot & \leftarrow & \sim p, \sim q; \\ r_{\emptyset,q} & : & q & \leftarrow & \sim p. \end{array}$$

One feature of SE models is that they contain "more information" than answer sets, which makes them an appealing candidate for problems where programs are examined with respect to further extension (in fact, this is what strong equivalence is about). We illustrate this issue with the following well-known example, involving programs

$$P = \{p; q \leftarrow\} \quad \text{and} \quad Q = \left\{ \begin{array}{c} p \leftarrow \sim q \\ q \leftarrow \sim p \end{array} \right\}.$$

Here, we have $AS(P) = AS(Q) = \{\{p\}, \{q\}\}$. However, the SE models (we list them for $\mathcal{A} = \{p,q\}$) differ:

$$SE(P) = \{(p,p), (q,q), (p,pq), (q,pq), (pq,pq)\};$$
$$SE(Q) = \{(p,p), (q,q), (p,pq), (q,pq), (pq,pq), (\emptyset,pq)\}.$$

This is to be expected, since P and Q behave differently with respect to program extension (and thus are not strongly equivalent). Consider $R = \{p \leftarrow q, q \leftarrow p\}$. Then $AS(P \cup R) = \{\{p,q\}\}$, while $AS(Q \cup R)$ has no answer set.

Belief Revision

The AGM Approach The best known and, indeed, seminal work in belief revision is the *AGM approach* (Alchourrón, Gärdenfors, and Makinson 1985; Gärdenfors 1988), in which standards for belief *revision* and *contraction* functions are given. In belief revision, a formula is added to a knowledge base such that the resulting knowledge base is consistent (unless the formula to be added is not). *Belief contraction* is a dual notion, in which information is removed from a knowledge base; given that it is of limited interest with respect to our approach, we do not consider it further. In the AGM approach it is assumed that a knowledge base is receiving information concerning a static[2] domain. Belief states are modeled by logically closed sets of sentences, called *belief sets*. A belief set is a set K of sentences which satisfies the constraint

$$\text{if } K \text{ logically entails } \beta, \text{ then } \beta \in K.$$

K can be seen as a partial theory of the world. For belief set K and formula α, $K + \alpha$ is the deductive closure of $K \cup \{\alpha\}$, called the *expansion* of K by α. $K_\bot$ is the inconsistent belief set (i.e., $K_\bot$ is the set of all formulas).

Subsequently, Katsuno and Mendelzon (1992) reformulated the AGM approach so that a knowledge base was also represented by a formula in some language $\mathcal{L}$. The following postulates comprise Katsuno and Mendelzon's reformulation of the AGM revision postulates, where $*$ is a function from $\mathcal{L} \times \mathcal{L}$ to $\mathcal{L}$:

R1: $\psi * \mu \vdash \mu$.

R2: If $\psi \wedge \mu$ is satisfiable, then $\psi * \mu \leftrightarrow \psi \wedge \mu$.

R3: If μ is satisfiable, then $\psi * \mu$ is also satisfiable.

R4: If $\psi_1 \leftrightarrow \psi_2$ and $\mu_1 \leftrightarrow \mu_2$, then $\psi_1 * \mu_1 \leftrightarrow \psi_2 * \mu_2$.

R5: $(\psi * \mu) \wedge \phi \vdash \psi * (\mu \wedge \phi)$.

R6: If $(\psi * \mu) \wedge \phi$ is satisfiable, then $\psi * (\mu \wedge \phi) \vdash (\psi * \mu) \wedge \phi$.

Thus revision is successful (R1), and corresponds to conjunction when the knowledge base and formula for revision are jointly consistent (R2). Revision leads to inconsistency only when the formula for revision is unsatisfiable (R3). Revision is also independent of syntactic representation (R4). Last, (R5) and (R6) express that revision by a conjunction is the same as revision by a conjunct conjoined with the other conjunct, when the result is satisfiable.

Specific Belief Revision Operators In classical belief change, the revision of a knowledge base represented by formula ψ by a formula μ, $\psi * \mu$, is a formula ϕ such that the models of ϕ are just those models of μ that are "closest" to those of ψ. There are two main specific approaches to distance-based revision. Both are based on the Hamming distance between two interpretations, that is on the set of atoms on which the interpretations disagree. The first, due to Dalal (1988), uses a distance measure based on the number of atoms with differing truth values in two interpretations. The second, by Satoh (1988), is based on set containment. A set containment-based approach seems more appropriate in the context of ASP, since answer sets are defined in terms of subset-minimal interpretations. Hence, we focus on Satoh (1988) here.

The *Satoh revision operator*, $\psi *_s \mu$, is defined as follows. Let $\triangle$ be the symmetric difference of two sets. For formulas α and β, define $\triangle^{min}(\alpha, \beta)$ as

$$min_\subseteq(\{w \triangle w' \mid w \in Mod(\alpha), w' \in Mod(\beta)\}).$$

[2] Note that "static" does not imply "with no mention of time". For example, one could have information in a knowledge base about the state of the world at different points in time, and revise information at these points in time.

Furthermore, define $Mod(\psi *_s \mu)$ as

$$\{w \in Mod(\mu) \mid \exists w' \in Mod(\psi) \text{ s.t. } w \triangle w' \in \triangle^{min}(\psi, \mu)\}.$$

Belief Change in Logic Programming

Most previous work on belief change for logic programs goes under the title of *update* (Foo and Zhang 1997; Przymusinski and Turner 1997; Zhang and Foo 1998; Alferes et al. 1998; 2000; Leite 2003; Inoue and Sakama 1999; Eiter et al. 2002; Zacarías et al. 2005; Delgrande, Schaub, and Tompits 2007). Strictly speaking, however, such approaches often do not address "update" as used in the belief revision community, in that the requirement that the underlying domain being modelled has changed is not taken into account. Following the investigations of the Lisbon group of researchers (Alferes et al. 1998; 2000; Leite 2003), a common feature of most update approaches is to consider a sequence $P_1, P_2, \ldots, P_n$ of programs where each P_i is a logic program. For P_i, P_j, $i > j$, the intuition is that P_i has higher priority or precedence. Given such a sequence, a set of answer sets is determined that in some sense respects the ordering. This may be done by translating the sequence into a "flat" logic program that contains an encoding of the priorities, or by treating the sequence as a prioritised logic program, or by some other appropriate method. The net result, one way or another, is to obtain a set of answer sets from such a program sequence, and not a single new program expressed in the language of the original logic programs. Hence, these approaches fall outside the general AGM belief revision paradigm.

However, various principles have been proposed for such approaches to logic program update. In particular, Eiter et al. (2002) consider the question of what principles the update of logic programs should satisfy. This is done by re-interpreting different AGM-style postulates for revising or updating classic knowledge bases, as well as introducing new principles. Among the latter, let us note the following:

Initialisation: $\emptyset * P \equiv P$.

Idempotency: $(P * P) \equiv P$.

Tautology: If Q is tautologous, then $P * Q \equiv P$.

Absorption: If $Q = R$, then $((P * Q) * R) \equiv (P * Q)$.

Augmentation: If $Q \subseteq R$, then $((P * Q) * R) \equiv (P * R)$.

In view of the failure of several of the discussed postulates in the approach of Eiter et al. (2002) (as well as in others), Osorio and Cuevas (2007) noted that for re-interpreting the standard AGM postulates in the context of logic programs, the logic underlying strong equivalence should be adopted. Since they studied programs with strong negation, in their case this logic is $\mathbf{N}_2$, an extension of HT by allowing strong negation.[3] They also introduced a new principle, which they called *weak independence of syntax* (WIS), which they proposed any update operator should satisfy:

WIS: If $Q \equiv_s R$, then $(P * Q) \equiv (P * R)$.

Indeed, following this spirit, the above absorption and augmentation principles can be accordingly changed by replacing their antecedents by "$Q \equiv_s R$" and "$Q \subseteq_s R$", respectively. We note that the WIS principle was also discussed in an update approach based on *abductive programs* (Zacarías et al. 2005).

