A Model-Theoretic Approach to Belief Change in Answer Set Programming

JAMES DELGRANDE, Simon Fraser University
TORSTEN SCHAUB, Universität Potsdam
HANS TOMPITS and STEFAN WOLTRAN, Technische Universität Wien

We address the problem of belief change in (nonmonotonic) logic programming under answer set semantics. Our formal techniques are analogous to those of distance-based belief revision in propositional logic. In particular, we build upon the model theory of logic programs furnished by SE interpretations, where an SE interpretation is a model of a logic program in the same way that a classical interpretation is a model of a propositional formula. Hence we extend techniques from the area of belief revision based on distance between models to belief change in logic programs.

We first consider belief revision: for logic programs P and Q, the goal is to determine a program R that corresponds to the revision of P by Q, denoted $P * Q$. We investigate several operators, including (logic program) expansion and two revision operators 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 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 all or nearly all of the AGM postulates for revision.

We next consider approaches for merging a set of logic programs, $P_1, \ldots, P_n$. Again, our formal techniques are based on notions of relative distance between the SE models of the logic programs. Two approaches are examined. The first informally selects for each program P_i those models of P_i that vary the least from models of the other programs. The second approach informally selects those models of a program P_0 that are closest to the models of programs $P_1, \ldots, P_n$. In this case, P_0 can be thought of as a set of database integrity constraints. We examine these operators with regards to how they satisfy relevant postulate sets.

Last, we present encodings for computing the revision as well as the merging of logic programs within the same logic programming framework. This gives rise to a direct implementation of our approach in terms of off-the-shelf answer set solvers. These encodings also reflect the fact that our change operators do not increase the complexity of the base formalism.

Categories and Subject Descriptors: I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Logic programming*; *Nonmonotonic reasoning and belief revision*; I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods—*Representation languages*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*Logic and constraint programming*

General Terms: Theory

J. Delgrande was supported by a Canadian NSERC Discovery Grant; T. Schaub was supported by the German Science Foundation (DFG) under grant SCHA 550/8-2; H. Tompits was supported by the Austrian Science Fund (FWF) under project P21698; and S. Woltran was supported by the Vienna University of Technology special fund "Innovative Projekte 9006.09/008".
T. Schaub is affiliated with Simon Fraser University, Burnaby, Canada, and Griffith University, Brisbane, Australia.
Authors' addresses: J. Delgrande, Simon Fraser University, Burnaby, B.C., Canada, V5A 1S6; email: jim@cs.sfu.ca; T. Schaub, Universität Potsdam, August-Bebel-Straße 89, D-14482 Potsdam, Germany; email: torsten@cs.uni-potsdam.de; H. Tompits and S. Woltran, Technische Universität Wien, Favoritenstraße 9-11, A-1040 Vienna, Austria; email: tompits@kr.tuwien.ac.at, woltran@dbai.tuwien.ac.at.

DOI: http://dx.doi.org/10.1145/2480759.2480766

Additional Key Words and Phrases: Answer set programming, belief revision, belief merging, program encodings, strong equivalence

ACM Reference Format:
Delgrande, J., Schaub, T., Tompits, H., and Woltran, S. 2013. A model-theoretic approach to belief change in answer set programming. *ACM Trans. Comput. Logic* 14, 2, Article 14 (June 2013), 46 pages.
DOI : http://dx.doi.org/10.1145/2480759.2480766

1. INTRODUCTION

Answer set programming (ASP) [Baral 2003; Gelfond and Lifschitz 1988] 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 applications to practical problems. However, as is the case with any large program or body of knowledge, a logic program is not a static object in general, but rather it will evolve and be subject to change, whether as a result of correcting information in the program, adding to the information already present, coalescing information in several programs, or in some other fashion modifying the knowledge represented in the program.

Since knowledge is continually evolving and subject to change, there is a need to be able to modify logic programs as new information is received. In KRR, the area of *belief revision* [Alchourrón et al. 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 characterization 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] and set containment with regards to propositional letters [Satoh 1988] are well known.

In addition to belief revision (along with the dual notion of belief *contraction*), a second major class of belief change operators addresses the *merging* of knowledge bases. The problem of merging multiple, potentially conflicting bodies of information arises in various different contexts. For example, an agent may receive reports from differing sources of knowledge, or from sets of sensors that need to be reconciled. As well, an increasingly common phenomenon is that collections of data may need to be combined into a coherent whole. In these cases, the problem is that of combining knowledge sets that may be jointly inconsistent in order to obtain a consistent set of merged beliefs. Again, as in belief revision, specific operators for merging knowledge bases have been developed based on the distance between models of the underlying knowledge bases [Baral et al. 1992; Konieczny and Pino Pérez 2002; Konieczny et al. 2002; Liberatore and Schaerf 1998; Meyer 2001; Revesz 1993].

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 (as discussed in the next section). Unfortunately, given the nonmonotonic nature of answer set programs, the problem of change in logic programs appears to be intrinsically more difficult than in a monotonic setting. In this article, our goal is to reformulate belief change in logic programs in a manner

analogous to belief change in classical propositional logic, and to investigate specific belief revision and merging operators for logic programs under the answer set semantics. Central to our approach are *SE models* [Turner 2003], which are semantic structures characterising *strong equivalence* between programs [Lifschitz et al. 2001]. This particular kind of equivalence plays a major role for different problems in logic programming—in particular, in program simplification and modularization. This is due to the fact that strong equivalence gives rise to a *substitution principle* in the sense that, for strongly equivalent programs P, Q, the programs $P \cup R$ and $Q \cup R$ have the same answer sets, for *any* program R. As is well known, 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. Moreover, following Osorio and Zacarías [2004] and Osorio and Cuevas [2007], strong equivalence amounts to *knowledge equivalence* of programs. That is, strong equivalence captures the *logical content* of a program.[1]

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 nonclassical but monotonic logic, and it is this point that we exploit here.

Given this monotonic characterization (via sets of SE models) of strong equivalence, we adapt techniques for belief change in propositional logic to belief change in logic programs. Hence we define 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 next develop revision operators based on notions of distance between SE models, and, following this, merging operators.

For a revision of logic program P by program Q, written $P * Q$, the resulting program is characterized by those SE models of Q that are closest to the SE models of P. We consider two notions of "closeness" between models; in both cases a notion of distance between models is defined in terms of the symmetric difference of the atoms true in each model. In one case, two models are of minimum distance if the symmetric difference is subset-minimal; in the other case they are of minimum distance if the symmetric difference is cardinality-minimal. These approaches to revision can be seen as extending the approaches of Satoh [1988] and Dalal [1988], respectively.

In characterising the merging of logic programs, the central idea is that the SE models of the merged program are again those that are in some sense "closest" to the SE models of the programs to be merged. However, as with merging knowledge bases in classical logic, there is no single preferred notion of distance nor closeness, and consequently different approaches have been defined for combining sources of information. We introduce two merging operators for logic programs under answer set semantics. Both operators take an arbitrary (multi)set of logic programs as argument. The first operator can be regarded an instance of what Liberatore and Schaerf [1998] call *arbitration*. Basically (SE) models are selected from among the SE models of the programs

[1]For a different concept of logical content of an answer set program; see Osorio and Zepeda [2003].

to be merged; in a sense this operator is a natural extension of our belief revision operator. The second merging operator can be regarded as an instance of the one discussed by Konieczny and Pino Pérez [2002]. Here, models of a designated program (representing information analogous to database integrity constraints) are selected that are closest to (or perhaps, informally, represent the best compromise among) the models of the programs to be merged.

Notably, in our approaches there is effectively no mention of answer sets; rather definitions of expansion, revision, and merging 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 (and in view of the intuitions of SE models as pointed out above), our operators deal with the *logical content* of a logic program.