Turning our attention to the few works on *revision* of logic programs, early work in this direction includes a series of investigations dealing with restoring consistency for programs possessing no answer sets (cf., e.g., Witteveen, van der Hoek, and de Nivelle (1994)). Other work uses logic programs under a variant of the stable semantics to specify database revision, i.e., the revision of knowledge bases given as sets of atomic facts (Marek and Truszczyński 1998). Finally, an approach following the spirit of AGM revision is discussed by Kudo and Murai (2004). In their work, they deal with the question of constructing revisions of form $P * A$, where P is an extended logic program and A is a conjunction of literals. They give a procedural algorithm to construct the revised programs; however no properties are analysed.

Belief Change in ASP based on SE Models

In AGM belief change, an agent's beliefs may be abstractly characterised in various different ways. In the classical AGM approach an agent's beliefs are given by a *belief set*, or deductively-closed set of sentences. As well, an agent's beliefs may also be characterised abstractly by a set of interpretations or *possible worlds*; these would correspond to models of the agent's beliefs. Last, as proposed in the Katsuno-Mendelzon formulation, and given the assumption of a finite language, an agent's beliefs can be specified by a formula. Given a finite language, it is straightforward to translate between these representations.

In ASP, there are notions analogous to the above for specifying an agent's beliefs. Though we do not get into it here, the notion of *strong equivalence* of logic programs can be employed to define a (*logic program*) *belief set*. Indeed, *SE models* characterise a class of equivalent logic programs. Hence the set of SE models of a program can be considered as the *proposition* expressed by the program. Continuing this analogy, a specific logic program can be considered to correspond to a formula or set of formulas in classical logic.

Belief Expansion in Logic Programs

Belief *expansion* is a belief change operator that is much more basic than revision or contraction, and in a certain sense is *prior* to revision and contraction (since in the AGM approach revision and contraction postulates make reference to expansion). Hence it is of interest to examine expansion from the point of view of logic programs. As well, it proves to be the case that expansion in logic programs is of interest in its own right.

The next definition corresponds model-theoretically with the usual definition of expansion in AGM belief change.

Definition 1 *For logic programs P and Q, define the* expansion *of P and Q, $P + Q$, to be a logic program R such that $SE(R) = SE(P) \cap SE(Q)$.*

[3] $\mathbf{N}_2$ itself traces back to an extension of intuitionist logic with strong negation first studied by Nelson (1949).

For illustration, consider the following examples:[4]

1. $\{p \leftarrow\} + \{\bot \leftarrow p\}$ has no SE models.
2. $\{p \leftarrow q\} + \{\bot \leftarrow p\}$ has SE model $(\emptyset, \emptyset)$.
3. $\{p \leftarrow\} + \{q \leftarrow p\} \equiv_s \{p \leftarrow\} + \{q \leftarrow\} \equiv_s \{p \leftarrow, q \leftarrow\}$.
4. $\{p \leftarrow \sim q\} + \{q \leftarrow \sim p\} \ \equiv_s \ \left\{ \begin{array}{l} p \leftarrow \sim q \\ q \leftarrow \sim p \end{array} \right\}$.
5. $\left\{ \begin{array}{l} p \leftarrow \sim q \\ q \leftarrow \sim p \end{array} \right\} + \{p \leftarrow q\} \ \equiv_s \ \left\{ \begin{array}{l} p \leftarrow q \\ p \leftarrow \sim q \end{array} \right\}$.
6. $\left\{ \begin{array}{l} p \leftarrow \sim q \\ q \leftarrow \sim p \end{array} \right\} + \{p; q \leftarrow\} \ \equiv_s \ \{p; q \leftarrow\}$.
7. $\{p; q \leftarrow\} + \{\bot \leftarrow q\} \ \equiv_s \ \left\{ \begin{array}{l} p \leftarrow \\ \bot \leftarrow q \end{array} \right\}$.
8. $\{p; q \leftarrow\} + \{\bot \leftarrow p, q\} \ \equiv_s \ \left\{ \begin{array}{l} p; q \leftarrow \\ \quad \bot \leftarrow p, q \end{array} \right\}$.

Belief expansion has desirable properties. The following all follow straightforwardly from the definition of expansion with respect to SE models.

Theorem 1 *Let P and Q be logic programs. Then:*

1. *$P + Q$ is a logic program* (*belief set*)*.*
2. *$P + Q \subseteq_s P$.*
3. *If $P \subseteq_s Q$, then $P + Q \equiv_s P$.*
4. *If $P \subseteq_s Q$, then $P + R \subseteq_s Q + R$.*
5. *If $SE(P)$ and $SE(Q)$ are defined, then so is $SE(P + Q)$.*
6. *If $SE(P)$ and $SE(Q)$ are complete, then so is $SE(P+Q)$.*
7. *If $Q \equiv_s \emptyset$, then $P + Q \equiv_s P$.*

While these results are indeed elementary, following as they do from the monotonicity of the SE interpretations framework, they are still of interest. Notably, virtually every previous approach to updating logic programs has trouble with the last property, expressing a *tautology* postulate. Here, expansion by a tautologous program presents no problem, as it corresponds to an intersection with the set of all SE interpretations. We note also that the other principles mentioned earlier—*initialisation*, *idempotency*, *absorption*, and *augmentation*—are trivially satisfied by expansion.

In classical logic, the expansion of two formulas can be given in terms of the intersection of their models. It should be clear from the preceding that the appropriate notion of the set of "models" of a logic program is given by a set of SE models, and not by a set of answer sets. Hence, there is no natural notion of expansion that is given in terms of answer sets. For instance, in Example 3, we have $AS(\{p \leftarrow\}) = \{\{p\}\}$ and $AS(\{q \leftarrow p\}) = \{\emptyset\}$ while $AS(\{p \leftarrow, q \leftarrow p\}) = \{\{p, q\}\}$. Likewise, in Example 4, the intersection of $AS(\{\{p \leftarrow \sim q\}\}) = \{\{p\}\}$ and $AS(\{\{q \leftarrow \sim p\}\}) = \{\{q\}\}$ is empty, whereas $AS(\{p \leftarrow \sim q,\ q \leftarrow \sim p\}) = \{\{p\}, \{q\}\}$.

[4] Unless otherwise noted, we assume that the language of discourse in each example consists of just the atoms mentioned.

Belief Revision

We next turn to a specific operator for belief revision. As discussed earlier, for a revision $P * Q$, we suggest that the most natural distance-based notion of revision for logic programs uses set containment as the appropriate means of relating SE interpretations. Hence, $P * Q$ is a logic program whose SE models are a subset of the SE models of Q, comprising just those models of Q that are closest to those of P. We note however that *any* reasonable notion of distance will do, for example an operator defined in terms of a cardinality-based distance measure.

To begin with, we extend the definition of symmetric difference so that it can be used with SE interpretations: If (X_1, X_2) and (Y_1, Y_2) are two SE interpretations, then $(X_1, X_2) \triangle (Y_1, Y_2)$ is defined as follows:

$$\begin{aligned} (X_1, X_2) \triangle (Y_1, Y_2) &= (X_1 \triangle Y_1, X_2 \triangle Y_2) \\ &= ((X_1 \setminus Y_1) \cup (Y_1 \setminus X_1), (X_2 \setminus Y_2) \cup (Y_2 \setminus X_2)). \end{aligned}$$

Similarly, $(X_1, X_2) \subseteq (Y_1, Y_2)$ iff $X_1 \subseteq Y_1$ and $X_2 \subseteq Y_2$, and moreover, $(X_1, X_2) \subset (Y_1, Y_2)$ iff $(X_1, X_2) \subseteq (Y_1, Y_2)$ and either $X_1 \subset Y_1$ or $X_2 \subset Y_2$.

Given this, we next define, for two sets E_1, E_2 of interpretations, the subset of E_1 that is closest to E_2, where the notion of "closest" is given in terms of symmetric difference.

Definition 2 *Let E_1, E_2 be two sets of either classical or SE interpretations. Then:*

$$\begin{aligned} \sigma(E_1, E_2) = \{A \in E_1 \mid \exists B \in E_2 \ \textit{such that} \\ \forall A' \in E_1, \forall B' \in E_2, A' \Delta B' \not\subset A \Delta B\}. \end{aligned}$$

It might seem that we could now define the SE models of $P * Q$ to be given by $\sigma(SE(Q), SE(P))$. However, for our revision operator to be meaningful, it must also produce a *well-defined* set of SE models. Unfortunately, it proves to be the case that Definition 2 does not preserve well-definedness. For an example, consider $P = \{\bot \leftarrow p\}$ and $Q = \{p \leftarrow \sim p\}$. Then, $SE(P) = \{(\emptyset, \emptyset)\}$ and $SE(Q) = \{(\emptyset, p), (p, p)\}$, and so $\sigma(SE(Q), SE(P)) = \{(\emptyset, p)\}$. However $\{(\emptyset, p)\}$ is not well-defined.