Last, we show how our belief change approaches can be implemented. We do this by providing modular encodings for these operators in terms of a fixed nonground answer-set program. These encodings serve several purposes. First, they provide a proof-of-concept realization for the approaches. The encodings also help shed light on the details of the respective approaches. As well, they provide a tool for experimenting with the approaches.

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 in Section 3 our approaches to belief expansion and revision in ASP. We then employ these techniques in the following section to address the merging of logic programs. In each case, we discuss central properties and give complexity results. Then, in Section 5, we show how we can in fact express the process of belief change in ASP itself, giving a direct way to compute our introduced belief change operators, and we provide a general complexity analysis of our belief change approach. Finally, the article is concluded with a discussion. Proofs of results are contained in an appendix. Preliminary versions of the material in Sections 3 and 4 appeared in previous work Delgrande [2008, 2009].

2. BACKGROUND AND FORMAL PRELIMINARIES

2.1. Answer Set Programming

2.1.1. Syntax and Semantics. Let $\mathcal{A}$ be an alphabet, consisting of a set of *atoms*. A (*generalized*) *logic program*[2] (GLP) over $\mathcal{A}$ is a finite set of rules of the form

$$a_1; \ldots; a_m; \sim b_1; \ldots; \sim b_n \leftarrow c_1, \ldots, c_o, \sim d_1, \ldots, \sim d_p, \tag{1}$$

where $a_i, b_j, c_k, d_l \in \mathcal{A}$, and where $m, n, o, p \geq 0$ and $m+n+o+p > 0$. Binary operators ';' and ',' express disjunctive and conjunctive connectives. A *default literal* is an atom a or its (default) negation $\sim a$. It is convenient to distinguish various types of rules. A rule r as in (1) is:

— a *fact* if $m = 1$ and $n = o = p = 0$,
— *normal* if $m = 1$ and $n = 0$,
— *positive* if $n = p = 0$,
— *disjunctive* if $n = 0$, and
— an *integrity constraint* if $m = n = 0$.

[2]Such programs were first considered by Lifschitz and Woo [1992] and called *generalized disjunctive logic programs* by Inoue and Sakama [1998].

For example, $p \leftarrow$ is a fact (some papers also express this as 'p.'); $p \leftarrow q, {\sim}r$ is normal; $p; q \leftarrow r, s$ is positive; $p; q \leftarrow r, {\sim}s$ is disjunctive; and $\leftarrow q, {\sim}r$ is an integrity constraint, sometimes also written $\bot \leftarrow q, {\sim}r$.

Accordingly, a program is called *disjunctive* (or a DLP) if it consists of disjunctive rules only. Likewise, a program is *normal* (resp., *positive*) iff all rules in it are normal (resp., positive). We furthermore define the *head* and *body* of a rule, $H(r)$ and $B(r)$, by:

$$\begin{aligned} H(r) &= \{a_1, \ldots, a_m, {\sim}b_1, \ldots, {\sim}b_n\} \qquad \text{and} \\ B(r) &= \{c_1, \ldots, c_o, {\sim}d_1, \ldots, {\sim}d_p\}. \end{aligned}$$

Moreover, given a set X of literals, we define

$$\begin{aligned} X^+ &= \{a \in \mathcal{A} \mid a \in X\}, \\ X^- &= \{a \in \mathcal{A} \mid {\sim}a \in X\}, \text{ and} \\ {\sim}X &= \{{\sim}a \mid a \in X \cap \mathcal{A}\}. \end{aligned}$$

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. The *reduct* of a program P with respect to a set of atoms Y, denoted P^Y, is the set of rules:

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

Note that the reduct consists of negation-free rules only. An *answer set* Y of a program P is a subset-minimal model of P^Y. 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 $AS(P) = \{\{a, c\}, \{a, d\}\}$.

2.1.2. SE Models. 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 $SE(P)$. Then, 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 $SE(P) \neq \emptyset$. Note that many authors in the literature define satisfiability in terms of answer sets, in that for them a program is satisfiable if it has an answer set, that is, $AS(P) \neq \emptyset$. Thus, for example, we consider $P = \{p \leftarrow {\sim}p\}$ to be satisfiable, since $SE(P) \neq \emptyset$ even though $AS(P) = \emptyset$.[3] 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 et al. 2001]. We also write $P \models_s Q$ iff $SE(P) \subseteq SE(Q)$. For simplicity, we often drop set-notation within SE interpretations and simply write, for instance, (a, ab) instead of $(\{a\}, \{a, b\})$.

A 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

[3]Given the correspondence between SE model theory and the logic HT, our notion of satisfiability is tantamount to satisfiability in HT.

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

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

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

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

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.

We next adopt concepts introduced by Eiter et al. [2005] which are instrumental for our purposes. Let us call a set S of SE interpretations *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 $Z \supseteq Y$ with $(Z,Z) \in S$.

We have the following properties.

(1) For each GLP P, $SE(P)$ is well-defined, and for each DLP P, $SE(P)$ is complete.
(2) Conversely, 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 the latter conditions can be constructed thus [Cabalar and Ferraris 2007; Eiter et al. 2005]: In case S is a well-defined 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:

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

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

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

We conclude this subsection by introducing definitions for ordering SE models that will be needed when we come to define our belief change operators. Let $\ominus$ denote the symmetric difference operator between sets, that is, $X \ominus Y = (X \setminus Y) \cup (Y \setminus X)$ for every set X, Y. We extend $\ominus$ so that it is defined for ordered pairs, as follows.

Definition 2.1. For every pair (X_1, X_2), (Y_1, Y_2),

$$(X_1, X_2) \ominus (Y_1, Y_2) = (X_1 \ominus Y_1, X_2 \ominus Y_2).$$

Similarly, we define a notion of set containment, suitable for ordered pairs, as follows.

Definition 2.2. For every pair (X_1, X_2), (Y_1, Y_2),
$(X_1, X_2) \subseteq (Y_1, Y_2)$ iff $X_2 \subseteq Y_2$, and if $X_2 = Y_2$ then $X_1 \subseteq Y_1$.
$(X_1, X_2) \subset (Y_1, Y_2)$ iff $(X_1, X_2) \subseteq (Y_1, Y_2)$ and not $(Y_1, Y_2) \subseteq (X_1, X_2)$.

As will be seen, these definitions are appropriate for SE interpretations, as they give preference to the second element of an SE interpretation.

Set cardinality is denoted as usual by $|\cdot|$. We define a cardinality-based ordering over ordered pairs of sets as follows.

Definition 2.3. For every pair (X_1, X_2), (Y_1, Y_2),
$|(X_1, X_2)| \leq |(Y_1, Y_2)|$ iff $|X_2| \leq |Y_2|$ and if $|X_2| = |Y_2|$ then $|X_1| \leq |Y_1|$.
$|(X_1, X_2)| < |(Y_1, Y_2)|$ iff $|(X_1, X_2)| \leq |(Y_1, Y_2)|$ and not $|(Y_1, Y_2)| \leq |(X_1, X_2)|$.

As with Definition 2.2, this definition gives preference to the second element of an ordered pair. It can be observed that the definition yields a total preorder over ordered pairs. In the next section we return to the suitability of this definition, once our revision operators have been presented.

2.2. Belief Change

2.2.1. Belief Revision. The best known and, indeed, seminal work in belief revision is the *AGM approach* [Alchourrón et al. 1985; Gärdenfors 1988], in which standards for belief *revision* and *contraction* functions are given. In the revision of a knowledge base K by a formula ϕ, the intent is that the resulting knowledge base contains ϕ, be consistent (unless ϕ is not), while keeping whatever information from K can be "reasonably" retained. *Belief contraction* is a dual notion, in which information is removed from a knowledge base. While belief contraction is independently motivated and defined, it is generally accepted that in classical logic a contraction function can be obtained from a revision function by the so-called *Harper identity*, and the reverse obtained via the *Levi identity*;[4] see Gärdenfors [1988] for details.

In the AGM approach it is assumed that a knowledge base receives information concerning a static[5] domain. Belief states are modelled 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. $K_\perp$ is the inconsistent belief set (i.e., $K_\perp$ is the set of all formulas).

[4]The Harper identity states that a contraction of ϕ can be obtained by revising by $\neg\phi$ and then intersecting the result with the original belief set. The Levi identity states that a revision by ϕ can be obtained by contracting the belief set by $\neg\phi$ and then expanding the belief set by ϕ.

[5] 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.

Subsequently, Katsuno and Mendelzon [1992] reformulated the AGM approach so that a knowledge base was 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 one conjunct conjoined with the other conjunct, when the result is satisfiable.

A second major branch of belief change research concerns *belief bases* [Hansson 1999], wherein an agent's beliefs are represented by an arbitrary set of formulas, and so may not be deductively closed. Consider the two sets of sentences

$$K_1 = \{p, q\}, \quad K_2 = \{p, p \supset q\}.$$

Clearly the logical content of K_1 and K_2 is the same. In the AGM approach, wherein syntactic details of a knowledge base are suppressed, revising these knowledge bases by the same formula will give the same results. In a belief base approach, where syntactic details do matter, revision by the same formula may yield different results. Hence in the above example, if one were to revise by $\neg q$, then consistency can be maintained in K_1 by dropping q, whereas it can be maintained in K_2 by dropping either p or $p \supset q$.

2.2.2. 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 related to the Hamming distance between two interpretations, that is they are based on the set of atoms on which the interpretations disagree. The first, by Satoh [1988], is based on set containment. The second, due to Dalal [1988], uses a distance measure based on the number of atoms with differing truth values in two interpretations. 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 the method of Satoh [1988], although we also consider Dalal-style revision, since it has some technical interest with respect to ASP revision.

The *Satoh revision operator*, $\psi *_s \mu$, is defined as follows. For formulas α and β, define $\ominus^{min}(\alpha, \beta)$ as

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

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

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

The *cardinality-based*, or *Dalal revision operator*, $\psi *_d \mu$, is defined as follows. For formulas α and β, define $|\ominus|^{min}(\alpha, \beta)$ as

$$min_{\leq}(\{|w \ominus w'| \mid w \in Mod(\alpha), w' \in Mod(\beta)\}).$$

Then, $Mod(\psi *_d \mu)$ is given as

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

2.2.3. Belief Merging. Early work on merging operators includes approaches by Baral et al. [1992] and Revesz [1993]. The former authors propose various theory merging operators based on the selection of maximum consistent subsets in the union of the belief bases. The latter proposes an "arbitration" operator (as we shall see) that, intuitively, selects from among the models of the belief sets being merged. Lin and Mendelzon [1999] examine *majority* merging, in which, if a plurality of knowledge bases hold ϕ to be true, then ϕ is true in the merging. Liberatore and Schaerf [1998] address arbitration in general, while Konieczny and Pino Pérez [2002] consider a general approach in which merging takes place with respect to a set of global constraints, or formulas, that must hold in the merging. We examine these latter two approaches in detail below.

Konieczny et al. [2002] describe a very general framework in which a family of merging operators is parametrized by a distance between interpretations and aggregating functions. More or less concurrently, Meyer [2001] proposed a general approach to formulating merging functions based on ordinal conditional functions [Spohn 1988]. Booth [2002] also considers the problem of an agent merging information from different sources, via what is called *social contraction*. Last, much work has been carried out in merging possibilistic knowledge bases; we mention here, for instance, the method of Benferhat et al. [2003].

We next describe the approaches of Liberatore and Schaerf [1998] and by Konieczny and Pino Pérez [2002], since we use the intuitions underlying these approaches as the basis for our merging technique. First, Liberatore and Schaerf [1998] consider merging two belief bases built on the intuition that models of the merged bases should be taken from those of each belief base closest to the other. This is called an *arbitration operator* (Konieczny and Pino Pérez [2002] call it a *commutative revision operator*). They consider a propositional language over a finite set of atoms; consequently their merging operator can be expressed as a binary operator on formulas. The following postulates characterize this operator.

Definition 2.4. $\diamond$ is an *arbitration operator* if $\diamond$ satisfies the following postulates.

($LS1$) $\alpha \diamond \beta \equiv \beta \diamond \alpha$.
($LS2$) $\alpha \wedge \beta$ implies $\alpha \diamond \beta$.
($LS3$) If $\alpha \wedge \beta$ is satisfiable then $\alpha \diamond \beta$ implies $\alpha \wedge \beta$.
($LS4$) $\alpha \diamond \beta$ is unsatisfiable iff α is unsatisfiable and β is unsatisfiable.
($LS5$) If $\alpha_1 \equiv \alpha_2$ and $\beta_1 \equiv \beta_2$ then $\alpha_1 \diamond \beta_1 \equiv \alpha_2 \diamond \beta_2$.
($LS6$) $\alpha \diamond (\beta_1 \vee \beta_2) = \begin{cases} \alpha \diamond \beta_1 & \text{or} \\ \alpha \diamond \beta_2 & \text{or} \\ (\alpha \diamond \beta_1) \vee (\alpha \diamond \beta_2). \end{cases}$
($LS7$) $(\alpha \diamond \beta)$ implies $(\alpha \vee \beta)$.
($LS8$) If α is satisfiable then $\alpha \wedge (\alpha \diamond \beta)$ is satisfiable.

The first postulate asserts that merging is commutative, while the next two assert that, for mutually consistent formulas, merging corresponds to their conjunction. ($LS5$) ensures that the operator is independent of syntax, while ($LS6$) provides a "factoring" postulate, analogous to a similar factoring result in (AGM-style) belief revision and contraction. Postulate ($LS7$) can be taken as distinguishing $\diamond$ from other such operators; it asserts that the result of merging implies the disjunction of the original

formulas. The last postulate informally constrains the result of merging so that each operator "contributes to" (i.e., is consistent with) the final result.

Next, Konieczny and Pino Pérez [2002] consider the problem of merging possibly contradictory belief bases. To this end, they consider finite multisets of the form $\Psi = \{K_1, \ldots, K_n\}$. They assume that the belief sets K_i are consistent and finitely representable, and so representable by a formula. K^n is the multiset consisting of n copies of K. Following Konieczny and Pino Pérez [2002], let $\Delta^\mu(\Psi)$ denote the result of merging the multiset Ψ of belief bases given the entailment-based integrity constraint expressed by μ. The intent is that $\Delta^\mu(\Psi)$ is the belief base closest to the belief multiset Ψ. They provide the following set of postulates (multiset union is denoted by $\cup$).

Definition 2.5. Let Ψ be a multiset of sets of formulas, and ϕ, μ formulas (all possibly subscripted or primed). Then, Δ is an *IC merging operator* if it satisfies the following postulates.