The problem is that for programs P and Q, there may be an SE model (X, Y) of Q with $X \subset Y$ such that $(X, Y) \in \sigma(SE(Q), SE(P))$ but $(Y, Y) \notin \sigma(SE(Q), SE(P))$. So, in defining $P * Q$ in terms of $\sigma(SE(Q), SE(P))$, we must modify the set $\sigma(SE(Q), SE(P))$ in some fashion to obtain a well-defined set of models.

In view of this, our approach is based on the following idea to obtain a well-defined set of models of $P * Q$ based on our notion of distance given in σ:

1. Determine the "closest" models of Q to P of form (Y, Y).
2. Determine the "closest" models of Q to P limited to models (X, Y) of Q where (Y, Y) was found in the first step.

Thus, we give preference to potential answer sets, in the form of models (Y, Y), and then to general models.

Definition 3 *For logic programs P and Q, define the* revision *of P by Q, $P * Q$, to be a logic program such that:*

if $SE(P) = \emptyset$, then $SE(P * Q) = SE(Q)$;

otherwise

$$SE(P * Q) = \{(X,Y) \mid Y \in \sigma(Mod(Q), Mod(P)) \\ \textit{and if } X \subset Y \textit{ then } (X,Y) \in \sigma(SE(Q), SE(P))\}.$$

As is apparent, $SE(P * Q)$ is well-defined, and thus is representable through a canonical logic program. Furthermore, over classical models, the definition of revision reduces to Satoh revision. As we show below, the result of revising P by Q is identical to that of expanding P by Q whenever P and Q possess common SE models. Hence, all previous examples of expansions (when the result is non-empty) are also valid program revisions. We have the following examples of revision that do not reduce to expansion.[5]

1. $\{p \leftarrow \sim p\} * \{\bot \leftarrow p\} \equiv_s \{\bot \leftarrow p\}$.
   Over the language $\{p, q\}$, $\bot \leftarrow p$ has SE models $(\emptyset, \emptyset)$, $(\emptyset, q)$, and (q, q).
2. $\left\{ \begin{array}{l} p \leftarrow \\ q \leftarrow \end{array} \right\} * \{\bot \leftarrow q\} \equiv_s \left\{ \begin{array}{l} p \leftarrow \\ \bot \leftarrow q \end{array} \right\}$.
   The first program has a single SE model, (pq, pq), while the second has three, $(\emptyset, \emptyset)$, $(\emptyset, p)$, and (p, p). Among the latter, (p, p) has the least pairwise symmetric difference to (pq, pq). The program induced by the singleton set $\{(p, p)\}$ of SE models is $\{p \leftarrow, \ \bot \leftarrow q\}$.
3. $\left\{ \begin{array}{l} p \leftarrow \\ q \leftarrow \end{array} \right\} * \{\bot \leftarrow p, q\} \equiv_s \left\{ \begin{array}{l} p; q \leftarrow \\ \quad \bot \leftarrow p, q \end{array} \right\}$.
   Thus, if one originally believes that p and q are true, and revises by the fact that one is false, then the result is that precisely one of p, q is true.
4. $\left\{ \begin{array}{l} \bot \leftarrow \sim p \\ \bot \leftarrow \sim q \end{array} \right\} * \{\bot \leftarrow p, q\} \equiv_s \left\{ \begin{array}{l} \bot \leftarrow \sim p, \sim q \\ \bot \leftarrow p, q \end{array} \right\}$.
   Observe that the classical models in the programs here are exactly the same as above. This example shows that the use of SE models provides finer "granularity" compared to using classical models of programs together with known revision techniques.
5. $\left\{ \begin{array}{l} \bot \leftarrow p \\ \bot \leftarrow q \end{array} \right\} * \{p; q \leftarrow\} \equiv_s \left\{ \begin{array}{l} p; q \leftarrow \\ \quad \bot \leftarrow p, q \end{array} \right\}$.

We next rephrase the Katsuno-Mendelzon postulates for belief revision. Here, $*$ is a function from ordered pairs of logic programs to logic programs.

RA1: $P * Q \subseteq_s Q$.

RA2: If $P + Q$ is satisfiable, then $P * Q \equiv_s P + Q$.

RA3: If Q is satisfiable, then $P * Q$ is satisfiable.

RA4: If $P_1 \equiv_s P_2$ and $Q_1 \equiv_s Q_2$, then $P_1 * Q_1 \equiv_s P_2 * Q_2$.

RA5: $(P * Q) + R \subseteq_s P * (Q + R)$.

RA6: If $(P * Q) + R$ is satisfiable, then $P * (Q + R) \subseteq_s (P * Q) + R$.

We obtain that logic program revision as given in Definition 3 satisfies the first five of the revision postulates.[6]

[5] Note that $\{p \leftarrow \sim p\}$ has SE models but no answer sets.

[6] We note in passing that this is analogous to set-containment based approaches in propositional logic.

Theorem 2 *The logic program revision operator $*$ from Definition 3 satisfies postulates RA1 – RA5.*

That our revision operator does not satisfy RA6 can be seen by the following example: Consider

$$\begin{aligned} P &= \{p; \sim p, \ q \leftarrow p, \ r \leftarrow p, \ s \leftarrow p, \ \bot \leftarrow \sim p, q, \\ &\qquad \bot \leftarrow \sim p, r, \ \bot \leftarrow \sim p, s\}, \\ Q &= \{p; r, \ \bot \leftarrow q, \ \bot \leftarrow p, r, \ \bot \leftarrow p, s, \ s; \sim s \leftarrow r\}, \\ R &= \{p; r, \ \bot \leftarrow q, \ \bot \leftarrow p, r, \ \bot \leftarrow p, s, \ s \leftarrow r\}. \end{aligned}$$

Straightforward computations show that

$$\begin{array}{rcll} SE(P * (Q + R)) & = & \{(rs, rs), (p, p)\} & \quad \text{while} \\ SE((P * Q) + R) & = & \{(p, p)\}. & \end{array}$$

So, $P*(Q+R) \not\subseteq_s (P*Q)+R$. Since $SE((P*Q)+R) \neq \emptyset$, this shows that RA6 indeed fails.

Last, we have the following result concerning other principles for updating logic programs listed earlier:

Theorem 3 *Let P and Q be logic programs. Then, $P * Q$ satisfies initialisation, idempotency, tautology, and absorption with respect to strong equivalence.*

Augmentation however does not hold, nor would one expect it to hold in a distance-based approach. For example, consider the case where P, Q, and R are characterised by models $SE(P) = \{(a, a), (ab, ab)\}$, $SE(Q) = \{(ab, ab), (ac, ac), (b, b)\}$, and $SE(R) = \{(ac, ac), (b, b)\}$. Thus $SE(R) \subseteq SE(Q)$ and so $Q \subseteq R$ for the underlying programs. We obtain that $SE(P * Q) = SE(P + Q) = \{(ab, ab)\}$, and thus $SE((P * Q) * R) = \{(b, b)\}$. However $SE(P * R) = \{(b, b), (c, c)\}$, contradicting augmentation.

Definition 3 is certainly not the only possibility to construct a revision operator. Let us now briefly discuss an alternative definition for revision.