($IC0$) $\Delta^\mu(\Psi) \vdash \mu$.
($IC1$) If $\mu \not\vdash \bot$ then $\Delta^\mu(\Psi) \not\vdash \bot$.
($IC2$) If $\bigwedge \Psi \not\vdash \neg\mu$ then $\Delta^\mu(\Psi) \equiv \bigwedge \Psi \wedge \mu$.
($IC3$) If $\Psi_1 \equiv \Psi_2$ and $\mu_1 \equiv \mu_2$ then $\Delta^{\mu_1}(\Psi_1) \equiv \Delta^{\mu_2}(\Psi_2)$.
($IC4$) If $\phi \vdash \mu$ and $\phi' \vdash \mu$ then $\Delta^\mu(\phi \cup \phi') \wedge \phi \not\vdash \bot$ implies $\Delta^\mu(\phi \cup \phi') \wedge \phi' \not\vdash \bot$.
($IC5$) $\Delta^\mu(\Psi_1) \wedge \Delta^\mu(\Psi_2) \vdash \Delta^\mu(\Psi_1 \cup \Psi_2)$.
($IC6$) If $\Delta^\mu(\Psi_1) \wedge \Delta^\mu(\Psi_2) \not\vdash \bot$ then $\Delta^\mu(\Psi_1 \cup \Psi_2) \vdash \Delta^\mu(\Psi_1) \wedge \Delta^\mu(\Psi_2)$.
($IC7$) $\Delta^{\mu_1}(\Psi) \wedge \mu_2 \vdash \Delta^{\mu_1 \wedge \mu_2}(\Psi)$.
($IC8$) If $\Delta^{\mu_1}(\Psi) \wedge \mu_2 \not\vdash \bot$ then $\Delta^{\mu_1 \wedge \mu_2}(\Psi) \vdash \Delta^{\mu_1}(\Psi) \wedge \mu_2$.

($IC2$) states that, when consistent, the result of merging is simply the conjunction of the belief bases and integrity constraints. ($IC4$) asserts that when two belief bases disagree, merging does not give preference to one of them. ($IC5$) states that a model of two mergings is in the union of their merging. With ($IC5$) we get that if two mergings are consistent then their merging is implied by their conjunction. Note that merging operators are trivially commutative. ($IC7$) and ($IC8$) correspond to the extended AGM postulates ($K*7$) and ($K*8$) for revision (cf. Alchourrón et al. [1985] and Gärdenfors [1988]), but with respect to the integrity constraints.

2.3. Belief Change in Logic Programming

Most previous work on belief change for logic programs goes under the title of *update* [Alferes et al. 2000; Delgrande et al. 2007; Eiter et al. 2002; Leite 2003; Przymusinski and Turner 1997; Sakama and Inoue 2003; Zacarías et al. 2005; Zhang and Foo 1997, 1998]. Strictly speaking, however, such approaches generally do not address "update," at least insofar as the term is understood in the belief revision community. In this community, update refers to a belief change in response to a change in the world being modelled [Katsuno and Mendelzon 1992]; hence update is concerned with a setting in which the world has evolved to a new state. This notion of change is not taken into account in the previously cited work; instead, it is possible that change is with respect to a static world, in which a logic program is modified to better represent this domain.[6]

Following the investigations of the Lisbon group of researchers [Alferes et al. 2000; Leite 2003], a typical setting for update approaches is to consider a sequence $P_1, P_2, \ldots, P_n$ of programs where each P_i is a logic program (this is done, e.g., in the approaches of

[6]To be clear, our interests in this article lie with *revision*, where a logic program is revised with respect to some (static) underlying domain or world.

Eiter et al. [2002], Zacarías et al. [2005], and Delgrande et al. [2007]). For P_i, P_j, and $i > j$, 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 single logic program that contains an encoding of the priorities, or by treating the sequence as a prioritized logic program, or by some other appropriate method. Most such approaches are founded on the notion of *causal rejection*. According to this principle, a rule r is rejected if there is another, higher-ranked rule r' which conflicts with r. That is, if both r and r' are applicable and have conflicting heads then only r' is applied. The net result, one way or another, is that one obtains a set of answer sets from such a program sequence. That is, one does not obtain a single, new program expressed in the language of the original logic programs. Hence, these approaches fall outside the general AGM belief revision paradigm. As well, it should be clear that such approaches are syntactic in nature, and so such approaches fall into the *belief base* category, rather than the *belief set* category.

For illustration, we briefly consider one such approach, that of Eiter et al. [2002]. In this approach the semantics of an $(n-1)$-fold update $P_1 \circ \cdots \circ P_n$ is given by the semantics of an (ordinary) program $P_{\lhd}$, containing the following elements:

(1) all integrity constraints in P_i, $1 \leq i \leq n$;
(2) for each $r \in P_i$, $1 \leq i \leq n$:
$$l_i \leftarrow B(r), {\sim}rej(r), \quad \text{where } H(r) = l;$$
(3) for each $r \in P_i$, $1 \leq i < n$:
$$rej(r) \leftarrow B(r), \neg l_{i+1}, \quad \text{where } H(r) = l;$$
(4) for each literal l occurring in P $(1 \leq i < n)$:
$$l_i \leftarrow l_{i+1}; \qquad l \leftarrow l_1.$$

Here, for each rule r, $rej(r)$ is a new atom not occurring in $P_1, \ldots, P_n$. Intuitively, $rej(r)$ expresses that r is "rejected." Similarly, each l_i, $1 \leq i \leq n$, is a new atom not occurring in $P_1, \ldots, P_n$. Answer sets of $P_1 \circ \cdots \circ P_n$ are given by the answer sets of $P_{\lhd}$, intersected with the original language.

Consider the following example adapted from Alferes et al. [2000]. We have the update of P_1 by P_2, where

$$\begin{aligned} P_1 &= \{\, r_1 : sleep \leftarrow {\sim}tv\_on, \quad r_2 : night \leftarrow, \\ &\qquad r_3 : tv\_on \leftarrow, \quad r_4 : watch\_tv \leftarrow tv\_on \,\}, \\ P_2 &= \{\, r_5 : \neg tv\_on \leftarrow power\_failure,\ r_6 : power\_failure \leftarrow \,\}. \end{aligned}$$

The single answer set of $P_1 \circ P_2$ is

$$S = \{power\_failure, \neg tv\_on, sleep, night\}.$$

If new information arrives as program P_3, given by

$$P_3 = \{\, r_7 : \neg power\_failure \leftarrow \,\},$$

then $P_1 \circ P_2 \circ P_3$ has the unique answer set

$$T = \{\, \neg power\_failure, tv\_on, watch\_tv, night \,\}.$$

Again, it can be noted that this approach deals with the rules in a program; hence this approach, as with related approaches, falls into the category of *base* revision.

However, various principles have nonetheless 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 reinterpreting

different AGM-style postulates for revising or updating classic knowledge bases, as well as introducing new principles. Among the latter, we note the following:

Initialization $\emptyset * P \equiv P$.
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 reinterpreting the standard AGM postulates in the context of logic programs, the logic underlying strong equivalence should be adopted. Since Osorio and Cuevas [2007] studied programs with strong negation,[7] this led them to consider the logic $\mathbf{N}_2$, an extension of HT by allowing strong negation.[8] They rephrased the AGM postulates in terms of the logic $\mathbf{N}_2$ and also introduced a new principle, referred to as *weak independence of syntax* (WIS), which they proposed that any update operator should satisfy:

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

Indeed, following this spirit, the preceding absorption and augmentation principles can be accordingly changed by replacing their antecedents by "$Q \equiv_s R$" and "$Q \models_s R$", respectively. Osorio and Cuevas [2007] defined a variation of the semantics by Eiter et al. [2002] and showed that it satisfies all their adapted AGM postulates. Furthermore, they show that a further variant of their semantics is equivalent to the semantics by Eiter et al. [2002] for a certain class of programs which satisfies all except one of the adapted AGM postulates. We note that the WIS principle was also discussed in an update approach based on *abductive programs* [Zacarías et al. 2005].

In contrast to the works discussed above, Sakama and Inoue [2003] do not deal with sequences of programs but with characterising different kinds of knowledge-base updates in terms of extended abduction [Inoue and Sakama 1995]. In particular, they discuss, besides usual theory update, view update and consistency restoration. For view update and consistency restoration, it is assumed that programs are divided into a variable and an invariable part. In view update, the task is to change the variable part given an update request in the form of a fact, while in consistency restoration, the problem is to modify the variable part of a program P whose constraint-free part violates the constraints of P (and it is assumed that constraints are themselves supposed to be invariant). Let us have a closer look at their method of theory update. Given programs P_1 and P_2, an update of P_1 by P_2 is a largest program Q such that $P_1 \subseteq Q \subseteq P_1 \cup P_2$ and Q has a consistent answer set. This problem is then reduced to the problem of computing a minimal set R of abducible rules such that $R \subseteq P_1 \setminus P_2$ and $(P_1 \cup P_2) \setminus R$ has an answer set. The intended update is realized via a minimal *anti-explanation* for falsity, which removes abducible rules in order to restore consistency. As demonstrated by Eiter et al. [2002], this approach violates causal rejection. Furthermore, it is different from our approach to revision. For instance, updating $P_1 = \{p \leftarrow;\ q \leftarrow\}$ by $P_2 = \{\bot \leftarrow p, q\}$ in Sakama and Inoue's approach results in the removal of one of the two rules in P_1, while our approach yields a program having $\{p\}$ and $\{q\}$ as answer sets.

[7]Strong negation is sometimes also referred to as *classical negation*, but this is a misnomer as it does not enjoy all properties of classical negation.

[8]$\mathbf{N}_2$ itself traces back to an extension of intuitionistic logic with strong negation, first studied by Nelson [1949].

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 et al. [1994]). Other work uses logic programs under a variant of the stable semantics to specify database revision, that is, 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.

In belief change, the operation of belief *contraction* is a dual to revision. The idea is that in contracting by a formula ϕ, the agent ceases to believe that ϕ (if it ever did believe ϕ), while not necessarily believing $\neg\phi$. Then a revision by ψ can be implemented by first contracting $\neg\psi$ and then adding ψ. There has been, insofar as we are aware, no work on belief contraction with respect to logic programs. There has been some work on the related notion of *forget* [Eiter and Wang 2008; Zhang and Foo 2006] in which forgetting a literal is akin to shrinking the language by the corresponding atom. It can be noted first of all that the cited works are syntactic, in that they are couched with reference to underlying answer sets. (In fact, Eiter and Wang [2008] discuss the relation between their notion of forgetting and logic program update.) However, more pertinently, forgetting appears to be too drastic to yield an acceptable approach to revision. Consider for example where a knowledge base believes that a penguin flies, $F \leftarrow P$, and one wants to revise by the fact that a penguin does not fly, say $\bot \leftarrow P, F$. We can remove the conflict by forgetting P or F (or both). However, if one forgets P, then all information about penguins (say that they eat fish or walk upright) is lost; similarly if one forgets F, then all information about flight (say that bats fly) is lost. So forgetting P or F will allow $\bot \leftarrow P, F$ to be consistently added, but at much too great a cost.

With respect to merging logic programs, we have already mentioned updating logic programs, which can also be considered as *prioritized logic program merging*. With respect to merging unprioritized logic programs, Baral et al. [1991] describes an algorithm for combining a set of normal, stratified logic programs in which the union of the programs is also stratified. In their approach the combination is carried out so that a set of global integrity constraints, which is satisfied by individual programs, is also satisfied by the combination. As well, in this approach (as with related approaches to logic program merging), logic program combining is obtained via manipulating and transforming program rules. Hence the result is dependent on program syntax, and so syntax independence as exemplified by (IC3) in Definition 2.5 is not obtained in these approaches. In contrast, in the approaches described in Section 4, syntax independence is obtained.

Buccafurri and Gottlob [2002] present an interesting approach whereby rules in a given program encode desires for a corresponding agent. A predicate *okay* indicates that an atom is acceptable to an agent. Answer sets of these *compromise logic programs* represent acceptable compromises between agents. While it is shown that the joint fixpoints of such logic programs can be computed as answer sets and complexity results are presented, the approach is not analysed from the standpoint of properties of merging.

In a succession of works, Sakama and Inoue [2006, 2007, 2008] address what they respectively call *coordination*, *composition*, and *consensus* between logic programs. In short, given programs P_1 and P_2, a program Q is a *generous coordination* of P_1 and P_2 if $AS(Q) = AS(P_1) \cup AS(P_2)$ while it is a *rigorous coordination* of P_1 and P_2 if $AS(Q) = AS(P_1) \cap AS(P_2)$. On the other hand, the *composition* of P_1 and P_2 is defined

as a program whose answer sets are given by $\min\{S \uplus T \mid S \in AS(P_1), T \in AS(P_2)\}$, where $S \uplus T$ is $S \cup T$ if $S \cup T$ is consistent, otherwise $S \uplus T$ is the set of all literals.[9] Finally, Q is a *minimal consensus* between P_1 and P_2 if its answer sets are given by $\min\{S \cap T \mid S \in AS(P_1), T \in AS(P_2)\}$ and it is a *maximal consensus* between P_1 and P_2 if its answer sets are given by $\max\{S \cap T \mid S \in AS(P_1), T \in AS(P_2)\}$. Intuitively, the coordination of two programs results in a program collecting either all answer sets (in case of generous coordination) or picks just the common answer sets (in case of rigorous coordination). While this construction leaves the original answer sets unchanged, the composition of programs *combines* the answer sets of the given programs. Finally, the consensus of two programs is based on the intersection of the answer sets of P_1 and P_2 and reflects, in a sense, the meaning of the original programs. As the authors of these approaches maintain, coordination, composition, and consensus formalize different types of social behaviors of logical agents and their goals differ from merging and revision.

3. BELIEF CHANGE IN ASP BASED ON SE MODELS

In AGM belief change, an agent's beliefs can be abstractly characterized in various different ways. In the classical AGM approach an agent's beliefs are given by a *belief set*, that is, a deductively-closed set of sentences. As well, an agent's beliefs may also be characterized 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, where equivalent formulas express the same knowledge. Given a finite language, it is straightforward to translate between these representations.

In ASP, there are notions analogous to the above for abstractly characterising an agent's beliefs. Thus, given a logic program P, the belief set corresponding to this program could be taken as the maximal program $Th(P)$ that is strongly equivalent to P, that is, $Th(P) = \bigcup\{Q \mid Q \equiv_s P\}$. Thus we would have $P \equiv_s Q$ iff $Th(P) = Th(Q)$, and so $Th(P)$ would be an abstract representation of a logic program that may have a specific syntactic representation given by P.

Similarly, the set of SE models of a program may be taken as an abstract characterization of that program. Thus, in a sense, the set of SE models of a program can be considered as the *proposition* expressed by the program, and so the set of SE models arguably provides an appropriate abstract representation of the logical content of a particular logic program. As we show below, this level of abstraction allows the definition of specific belief change operators, analogous to operators in belief change in propositional logic and possessing good formal properties. Hence at this level, we are able to study belief change, independently of how knowledge is represented in a logic program and instead, again, focus on the logical content of the program.

3.1. Logic Program Expansion

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.

[9]Note that Sakama and Inoue [2006] use strong negation.

Definition 3.1. For logic programs P and Q, define the *expansion* of P and Q, $P+Q$, to be a logic program R such that $SE(R) = SE(P) \cap SE(Q)$.

For illustration, consider the following examples:[10]

(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 are straightforward consequences of the definition of expansion with respect to SE models.