Definition 4 *For logic programs P and Q, define the* weak revision *of P by Q to be a logic program $P *_w Q$ such that:*

$$\textit{if } SE(P) = \emptyset, \textit{ then } SE(P *_w Q) = SE(Q);$$

otherwise

$$SE(P *_w Q) = \sigma(SE(Q), SE(P)) \cup \\ \{(Y, Y) \mid \exists X \textit{ s.t. } (X, Y) \in \sigma(SE(Q), SE(P))\}.$$

The main drawback to this approach is that it introduces possibly irrelevant interpretations in order to obtain well-definedness. As well, Definition 3 appears to be the more natural. Consider the following example, which also serves to distinguish Definition 3 from Definition 4. Let

$$\begin{array}{rcl} P & = & \{\bot \leftarrow p, \ \bot \leftarrow q, \ \bot \leftarrow r\}, \\ Q & = & \{ \ r, \ \ p \leftarrow q, \ \ p \leftarrow \sim q \ \}. \end{array}$$

Then, we get the following SE models:

$$\begin{array}{rcl} SE(P) & = & \{(\emptyset, \emptyset)\}, \\ SE(Q) & = & \{(r, pqr), (pr, pr), (pr, pqr), (pqr, pqr)\}, \end{array}$$

and

$$\begin{array}{rcl} SE(P * Q) & = & \{(pr, pr)\}, \\ SE(P *_w Q) & = & SE(Q) \setminus \{(pr, pqr)\}. \end{array}$$

Consequently, $P*Q$ is given by the program $\{p, \bot \leftarrow q, r\}$. Thus, in this example, $P*Q$ gives the desired result, preserving the falsity of q from P, while incorporating the truth of r and p from Q. This then reflects the assumption of minimal change to the program being revised, in this case P. $P *_w Q$ on the other hand represents a very cautious approach to program revision.

Finally, we have that our definition of revision is strictly stronger than the alternative given by $*_w$:

Theorem 4 *Let P and Q be programs. Then, $P * Q \subseteq_s P *_w Q$.*

For completeness, let us mention that enforcing well-definedness by simply determine the "closest" models of Q to P of form (Y, Y) is inadequate. For our motivating example, we would obtain $SE(\{p \leftarrow \sim p\} * \{\bot \leftarrow p\}) = \emptyset$, violating the key postulate RA3, that the result of revising by a satisfiable program results in a satisfiable revision.

Computational Aspects

We first consider the worst-case complexity of our approach to revision. The standard decision problem for revision in classical logic is: Given formulas P, Q, R, does $P*Q$ entail R? Eiter and Gottlob (1992) showed that approaches to classical propositional revision are Π_2^P-complete. The next result shows that this property carries over to our approach for program revision.

Theorem 5 *Deciding whether $P * Q \subseteq_s R$ holds, for given GLPs P, Q, R, is Π_2^P-complete. Moreover, hardness holds already for P being a set of facts, Q being positive or normal, and R being a single fact.*

Although we do not show it here, we mention that the same results holds for the cautious revision operator $*_w$.

It is not difficult to come up with an algorithm implementing our approaches to expansion and revision: given programs P and Q, the set of SE models of each can be generated straightforwardly (Turner 2003). The resulting SE models for expansion or revision can be determined by an appropriate implementation of Definition 1 or 3. Then, given the resulting set of SE models, a corresponding GLP can be determined using the method of canonical programs.

Rather, our interest now is to consider the question of computing revisions more abstractly. We address the following issue: Can we find an encoding schema S such that, for all programs P, Q, there is a one-to-one correspondence between the answer sets of the program $S[P,Q]$ and elements in $SE(P*Q)$? By our complexity result, efficient construction of $S[P,Q]$, given P, Q, is possible, although disjunction is required in $S[P,Q]$.

It is well known how classical models or SE models can be characterized by means of answer sets (see, e.g., Eiter et al. (2004)). However, the encodings of the checks for containment in $\sigma(\cdot,\cdot)$ are a bit cumbersome. Therefore, instead of a full formal proof, we introduce $S[P,Q]$ step-by-step and describe the functioning of the different parts in some detail. Basically, the programs follows the argumentation used in the membership part of the proof of Theorem 5.

In what follows, we make use of the universe $\mathcal{A}$, but mention that for $S[P,Q]$, $\mathcal{A}$ can always be set to $var(P \cup Q)$. Moreover, we need to make several copies of $\mathcal{A}$: Therefore, for $j \in \{1,\ldots,5\}$ and $w \in \{h,t\}$, denote by $\mathcal{A}_w^j$ the set $\{a_w^j \mid a \in \mathcal{A}\}$, and by $\bar{\mathcal{A}}_w^j$ the set $\{\bar{a}_w^j \mid a \in \mathcal{A}\}$. All these new atoms are mutually distinct. The role of these sets in the subsequent encoding is that, for each j, $\mathcal{A}_h^j$ together with $\mathcal{A}_t^j$ are used to guess two sets X (via $\mathcal{A}_h^j$) and Y (via $\mathcal{A}_t^j$) which are then checked for being an SE model (X,Y) and for further properties. The sets $\bar{\mathcal{A}}_h^j$ and $\bar{\mathcal{A}}_t^j$ are used to support the guess as usual. The superscript j will allow us to deal with several SE interpretations at once in a single program.

We also need a corresponding renaming schema for the rules from the original programs P and Q. In what follows, r_w^j denotes the rule r after replacing each atom a by a_w^j. Accordingly, $\bar{r}_w^j$ replaces atoms a by $\bar{a}_w^j$.

Finally, to link arbitrary interpretations over $I \subseteq \bigcup_j \mathcal{A}_h^j \cup \mathcal{A}_t^j$ back to SE interpretations over $\mathcal{A}$, we use the following mappings: For an interpretation I and an index j, let $\pi^j(I) = \{(X,Y) \mid X, Y \subseteq \mathcal{A}, X_h^j = I \cap \mathcal{A}_h^j, Y_t^j = I \cap \mathcal{A}_t^j\}$, and, for a set $\mathcal{I}$ of interpretations, let $\Pi^j(\mathcal{I}) = \bigcup_{I \in \mathcal{I}} \pi^j(I)$. We define a first module as follows:

$$\begin{aligned} M[P,j] = \{ & a_w^j ; \bar{a}_w^j \leftarrow,\ \bot \leftarrow a_w^j, \bar{a}_w^j,\ \bot \leftarrow a_h^j, \bar{a}_t^j \mid \\ & a \in \mathcal{A}, w \in \{h,t\}\} \cup \\ & \{\bot \leftarrow H^+(\bar{r}_t^j), H^-(r_t^j), B^+(r_t^j), B^-(\bar{r}_t^j), \\ & \bot \leftarrow H^+(\bar{r}_h^j), H^-(r_t^j), B^+(r_h^j), B^-(\bar{r}_t^j) \mid r \in P\}. \end{aligned}$$

Then, we have for any program P and any index j,

$$\Pi^j(AS(M[P,j])) = SE(P).$$

To avoid an additional module for classical models, we will sometimes use SE models (U,V) where only the V-part comes into play. Our goal now is to filter those $(X_1,Y_1) \in SE(Q)$ such that $(X_1,Y_1) \in SE(P*Q)$. To this end, we first compute all possible combinations $(X_1,Y_1) \in SE(Q)$, $(X_2,Y_2) \in SE(P)$, and $(X_3,Y_3) \in SE(P)$ (via $M[Q,1]$, $M[P,2]$, $M[P,3]$) and then check: (i) whether for each further pairs of SE models $(X_4,Y_4) \in SE(Q)$, $(X_5,Y_5) \in SE(P)$ it holds that $Y_4 \triangle Y_5 \not\subset Y_1 \triangle Y_2$ and $(X_4,Y_4) \triangle (X_5,Y_5) \not\subset (X_1,Y_1) \triangle (X_3,Y_3)$ (this is just along the lines of Definition 3). Our second module is used to guess such further pairs $(X_4,Y_4) \in SE(Q)$ and $(X_5,Y_5) \in SE(P)$. However, compared to $M[P,j]$, we now use a spoiling technique rather than constraints to exclude SE interpretations which are not SE models of the respective program. This spoiling technique is important in the final program, which has to ensure that no such further pair (X_4,Y_4), (X_5,Y_5) exists satisfying $Y_4 \triangle Y_5 \subset Y_1 \triangle Y_2$ or $(X_4,Y_4) \triangle (X_5,Y_5) \subset (X_1,Y_1) \triangle (X_3,Y_3)$.

We use the same renaming concepts as before plus a further new atom z, and define:

$$\begin{aligned} N[P,j] = \{ & a_w^j ; \bar{a}_w^j \leftarrow,\ z \leftarrow a_w^j, \bar{a}_w^j,\ a_w^j \leftarrow z,\ \bar{a}_w^j \leftarrow z, \\ & z \leftarrow a_h^j, \bar{a}_t^j \mid a \in \mathcal{A}, w \in \{h,t\}\} \cup \\ & \{\bot \leftarrow H^+(\bar{r}_t^j), H^-(r_t^j), B^+(r_t^j), B^-(\bar{r}_t^j), \\ & \bot \leftarrow H^+(\bar{r}_h^j), H^-(r_t^j), B^+(r_h^j), B^-(\bar{r}_t^j) \mid r \in P\}\}. \end{aligned}$$

Instead of answer sets, we investigate the classical models of $N[P,j]$ (over $var(N[P,j])$): First, we have that the spoiled interpretation $O^j = \{z\} \cup \bigcup_{w\in\{h,t\}} \mathcal{A}^j_w$ is a model of $N[P,j]$. The remaining models are in relation to the SE models again, i.e., $\Pi^j(Mod(N[P,j]) \setminus O^j) = SE(P)$.

We need two final modules to compare: (i) $Y_4 \triangle Y_5$ with $Y_1 \triangle Y_2$; (ii) $(X_4,Y_4)\triangle(X_5,Y_5)$ with $(X_1,Y_1)\triangle(X_3,Y_3)$. Let us first give the comparison module for (i): The basic idea hereby is as follows: If $Y_4 \triangle Y_5 \not\subset Y_1 \triangle Y_2$ holds, we derive the dedicated atom z, already used in modules $N[\cdot,\cdot]$:

$$
\begin{aligned}
C_1 = \{ & z \leftarrow a^1_t, a^2_t, a^4_t, \bar{a}^5_t, \quad z \leftarrow a^1_t, a^2_t, \bar{a}^4_t, a^5_t,\\
& z \leftarrow \bar{a}^1_t, \bar{a}^2_t, a^4_t, \bar{a}^5_t, \quad z \leftarrow \bar{a}^1_t, \bar{a}^2_t, \bar{a}^4_t, a^5_t,\\
& a^\delta_t \leftarrow a^1_t, \bar{a}^2_t, a^4_t, \bar{a}^5_t,\ a^\delta_t \leftarrow \bar{a}^1_t, a^2_t, a^4_t, \bar{a}^5_t,\\
& a^\delta_t \leftarrow a^1_t, \bar{a}^2_t, \bar{a}^4_t, a^5_t,\ a^\delta_t \leftarrow \bar{a}^1_t, a^2_t, \bar{a}^4_t, a^5_t,\\
& a^\delta_t \leftarrow a^1_t, a^2_t, a^4_t, a^5_t,\ a^\delta_t \leftarrow a^1_t, a^2_t, \bar{a}^4_t, \bar{a}^5_t,\\
& a^\delta_t \leftarrow \bar{a}^1_t, \bar{a}^2_t, a^4_t, a^5_t,\ a^\delta_t \leftarrow \bar{a}^1_t, \bar{a}^2_t, \bar{a}^4_t, \bar{a}^5_t \mid a \in \mathcal{A}\}\\
\cup \{ & z \leftarrow \mathcal{A}^\delta_t \quad \},
\end{aligned}
$$

where $\mathcal{A}^\delta_t$ is a set of new atoms. The appearance of set $\mathcal{A}^\delta_t$ in a rule body stands for the sequence of all its elements.

The second comparison module C_2 is obtained from C_1 as follows: replace each atom a^2_t (resp., $\bar{a}^2_t$) by a^3_t (resp., $\bar{a}^3_t$); make a copy of each rule except $z \leftarrow \mathcal{A}^\delta_t$ and exchange in the copy each subscript t by h; finally, replace $z \leftarrow \mathcal{A}^\delta_t$ by $z \leftarrow \mathcal{A}^\delta_h, \mathcal{A}^\delta_t$.

Now it can be observed that z is derived for a guess of (X_4,Y_4), (X_5,Y_4) if neither $Y_4 \triangle Y_5 \subset Y_1 \triangle Y_2$ nor $(X_4,Y_4)\triangle(X_5,Y_5) \subset (X_1,Y_1)\triangle(X_3,Y_3)$. If this is the case for all such guesses, we get that $(X_1,Y_1) \in SE(P*Q)$ and thus the corresponding answer set should survive. On the other hand if some guess does not require w to be in the model, the corresponding answer set for (X_1,Y_1) should not survive. Due to the spoiling technique, this behaviour is exactly matched by adding a single constraint $\bot \leftarrow {\sim}z$. Thus, we put our modules as follows together:

$$
\begin{aligned}
S[P,Q] \quad = \quad & M[Q,1] \cup M[P,2] \cup M[P,3] \cup N[Q,4] \cup \\
& N[P,5] \cup C_1 \cup C_2 \cup \{\bot \leftarrow {\sim}z\},
\end{aligned}
$$

and obtain as result:

Theorem 6 *For all programs P and Q, $SE(P*Q) = \Pi^1(AS(S[P,Q]))$.*

Discussion

We have addressed the problem of belief revision in logic programming under the answer set semantics. Our approach is based on a monotonic characterisation of logic programs, given in terms of the set of SE models of a program. Based on the latter, we defined and examined operators for logic program expansion and revision. As well as giving properties of these operators, we also considered the complexity and an encoding scheme for revision. This work is novel, in that it addresses belief change in terms familiar to researchers in belief revision: expansion is characterised in terms of intersections of models, and revision is characterized in terms of minimal distance between models. While we considered set-containment-based revision here, cardinality-based revision can be defined also. In future work we will consider more general notions of distance; as well we will separately address the issue of general characterisations and representation results for logic programs, again via SE models and the logic of here and there.

We finally note that previous work on logic program revision was formulated at the level of the individual program, and not in terms of an abstract characterisation (via strong equivalence or sets of SE interpretations). The net result is that such previous work is generally difficult to work with: properties are difficult to come by, and often desirable properties (such as *tautology*) are lacking. The main point of departure for the current approach then is to lift the problem of logic program revision from the program (or syntactic) level to an abstract (or semantic) level.

Appendix

Proof of Theorem 1

Most of the parts follow immediately from the fact that $SE(P+Q) = SE(P) \cap SE(Q)$.

1. We need to show that Definition 1 results in a well-defined set of SE models.

   For $SE(P) \cap SE(Q) = \emptyset$ we have that $\emptyset$ is trivially well-defined (and R can be given by $\bot \leftarrow$).

   Otherwise, for $SE(P) \cap SE(Q) \neq \emptyset$, we have the following: If $(X,Y) \in SE(P) \cap SE(Q)$, then $(X,Y) \in SE(P)$ and $(X,Y) \in SE(Q)$; whence $(Y,Y) \in SE(P)$ and $(Y,Y) \in SE(Q)$ since $SE(P)$ and $SE(Q)$ are well-defined by virtue of P and Q being logic programs. Hence, $(Y,Y) \in SE(P) \cap SE(Q)$. Since this holds for arbitrary $(X,Y) \in SE(P) \cap SE(Q)$ we have that $SE(P) \cap SE(Q)$ is well-defined.
2. Immediate from the definition of $+$.
3. If $P \subseteq_s Q$, then $SE(P) \subseteq SE(Q)$. Hence, $SE(P) \cap SE(Q) = SE(P)$, or $P + Q \equiv_s P$.
4. Similar to the previous part.
5. This was established in the first part.
6. To show completeness, we need to show that for any $(X,Y) \in SE(P+Q)$ and $(Y \cup Y', Y \cup Y') \in SE(P+Q)$ that $(X, Y \cup Y') \in SE(P+Q)$.

   If $(X,Y) \in SE(P+Q)$ and $(Y \cup Y', Y \cup Y') \in SE(P+Q)$, then $(X,Y) \in SE(P) \cap SE(Q)$ and $(Y \cup Y', Y \cup Y') \in SE(P) \cap SE(Q)$. Hence, $(X,Y) \in SE(P)$ and $(Y \cup Y', Y \cup Y') \in SE(P)$, and so, since $SE(P)$ is complete by assumption, we have $(X, Y \cup Y') \in SE(P)$. The same argument gives that $(X, Y \cup Y') \in SE(Q)$, whence $(X, Y \cup Y') \in SE(P) \cap SE(Q)$ and $(X, Y \cup Y') \in SE(P+Q)$.
7. If $Q \equiv_s \emptyset$, then $SE(Q) = \{(X,Y) \mid X \subseteq Y \subseteq \mathcal{A}\}$ from which the result follows immediately. $\Box$

Proof of Theorem 2

RA1: This postulate is an immediate consequence of Definition 3. Note that $(X,Y) \in SE(P*Q)$ only if $Y \in \sigma(Mod(Q), Mod(P))$. Therefore, we get that $(Y,Y) \in \sigma(SE(Q), SE(P))$. So, $SE(P*Q)$ is well-defined.