THEOREM 3.2. *Let P and Q be logic programs. Then:*

(1) $P + Q$ is always defined.
(2) $P + Q \models_s P$.
(3) If $P \models_s Q$, then $P + Q \equiv_s P$.
(4) If $P \models_s Q$, then $P + R \models_s Q + R$.
(5) $SE(P + Q)$ is well-defined.
(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, much earlier work in updating logic programs had trouble with the last property, expressing a *tautology* postulate (though this has been addressed in more recent work such as that of Alferes et al. [2005]). In the current approach, 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—*initialization*, *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 $AS(\{p \leftarrow \sim q,\ q \leftarrow \sim p\}) = \{\{p\}, \{q\}\}$. Last, in Example (5), it can be seen that expanding by a program with no answer sets may nonetheless result in a program that has answer sets.

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

The overall result is that expansion with respect to logic programs is a meaningful and arguably interesting operator. The expansion of two programs provides a meaningful result whenever the two programs have an SE model in common; if they don't have an SE model in common then expansion results in an unsatisfiable program. Logic program revision, as examined next, generalizes expansion, in that it produces a meaningful result whenever the programs involved are each separately satisfiable.

3.2. Logic Program Revision

We next turn to specific operators 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, we begin by considering set-containment based revision. Thus, $P * Q$ will be 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. Following the development of this operator we also consider cardinality-based revision, as a point of contrast. While these two approaches correspond to the two best-known ways of incorporating distance based revision, they are not exhaustive and any other reasonable notion of distance could also be employed.

3.2.1. Set-Containment Based Revision. The following definition gives, for sets E_1 and 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 3.3. Let E_1, E_2 be two sets of either classical or SE interpretations. Then:

$$\sigma(E_1, E_2) = \{A \in E_1 \mid \exists B \in E_2 \text{ such that } \forall A' \in E_1, \forall B' \in E_2, A' \ominus B' \not\subset A \ominus B\}.$$

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.[11] Unfortunately, Definition 3.3 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 $(Y, Y) \notin \sigma(SE(Q), SE(P))$. Hence, in defining $P * Q$ in terms of $\sigma(SE(Q), SE(P))$, we must elaborate the set $\sigma(SE(Q), SE(P))$ in some fashion to obtain a well-defined set of SE models. There are two ways in which this might be done:

(1) Determine a subset of $\sigma(SE(Q), SE(P))$ so that only well-defined sets of SE models are obtained, or
(2) determine a superset of $\sigma(SE(Q), SE(P))$ so that a well-defined set is obtained.

It proves to be the case that the second alternative produces overly weak results. In view of this, we adopt the first approach; however following the development of this approach, we briefly consider the alternative.

[11] Recall that generalized logic programs are characterized by well-defined sets of SE models. If we began with a disjunctive logic program, then our revision operator must also produce a *complete* set of SE models. Since our technique for obtaining a well-defined set of models is readily extendable to one that yields a complete set, in the interests of space, we omit the case for DLPs.

Our approach is based on the following idea to obtain a well-defined set of models of $P * Q$ based on the 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. It can also be observed that this approach parallels that of the definition of an SE model. That is, the SE models of a program P are those SE interpretations (X, Y) where Y is a classical model of P and X is a model of the reduct P^Y. Similarly, in determining models of a revision $P * Q$, we select those SE models of Q in which, for (X, Y), Y is a closest (classical) model to P, and then X is a closest model of the reducts.

We have the following definition for revision.

Definition 3.4. For logic programs P and Q, define the *revision* of P by Q, $P * Q$, to be a logic program such that:

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

otherwise

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

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 that of containment-based revision in propositional logic [Satoh 1988]. 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 nonempty expansions are also valid program revisions. We have the following examples of revision that do not reduce to expansion.[12]

(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.

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

(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)

$$\left\{ \begin{array}{l} \bot \leftarrow p \\ \bot \leftarrow q \end{array} \right\} * \{p; q \leftarrow\} \equiv_s \left\{ \begin{array}{l} p; q \leftarrow \\ \quad \bot \leftarrow p, q \end{array} \right\}.$$

Comparing these examples with the update approaches for logic programs as put forth by Alferes et al. [2000], Eiter et al. [2002] and Sakama and Inoue [2003], by updating, for instance, $\{p \leftarrow; q \leftarrow\}$ by $\{\bot \leftarrow p, q\}$ (corresponding to Example (3)), in the approaches of Alferes et al. [2000] and Eiter et al. [2002] we get no answer set (simply because there are no conflicting heads), while in the approach by Sakama and Inoue [2003], one has to remove one of the two facts in $\{p \leftarrow; q \leftarrow\}$ (as already pointed out previously).

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

($RA1$) $P * Q \models_s Q$.
($RA2$) 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 $P_1 * Q_1 \equiv_s P_2 * Q_2$.
($RA5$) $(P * Q) + R \models_s P * (Q + R)$.
($RA6$) If $(P * Q) + R$ is satisfiable, then $P * (Q + R) \models_s (P * Q) + R$.

We obtain that logic program revision as given in Definition 3.4 satisfies the first five of the revision postulates. Unsurprisingly, this is analogous to set-containment based revision in propositional logic.

THEOREM 3.5. *The logic program revision operator $*$ from Definition 3.4 satisfies postulates* ($RA1$)–($RA5$).

The fact that our revision operator does not satisfy ($RA6$) can be seen by the following example:

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

Straightforward computations show that

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

So, $P * (Q + R) \not\models_s (P * Q) + R$. 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 3.6. *Let P and Q be logic programs. Then, $P * Q$ satisfies initialization, idempotency, and absorption with respect to strong equivalence. If P is satisfiable, then $P * Q$ satisfies tautology.*

It can be noted that if program P is unsatisfiable but Q is tautologous, then for $P * Q$, the principle *tautology* conflicts with (*RA*3). For our definition of revision in Definition 3.4, we elected to satisfy (*RA*3) in this case, in order to adhere with the AGM approach; we could as easily have decided to satisfy tautology and not (*RA*3).

It can also be noted that augmentation does not hold; nor in fact would one expect it to hold in a distance-based approach. For example, consider the case where P, Q, and R are characterized by models

$$\begin{aligned} SE(P) &= \{(a,a),(ab,ab)\},\\ SE(Q) &= \{(ab,ab),(ac,ac),(b,b)\},\\ SE(R) &= \{(ac,ac),(b,b)\}. \end{aligned}$$

Thus $SE(R) \subseteq SE(Q)$. We obtain that $SE(P * Q) = SE(P + Q) = \{(ab,ab)\}$, and thus $SE((P * Q) * R) = \{(b,b)\}$. However, $SE(P * R) = \{(ac,ac),(b,b)\}$, contradicting augmentation.

Definition 3.4 seems to be the most natural approach for constructing a set-containment based revision operator. However, it is not the only such possibility. We next briefly discuss an alternative definition for revision. The idea here is that for the revision of P by Q, we select the closest models of Q to P, and then add interpretations to make the result well-defined.

Definition 3.7. For logic programs P and Q, define the *weak revision* of P by Q to be a logic program $P *_w Q$ such that:

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

otherwise

$$\begin{aligned} SE(P *_w Q) = \sigma(SE(Q), SE(P)) \cup \\ \{(Y,Y) \mid (X,Y) \in \sigma(SE(Q), SE(P)) \text{ for some } X\}. \end{aligned}$$

The drawback to this approach is that it introduces possibly irrelevant interpretations in order to obtain well-definedness. As well, Definition 3.4 appears to be the more natural. Consider the following example, which also serves to distinguish Definition 3.4 from Definition 3.7. Let

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

Then, we have the following SE models:

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

and

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

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 3.8. *Let P and Q be programs. Then, $P * Q \models_s P *_w Q$.*

For completeness, we mention the fact that it is easy to enforce well-definedness by simply considering only models of the form (Y, Y) in the revision of P by Q. However, this alternative is problematic for two reasons. First, information is lost in ignoring SE models of form (X, Y) where $X \subset Y$. Second, for our motivating example, we would obtain $SE(\{p \leftarrow \sim p\} * \{\bot \leftarrow p\}) = \emptyset$, violating the key postulate (*RA*3), that the result of revising by a satisfiable program results in a satisfiable revision.