RA2: If $P + Q$ is satisfiable, then $\sigma(Mod(Q), Mod(P)) \neq \emptyset$ and $\sigma(SE(Q), SE(P)) \neq \emptyset$.

Further, for $Y \in Mod(Q)$ (or $(X,Y) \in SE(Q)$) we have that there is $Y' \in Mod(P)$ (resp., $(X',Y') \in SE(P)$) such that $Y \Delta Y' = \emptyset$ ($(X,Y)\Delta(X',Y') = \emptyset$), from which our result follows.

RA3: From Definition 3 we have that, if P is unsatisfiable, then Q is satisfiable iff $P * Q$ is satisfiable.

Otherwise, if P is satisfiable and Q is satisfiable, then there is some $(Y,Y) \in \sigma(Mod(Q), Mod(P))$ (since $SE(Q)$ is well-defined and given Definition 2). Hence, $SE(P * Q) \neq \emptyset$.

RA4: Immediate from Definition 3.

RA5: If $SE(P) = \emptyset$, then the result follows immediately from the first part of Definition 3.

Otherwise, we show that, if (X,Y) is an SE model of $(P*Q)+R$, then (X,Y) is an SE model of $P*(Q+R)$.

Let $(X,Y) \in SE((P*Q)+R)$. Then, $(X,Y) \in SE(P*Q)$ and $(X,Y) \in SE(R)$. Since $(X,Y) \in SE(P*Q)$, by RA1 we have that $(X,Y) \in SE(Q)$, and so $(X,Y) \in SE(Q) \cap SE(R)$, or $(X,Y) \in SE(Q+R)$.

There are two cases to consider:

$X = Y$: Since $(X,Y) = (Y,Y) \in SE(P*Q)$, we have that $Y \in \sigma(Mod(Q), Mod(P))$. Hence, from Definition 2, $Y \in Mod(Q)$ and there is some $Y' \in Mod(P)$ such that there is no $Y_1 \in Mod(Q)$ and no $Y_2 \in Mod(P)$ such that $Y_1 \Delta Y_2 \subset Y \Delta Y'$.

We established at the outset that $(X,Y) \in SE(Q+R)$. Hence, $Y \in Mod(Q+R)$. This gives us that $Y \in Mod(Q+R)$ and there is some $Y' \in Mod(P)$ such that no Y_1, Y_2 exist with $Y_1 \in Mod(Q)$, $Y_2 \in Mod(P)$, and $Y_1 \Delta Y_2 \subset Y \Delta Y'$.

Clearly, in the above, if there is no $Y_1 \in Mod(Q)$ such that the above condition holds, then there is no $Y_1 \in Mod(Q+R)$ such that the above condition holds.

Thus, we have $Y \in Mod(Q+R)$ and there is some $Y' \in Mod(P)$ for which no $Y_1 \in Mod(Q+R)$ and no $Y_2 \in Mod(P)$ exists such that $Y_1 \Delta Y_2 \subset Y \Delta Y'$.

Thus, from Definition 2, we get $Y \in \sigma(Mod(Q+R), Mod(P))$, hence $(Y,Y) \in SE(P*(Q+R))$.

$X \subset Y$: We have $Y \in \sigma(Mod(Q), Mod(P))$ by virtue of $(X,Y) \in SE(P*Q)$. In the previous part we established that $Y \in \sigma(Mod(Q+R), Mod(P))$.

As well, $(X,Y) \in \sigma(SE(Q), SE(P))$ since $(X,Y) \in SE(P*Q)$. Thus, from Definition 2, we have that there is some $(X',Y') \in SE(P)$ such that no U, V, U', V' exist such that $(U,V) \in SE(Q)$, $(U',V') \in SE(P)$, and $(U,V)\Delta(U',V') \subset (X,Y)\Delta(X',Y')$.

Therefore, there is no $(U,V) \in SE(Q+R)$ and no $(U',V') \in SE(P)$ such that $(U,V)\Delta(U',V') \subset (X,Y)\Delta(X',Y')$.

We previously showed that $(X,Y) \in SE(Q+R)$. Consequently, from Definition 3, we obtain that $(X,Y) \in \sigma(SE(Q+R), SE(P))$. Hence, $(X,Y) \in SE(P*(Q+R))$.

Thus, in either case, we get $(X,Y) \in SE(P*(Q+R))$, which was to be shown. □

Proof of Theorem 3

For initialisation, idempotency, and tautology, in the left-hand side of the given equivalence, revision corresponds with expansion via RA2, from which the result is immediate.

For absorption, we have $Q = R$, and so $((P*Q)*R) = ((P*Q)*Q)$. Since $SE(P*Q) \subseteq SE(Q)$, then from Theorem 1, Part 3, we have that $(P*Q)+Q \equiv_s P*Q$. As well, $((P*Q)*Q) = ((P*Q)+Q)$, from which our result follows. □

Proof of Theorem 4

We need to show that $SE(P*Q) \subseteq SE(P*_w Q)$. First of all, if $SE(P) = \emptyset$, then $SE(P*Q) = SE(Q) = SE(P*_w Q)$. Otherwise, there are two cases to consider:

1. $(X,Y) \in SE(P*Q)$, where $X \subset Y$. By Definition 3, we get $(X,Y) \in \sigma(SE(P), SE(Q))$, and, by Definition 4, $(X,Y) \in SE(P*_w Q)$.
2. $(Y,Y) \in SE(P*Q)$. From Definition 3, we have that $Y \in \sigma(Mod(Q), Mod(P))$. $Y \in \sigma(Mod(Q), Mod(P))$ implies that $(Y,Y) \in \sigma(SE(Q), SE(P))$. Hence, according to Definition 4, $(Y,Y) \in SE(P*_w Q)$.

Therefore, $(X,Y) \in SE(P*Q)$ implies that $(X,Y) \in SE(P*_w Q)$, whence $SE(P*Q) \subseteq SE(P*_w Q)$. □

Proof of Theorem 5

Since we deal with a globally fixed language, we first need a few lemmata.

Lemma 1 *Let P, Q be programs, Y an interpretation, and $x \in Y \setminus var(P \cup Q)$. Then, $Y \in \sigma(Mod(Q), Mod(P))$ implies $Y \setminus \{x\} \in \sigma(Mod(Q), Mod(P))$.*

Proof. Since $Y \in \sigma(Mod(Q), Mod(P))$, so $Y \in Mod(Q)$ and there exists some $Z \in Mod(P)$ such that, for each $Y' \in Mod(Q)$ and $Z' \in Mod(P)$, $Y' \triangle Z' \not\subset Y \triangle Z$. We show that $x \in Z$ holds. Suppose this is not the case: Then, $x \in Y \triangle Z$, since $x \in Y$. Now, since $x \notin var(P)$, also $Z \cup \{x\} \in Mod(P)$. But then $x \notin Y \triangle (Z \cup \{x\})$ which yields $Y \triangle (Z \cup \{x\}) \subset Y \triangle Z$, a contradiction to our assumption. Hence, $x \in Z$. Now, since $Y \in Mod(Q)$, obviously $Y \setminus \{x\} \in Mod(Q)$ as well. We obtain $Y \triangle Z = (Y \setminus \{x\}) \triangle (Z \setminus \{x\})$, thus $Y \setminus \{x\} \in \sigma(Mod(Q), Mod(P))$ holds. □

Lemma 2 *Let P, Q be programs, let (X,Y) be an SE interpretation, and assume $x \in Y \setminus var(P \cup Q)$. Then, $(X,Y) \in \sigma(SE(Q), SE(P))$ implies $(X \setminus \{x\}, Y \setminus \{x\}) \in \sigma(SE(Q), SE(P))$.*

Proof. Since $(X,Y) \in \sigma(SE(Q), SE(P))$, we have that $(X,Y) \in SE(Q)$ and there exists some $(U,Z) \in SE(P)$ such that, for each $(X',Y') \in SE(Q)$ and $(U',Z') \in SE(P)$, $(X',Y') \triangle (U',Z') \not\subset (X,Y) \triangle (U,Z)$. We show that the following relations hold: (1) $x \in Z$ and (2) $x \in U$ iff $x \in X$. Towards a contradiction, first suppose that $x \notin Z$. Then, we have $x \in Y \triangle Z$, since $x \in Y$.

Now, since $x \notin var(P)$, also $(U, Z \cup \{x\}) \in SE(P)$ and $(U \cup \{x\}, Z \cup \{x\}) \in SE(P)$. We have $x \notin Y \triangle (Z \cup \{x\})$ which yields $Y \triangle (Z \cup \{x\}) \subset Y \triangle Z$. Therefore, $(X,Y) \triangle (U, Z \cup \{x\}) \subset (X,Y) \triangle (U,Z)$, which would be a contradiction to the assumption. Hence, $x \in Z$ follows. If (2) does not hold, we get $x \in X \triangle U$. Now, in case $x \in X$ and $x \notin U$, we have $(X,Y)\triangle(U \cup \{x\}, Z) \subset (X,Y)\triangle(U,Z)$. In case $x \in U$ and $x \notin X$, we have $(X,Y)\triangle(U \setminus \{x\}, Z) \subset (X,Y)\triangle(U,Z)$. Again both cases yield a contradiction. Clearly, $(X,Y) \in SE(Q)$ implies $(X \setminus \{x\}, Y \setminus \{x\}) \in SE(Q)$, and we obtain $(X,Y)\triangle(U,Z) = (X\setminus\{x\}, Y\setminus\{x\})\triangle(U\setminus\{x\}, Z\setminus\{x\})$. $(X \setminus \{x\}, Y \setminus \{x\}) \in \sigma(SE(Q), SE(P))$ thus follows. □

Lemma 3 *For any programs P, Q, R, $P * Q \not\subseteq_s R$ iff there exist $X \subseteq Y \subseteq var(P \cup Q \cup R)$ such that $(X,Y) \in SE(P * Q)$ and $(X,Y) \notin SE(R)$.*

Proof. The if-direction is by definition.

As for the only-if direction, since $P * Q \not\subseteq_s R$, there exists a pair (X,Y) such that $(X,Y) \in SE(P * Q)$ and $(X,Y) \notin SE(R)$. Let $V = var(P \cup Q \cup R)$. We first show that $(X \cap V, Y \cap Y) \in SE(P * Q)$. By definition, $(X,Y) \in SE(Q)$. If $SE(P) = \emptyset$, then $SE(P * Q) = SE(Q)$, and since $(X,Y) \in SE(Q)$ obviously implies $(X \cap V, Y \cap Y) \in SE(Q)$, $(X \cap V, Y \cap Y) \in SE(P * Q)$ thus follows in this case. So, suppose $SE(P) \neq \emptyset$. Then, $Y \in \sigma(Mod(Q), Mod(P))$. By iteratively applying Lemma 1, we obtain that also $Y \cap V \in \sigma(Mod(Q), Mod(P))$. Analogously using Lemma 2, $(X,Y) \in \sigma(SE(Q), SE(P))$ yields $(X \cap V, Y \cap V) \in \sigma(SE(Q), SE(P))$. By Definition 3, we get $(X \cap V, Y \cap V) \in SE(P * Q)$. Finally, it is clear that $(X,Y) \notin SE(R)$ implies $(X \cap V, Y \cap V) \notin SE(R)$. □

We now proceed with the proof of Theorem 5.

We first show membership in Σ_2^P for the complementary problem. From Lemma 3, the complementary problem holds iff there exist $X, Y \subseteq var(P \cup Q \cup R)$ such that $(X,Y) \in SE(P * Q)$ and $(X,Y) \notin SE(R)$. In what follows, let $V = var(P \cup Q \cup R)$. We first state the following observation: Recall that $Y \in \sigma(Mod(Q), Mod(P))$ iff $Y \in Mod(Q)$ and there exists a $W \in Mod(P)$ such that $W \subseteq V$ and, for each $Y' \in Mod(Q)$ and $W' \in Mod(P)$, $Y' \triangle W' \not\subset Y \triangle W$. Now, if $Y \subseteq V$, then there is also a $W \subseteq V$ satisfying the above test (this is seen by the arguments used in the proof of Lemma 1). A similar observation holds for $(X,Y) \in \sigma(SE(Q), SE(P))$.

Thus, an algorithm to decide $P * Q \not\subseteq_s R$ is as follows. We guess interpretations $X, Y, W, U, Z \subseteq V$ and start with checking $(X,Y) \in SE(Q)$ and $(X,Y) \notin SE(R)$. Then, we check whether $SE(P) = \emptyset$, which can be done via a single call to an NP-oracle. If the answer is yes, we already have found an SE interpretation (X,Y), such that $(X,Y) \in SE(P * Q)$ and $(X,Y) \notin SE(R)$, and thus the complementary problem holds. If the answer is no, we next check $(U,Z) \in SE(P)$, $W \in Mod(P)$. Then, (1) given Y and W, we check whether for each $Y' \subseteq V$ and each $W' \subseteq V$ such that $Y' \in Mod(Q)$ and $W' \in Mod(P)$, $Y' \triangle W' \not\subset Y \triangle W$ holds. It is easy to see that then the same relation holds for arbitrary models Y' and W'. From that we can conclude that $Y \in \sigma(Mod(Q), Mod(P))$. Next, (2) given (X,Y) and (U,Z), we check whether for each $X' \subseteq Y' \subseteq V$ and each $U' \subseteq Z' \subseteq V$ such that $(X',Y') \in SE(Q)$, $(U',W') \in SE(P)$, $(X',Y')\triangle(U',W') \not\subset (X,Y)\triangle(U,W)$. Again, it is easy to see that in this case $(X,Y) \in \sigma(SE(Q), SE(P))$ follows. But then we obtain $(X,Y) \in SE(P * Q)$ by Definition 3, which together with $(X,Y) \notin SE(R)$ solves the complementary problem.

We recall that model checking as well as SE model checking are in P. So, most of the checks used above are in P (except the already mentioned call to an NP-oracle) and it remains to settle the complexity of the checks (1) and (2). As well they can be done by an NP-oracle. This can be seen by considering the respective complementary problems, where one guesses the sets Y', W' (resp., X', Y', U', Z') and then performs model checking or SE model checking together with some other simple tests which are all in P. Thus, the overall algorithm runs in nondeterministic polynomial time with access to an NP-oracle. This shows the Σ_2^P-membership as desired.

As for the hardness-part, we use a reduction from $(\forall, 2)$-QSAT, which is the prototypical complete problem for Π_2^P. Let $\Phi = \forall Y \exists X \varphi$ be a QBF where φ is a CNF over $X \cup Y$. In what follows, let, for each $z \in X \cup Y$, z' be a new atom, and, for each clause $c = z_1 \vee \cdots \vee z_k \vee \neg z_{k+1} \vee \cdots \vee \neg z_m$ in φ, let $\hat{c}$ be the sequence $z'_1, \ldots, z'_k, z_{k+1}, \ldots, z_m$. Moreover, let w be a further new atom and $V = X \cup Y \cup \{z' \mid z \in X \cup Y\} \cup \{w\}$. We define the following programs: $P_\Phi = \{v \leftarrow \mid v \in V\}$, $R_\Phi = \{w \leftarrow\}$, and

$$\begin{aligned} Q_\Phi \;=\; & \{y \leftarrow {\sim}y';\; y' \leftarrow {\sim}y;\; \bot \leftarrow y, y' \mid y \in Y\} \cup \\ & \{x \leftarrow {\sim}x', w;\; x' \leftarrow {\sim}x, w;\; w \leftarrow x;\; w \leftarrow x'; \\ & \quad \bot \leftarrow x, x' \mid x \in X\} \cup \\ & \{\bot \leftarrow \hat{c}, w \mid c \text{ a clause in } \varphi\}. \end{aligned}$$

The SE models over V of these programs are as follows (for a set Z of atoms, Z' stands for $\{z' \mid z \in Z\}$):

$$\begin{aligned} SE(P_\Phi) \;&=\; \{(V,V)\}; \\ SE(Q_\Phi) \;&=\; \{(S,S) \mid S = I \cup (Y \setminus I)', I \subseteq Y\} \cup \\ & \quad\; \{(S,T), (T,T) \mid S = I \cup (Y \setminus I)', \\ & \qquad T = \{w\} \cup S \cup J \cup (X \setminus J)', \\ & \qquad I \subseteq Y, J \subseteq X, I \cup J \models \varphi\}; \\ SE(R_\Phi) \;&=\; \{(W_1, W_2) \mid \{w\} \subseteq W_1 \subseteq W_2 \subseteq V\}. \end{aligned}$$

We show that Φ is true iff $P_\Phi * Q_\Phi \subseteq_s R_\Phi$ holds.