3.2.2. Cardinality-Based Revision. We next briefly recapitulate the previous development but in terms of cardinality-based revision. Define, for two sets of interpretations, E_1, E_2, the subset of E_1 that is closest to E_2, where the notion of "closest" is now given in terms of cardinality.

Definition 3.9. Let E_1, E_2 be two sets of either classical or SE interpretations. Then:

$$\sigma_{||}(E_1, E_2) = \{A \in E_1 \mid \exists B \in E_2 \text{ such that } \forall A' \in E_1, \forall B' \in E_2, |A' \ominus B'| \not< |A \ominus B|\}.$$

As with set containment-based revision, we must ensure that our operator results in a well-defined set of SE models. Again, we first give preference to potential answer sets, in the form of models (Y, Y), and then to general models.

Definition 3.10. For logic programs P and Q, define the *(cardinality-based) revision* of P by Q, $P *_c Q$, to be a logic program such that:

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

otherwise

$$SE(P *_c Q) = \{(X, Y) \mid Y \in \sigma_{||}(Mod(Q), Mod(P)), X \subseteq Y, \text{ and if } X \subset Y \text{ then } (X, Y) \in \sigma_{||}(SE(Q), SE(P))\}.$$

$P *_c Q$ can be seen to be well-defined, and so can be represented through a canonical logic program. As well, over classical, propositional models the definition reduces to cardinality-based revision in propositional logic [Dalal 1988].

We observe from the respective definitions that

$$SE(P *_c Q) \subseteq SE(P * Q).$$

That the two revision operators differ is easily shown: For example, if

$$P = \left\{ \begin{array}{l} p \leftarrow \\ q \leftarrow \\ r \leftarrow \end{array} \right\} \text{ and } Q = \left\{ \begin{array}{ll} p ; q & \leftarrow \\ r & \leftarrow q \\ & \leftarrow p, r \end{array} \right\}$$

we get $SE(P) = \{(pqr, pqr)\}$ and $SE(Q) = \{(p, p), (qr, qr)\}$. This yields $SE(P * Q) = \{(p, p), (qr, qr)\}$ while $SE(P *_c Q) = \{(qr, qr)\}$.

It can be observed that $P *_c Q$ yields the same results as $P * Q$ for the five examples given in the previous section. However, cardinality-based revision fully aligns with the AGM postulates.

THEOREM 3.11. *Let P and Q be logic programs. Then, $P *_c Q$ satisfies postulates* ($RA1$)–($RA6$).

As well, the following result is straightforward.

THEOREM 3.12. *Let P and Q be logic programs. Then, $P *_c Q$ satisfies initialization, idempotency, and absorption with respect to strong equivalence. If P is satisfiable then $P *_c Q$ also satisfies tautology.*

Finally, we remark that another plausible definition of an ordering underlying cardinality-based revision would be the following:

$$|(X_1, X_2)| \leq' |(Y_1, Y_2)| \text{ iff } |X_1| \leq |Y_1| \text{ and } |X_2| \leq |Y_2|.$$

However, this ordering yields a partial preorder, and a revision operator based on this notion of distance would be very similar to $P * Q$; in particular the postulate ($RA6$) would not be satisfied. Since this operator is of at best marginal interest, we do not explore it further.

3.2.3. Remarks. Both of our proposed approaches to revising logic programs are based on a notion of distance between SE models. In the first, a partial preorder was induced between SE models, while in the second a total preorder resulted. We note that any definition of distance that results in a partial (resp., total) preorder among SE models could have been used, with the same technical results obtaining (but not, of course, the same examples). Hence, these approaches are exemplars of the two most common types of revision, expressed in terms of differences among truth values of atoms in models. As such, our specific approaches can be seen as natural generalizations of the approaches of Satoh [1988] and Dalal [1988].

We have suggested earlier that the approach based on set containment is the more natural or plausible approach, even though it does not satisfy all of the AGM postulates. This is because the cardinality-based approach may make somewhat arbitrary distinctions in arriving at a total preorder over SE interpretations. Recall the example we used to illustrate the difference between the approaches:

$$SE(P) = \{(pqr, pqr)\} \text{ and } SE(Q) = \{(p, p), (qr, qr)\},$$

yielding

$$SE(P * Q) = \{(p, p), (qr, qr)\} \text{ and } SE(P *_c Q) = \{(qr, qr)\}.$$

Given that we have no information concerning the ontological import of the atoms involved, it seems somewhat arbitrary to decide (in the case of $*_c$) that qr should take priority over p. As an alternative argument, consider where for some large n we have

$$SE(P) = \{(p_1 \ldots p_{2n}, p_1 \ldots p_{2n})\} \quad \text{and}$$
$$SE(Q) = \{(p_1 \ldots p_{n+1}, p_1 \ldots p_{n+1}), (p_1 \ldots p_n, p_1 \ldots p_n)\}.$$

So, in this example it is quite arbitrary to select (as the cardinality-based approach does) $(p_1 \ldots p_{n+1}, p_1 \ldots p_{n+1})$ over $(p_1 \ldots p_n, p_1 \ldots p_n)$.

Finally, some comment should be made as to how one obtains a program after an expansion or revision. That is, expansion and revision are defined in terms of SE models, and then a program is given whose SE models corresponds to the result of the belief change operator. There are two means by which a resulting program may be obtained. First, one may have a good idea as to what the resulting program should look like. For example the third illustrative case in revision was $\{p \leftarrow,\ q \leftarrow\} * \{\leftarrow p, q\}$. Hence one believes that both p and q are true, and then learns that one of them must be false. Consequently, the revised knowledge base consists of the fact that precisely one is true and the other false, $\{p; q \leftarrow,\ \leftarrow p, q\}$; and this can be verified to be the case

by examining the SE models of the change operation. This of course will work only in simple cases. In general, however, one can construct the canonical DLP, as described in Section 2. This is a brute-force approach, and while it does yield a program, it does not yield a *perspicuous* program in general. Overall then, as with classical belief revision, obtaining a clear and understandable program is a difficult problem.

4. LOGIC PROGRAM MERGING

We denote (generalized) logic programs by $P_1, P_2, \ldots$, reserving P_0 for a program representing global constraints, as described later. For logic programs P_1 and P_2, we define $P_1 \sqcap P_2$ to be a program with SE models equal to $SE(P_1) \cap SE(P_2)$ and $P_1 \sqcup P_2$ to be a program with SE models equal to $SE(P_1) \cup SE(P_2)$. By a *belief profile*, Ψ, we understand a sequence[13] $\langle P_1, \ldots, P_n \rangle$ of (generalized) logic programs. For $\Psi = \langle P_1, \ldots, P_n \rangle$ we write $\sqcap\Psi$ for $P_1 \sqcap \cdots \sqcap P_n$. We write $\Psi_1 \circ \Psi_2$ for the (sequence) concatenation of belief profiles Ψ_1 and Ψ_2; and for logic program P_0 and $\Psi = \langle P_1, \ldots, P_n \rangle$ we abuse notation by writing $\langle P_0, \Psi \rangle$ for $\langle P_0, P_1, \ldots, P_n \rangle$. A belief profile Ψ is *satisfiable* just if each component logic program is satisfiable. The set of SE models of Ψ is given by $SE(\Psi) = SE(P_1) \times \cdots \times SE(P_n)$. For $\overline{S} \in SE(\Psi)$ such that $\overline{S} = \langle S_1, \ldots, S_n \rangle$, we use S_i to denote the ith component of $\overline{S}$. Thus, $S_i \in SE(P_i)$. Analogously, the set of classical propositional models of Ψ is given by $Mod(\Psi) = Mod(P_1) \times \cdots \times Mod(P_n)$; also we use X_i to denote the ith component of $\overline{X} \in Mod(\Psi)$.