Only-if direction: Suppose $P_\Phi * Q_\Phi \subseteq_s R_\Phi$ does not hold. By Lemma 3, there exist $S \subseteq T \subseteq var(P_\Phi \cup Q_\Phi \cup R_\Phi) = V$ such that $(S,T) \in SE(P_\Phi * Q_\Phi)$ and $(S,T) \notin SE(R_\Phi)$. Inspecting the SE models of R_Φ, we obtain that $w \notin S$. From $(S,T) \in SE(P_\Phi * Q_\Phi)$, $(S,T) \in SE(Q_\Phi)$, and thus S has to be of the form $I \cup (Y \setminus I)'$ for some $I \subseteq Y$. Recall that (V,V) is the only SE model of P_Φ over V. Hence, $S = T$ holds, since otherwise $(T,T)\triangle(V,V) \subset (S,T)\triangle(V,V)$, which is in contradiction to $(S,T) \in SE(P_\Phi * Q_\Phi)$. Now we observe that for each U with $S = T \subset U \subseteq V$,

$(U,U) \notin SE(Q_\Phi)$ has to hold (otherwise $(U,U)\triangle(V,V) \subset (S,S)\triangle(V,V)$). Inspecting the SE models of $SE(Q_\Phi)$, this only holds if, for each $J \subseteq X$, $I \cup J \not\models \varphi$. But then Φ is false.

If direction: Suppose Φ is false. Then, there exists an $I \subseteq Y$ such that for all $J \subseteq X$, $I \cup J \not\models \varphi$. We know that $(S,S) = (I \cup (Y \setminus I)', I \cup (Y \setminus I)') \in SE(Q_\Phi)$ and $(V,V) \in SE(P_\Phi)$. Next, to obtain $(S,S) \in SE(P_\Phi * Q_\Phi)$, we show $S \in \sigma(Mod(Q_\Phi), Mod(P_\Phi))$. Suppose this is not the case. Since $S \subset V$ and V is the minimal model of P_Φ, there has to exist an U with $S \subset U \subseteq V$ such that $U \in Mod(Q_\Phi)$. Recall that $S = I \cup (Y \setminus I)'$ and, by assumption, for all $J \subseteq X$, $I \cup J \not\models \varphi$. By inspecting the SE models of Q_Φ, it is clear that no such $U \in Mod(Q_\Phi)$ exists. By essentially the same arguments, $(S,S) \in \sigma(SE(Q_\Phi), SE(P_\Phi))$ can be shown. Therefore, $(S,S) \in SE(P_\Phi * Q_\Phi)$ and since $w \notin S$, $P_\Phi * Q_\Phi \subseteq_s R_\Phi$ does not hold.

This shows Π_2^P-hardness for normal programs Q. The result for positive programs Q is obtained by replacing in Q_Φ rules $y \leftarrow {\sim}y',\ y' \leftarrow {\sim}y$ by $y; y' \leftarrow$, and likewise rules $x \leftarrow {\sim}x', w,\ x' \leftarrow {\sim}x, w$ by $x; x' \leftarrow w$. Due to the presence of constraints $\perp \leftarrow y, y'$ and $\perp \leftarrow x, x'$, this modification does not change the program's SE models.

References

Alchourrón, C.; Gärdenfors, P.; and Makinson, D. 1985. On the logic of theory change: Partial meet functions for contraction and revision. *Journal of Symbolic Logic* 50(2):510–530.

Alferes, J.; Leite, J.; Pereira, L.; Przymusinska, H.; and Przymusinski, T. 1998. Dynamic logic programming. In *Proc. KR '98*, 98–109. Morgan Kaufmann.

Alferes, J.; Leite, J.; Pereira, L.; Przymusinska, H.; and Przymusinski, T. 2000. Dynamic updates of non-monotonic knowledge bases. *Journal of Logic Programming* 45(1–3):43–70.

Baral, C. 2003. *Knowledge Representation, Reasoning and Declarative Problem Solving*. Cambridge University Press.

Cabalar, P., and Ferraris, P. 2007. Propositional theories are strongly equivalent to logic programs. *Theory and Practice of Logic Programming* 7(6):745–759.

Dalal, M. 1988. Investigations into theory of knowledge base revision. In *Proc. AAAI '88*, 449–479.

Delgrande, J.; Schaub, T.; and Tompits, H. 2007. A preference-based framework for updating logic programs. In *Proc. LPNMR 2007*, 71–83. Springer.

Eiter, T., and Gottlob, G. 1992. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence* 57(2-3):227–270.

Eiter, T.; Fink, M.; Sabbatini, G.; and Tompits, H. 2002. On properties of update sequences based on causal rejection. *Theory and Practice of Logic Programming* 2(6):711–767.

Eiter, T.; Fink, M.; Tompits, H.; and Woltran, S. 2004. Simplifying logic programs under uniform and strong equivalence. In *Proc. LPNMR 2004*, 87–99. Springer.

Eiter, T.; Tompits, H.; and Woltran, S. 2005. On solution correspondences in answer-set programming. In *Proc. IJCAI 2005*, 97–102. Professional Book Center.

Foo, N., and Zhang, Y. 1997. Towards generalized rule-based updates. In *Proc. IJCAI '97*, 82–88. Morgan Kaufmann.

Gärdenfors, P. 1988. *Knowledge in Flux: Modelling the Dynamics of Epistemic States*. The MIT Press.

Inoue, K., and Sakama, C. 1998. Negation as failure in the head. *Journal of Logic Programming* 35(1):39–78.

Inoue, K., and Sakama, C. 1999. Updating extended logic programs through abduction. In *Proc. LPNMR '99*, 147–161. Springer.

Katsuno, H., and Mendelzon, A. 1992. On the difference between updating a knowledge base and revising it. In *Belief Revision*, 183–203. Cambridge University Press.

Kudo, Y., and Murai, T. 2004. A method of belief base revision for extended logic programs based on state transition diagrams. In *Proc. KES 2004*, 1079–1084. Springer.

Leite, J. 2003. *Evolving Knowledge Bases: Specification and Semantics*. IOS Press.

Lifschitz, V., and Woo, T. 1992. Answer sets in general nonmonotonic reasoning (Preliminary report). In *Proc. KR '92*, 603–614. Morgan Kaufmann.

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

Marek, V. W., and Truszczyński, M. 1998. Revision programming. *Theoretical Computer Science* 190:241–277.

Nelson, D. 1949. Constructible falsity. *Journal of Symbolic Logic* 14(2):16–26.

Osorio, M., and Cuevas, V. 2007. Updates in answer set programming: An approach based on basic structural properties. *Theory and Practice of Logic Programming* 7(4):451–479.

Przymusinski, T., and Turner, H. 1997. Update by means of inference rules. *Journal of Logic Programming* 30(2):125–143.

Satoh, K. 1988. Nonmonotonic reasoning by minimal belief revision. In *Proc. FGCS '88*, 455–462. Springer.

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

Witteveen, C.; van der Hoek, W.; and de Nivelle, H. 1994. Revision of non-monotonic theories: Some postulates and an application to logic programming. In *Proc. JELIA '94*, 137–151. Springer.

Zacarías, F.; Osorio, M.; Acosta Guadarrama, J. C.; and Dix, J. 2005. Updates in answer set programming based on structural properties. In *Proc. COMMONSENSE 2005*, 213–219. TUD–FI05–04, TU Dresden.

Zhang, Y., and Foo, N. Y. 1998. Updating logic programs. In *Proc. ECAI '98*, 403–407. IOS Press.