4.1. Arbitration Merging

For the first approach to merging, called *arbitration*, we consider models of Ψ and select those models in which, in a global sense, the constituent models vary minimally. The result of arbitration is a logic program made up of SE models from each of these minimally-varying tuples. Note that, in particular, if a set of programs is jointly consistent, then there are models of Ψ in which all constituent SE models are the same. That is, the models that vary minimally are those $\overline{S} \in SE(\Psi)$ in which $S_i = S_j$ for every $1 \leq i,j \leq n$; and merging is the same as simply taking the union of the programs.

The first definition provides a notion of distance between models of Ψ, while the second then defines merging in terms of this distance.

Definition 4.1. Let $\Psi = \langle P_1, \ldots, P_n \rangle$ be a satisfiable belief profile and let $\overline{S}$ and $\overline{T}$ be two SE models of Ψ (or two classical models of Ψ).

Then, define $\overline{S} \leq_a \overline{T}$, if $S_i \ominus S_j \subseteq T_i \ominus T_j$ for every $1 \leq i < j \leq n$.

Clearly, $\leq_a$ is a partial preorder. In what follows, let $Min_a(N)$ denote the set of all minimal elements of a set N of tuples relative to $\leq_a$, that is,

$$Min_a(N) = \{\overline{S} \in N \mid \overline{T} \leq_a \overline{S} \text{ implies } \overline{S} \leq_a \overline{T} \text{ for all } \overline{T} \in N\} .$$

Preparatory for our central definition to arbitration merging, we furthermore define, for a set N of n-tuples,

$$\cup N = \{S_i \mid \exists \overline{S} \in N \text{ such that } \overline{S} = \langle S_1, \ldots, S_n \rangle \text{ and } i \in \{1, \ldots, n\}\}.$$

Definition 4.2. Let $\Psi = \langle P_1, \ldots, P_n \rangle$ be a belief profile. Then, the *arbitration merging*, or simply *arbitration*, of Ψ, is a logic program $\nabla(\Psi)$ such that

$$SE(\nabla(\Psi)) = \{(X,Y) \mid Y \in \cup Min_a(Mod(\Psi)), X \subseteq Y, \\ \text{and if } X \subset Y \text{ then } (X,Y) \in \cup Min_a(SE(\Psi))\} ,$$

[13]This departs from usual practice, where a belief profile is usually taken to be a multiset.

Table I. Examples of Arbitration Merging

P_1	P_2	$SE(\nabla(\langle P_1,P_2\rangle))$	$\nabla(\langle P_1,P_2\rangle)$
$\{p\leftarrow\}$	$\{q\leftarrow\}$	$\{(pq,pq)\}$	$\{p\leftarrow, q\leftarrow\}$
$\{p\leftarrow\}$	$\{\leftarrow p\}$	$\{(p,p),(\emptyset,\emptyset)\}$	$\{p;\sim p\leftarrow\}$
$\{p\leftarrow\sim p\}$	$\{\leftarrow p\}$	$\{(\emptyset,p),(p,p),(\emptyset,\emptyset)\}$	$\{\}$
$\{p\leftarrow, q\leftarrow\}$	$\{\leftarrow p,q\}$	$\{(pq,pq),(p,p),(q,q)\}$	$\{p;q\leftarrow, p;\sim p\leftarrow, q;\sim q\leftarrow\}$
$\{\bot\leftarrow\sim p, \bot\leftarrow\sim q\}$	$\{\leftarrow p,q\}$	$\{S\in SE(\emptyset)\mid S\neq(\emptyset,\emptyset)\}$	$\{\bot\leftarrow\sim p,\sim q\}$
$\{\bot\leftarrow p, \bot\leftarrow q\}$	$\{p;q\leftarrow\}$	$\{(\emptyset,\emptyset),(p,p),(q,q)\}$	$\{\leftarrow p,q,\ p;\sim p\leftarrow,\ q;\sim q\leftarrow\}$

providing Ψ is satisfiable, otherwise, if P_i is unsatisfiable for some $1\leq i\leq n$, define $\nabla(\Psi)=\nabla(\langle P_1,\ldots,P_{i-1},P_{i+1},\ldots,P_n\rangle)$.

For illustration, consider the belief profile

$$\langle P_1,P_2\rangle=\langle\{p\leftarrow, u\leftarrow\},\ \{\leftarrow p, v\leftarrow\}\rangle\ . \tag{2}$$

Since $SE(P_1)=\{(pu,pu),(pu,puv),(puv,puv)\}$ and $SE(P_2)=\{(v,v),(v,uv),\ (uv,uv)\}$, we obtain nine SE models for $SE(\langle P_1,P_2\rangle)$. Among them, we find a unique $\leq_a$-minimal one, yielding $Min_a(SE(\langle P_1,P_2\rangle))=\{\langle(puv,puv),(uv,uv)\rangle\}$. Similarly, $\langle P_1,P_2\rangle$ has a single $\leq_a$-minimal collection of pairs of classical models, viz. $Min_a(Mod(\langle P_1,P_2\rangle))=\{\langle puv,uv\rangle\}$. Accordingly, we get

$$\begin{aligned}\cup Min_a(Mod(\langle P_1,P_2\rangle)) &= \{puv,uv\},\\ \cup Min_a(SE(\langle P_1,P_2\rangle)) &= \{(puv,puv),(uv,uv)\},\ \text{and}\\ SE(\nabla((P_1,P_2))) &= \cup Min_a(SE(\langle P_1,P_2\rangle))\ .\end{aligned}$$

We thus obtain the program $\nabla(\langle P_1,P_2\rangle)=\{p;\sim p\leftarrow, u\leftarrow, v\leftarrow\}$ as the resultant arbitration of P_1 and P_2.

For further illustration, consider the technical examples given in Table I.

We note that merging normal programs often leads to disjunctive or generalized programs. Although plausible, this is also unavoidable because merging does not preserve the model intersection property of the reduced program satisfied by normal programs.

We have the following general result.

THEOREM 4.3. *Let $\Psi=\langle P_1,P_2\rangle$ be a belief profile, and define $P_1\diamond P_2=\nabla(\Psi)$. Then, $\diamond$ satisfies the following versions of the postulates of Definition 2.4.*

($LS1'$) $P_1\diamond P_2\equiv_s P_2\diamond P_1$.
($LS2'$) $P_1\sqcap P_2\models_s P_1\diamond P_2$.
($LS3'$) *If $P_1\sqcap P_2$ is satisfiable then $P_1\diamond P_2\models_s P_1\sqcap P_2$.*
($LS4'$) *$P_1\diamond P_2$ is unsatisfiable iff P_1 is unsatisfiable and P_2 is unsatisfiable.*
($LS5'$) *If $P_1\equiv_s P_2$ and $P_1'\equiv_s P_2'$ then $P_1\diamond P_2\equiv_s P_1'\diamond P_2'$.*
($LS7'$) $P_1\diamond P_2\models_s P_1\sqcup P_2$.
($LS8'$) *If P_1 and P_2 are satisfiable then $P_1\sqcap(P_1\diamond P_2)$ is satisfiable.*

4.2. Basic Merging

For the second approach to merging, programs $P_1,\ldots,P_n$ are merged with respect to a target logic program P_0, so that the SE models in the merging are drawn from models of P_0. This operator is referred to as the (*basic*) *merging* of $P_1,\ldots,P_n$ with respect to P_0. The information in P_0 *must* hold in the merging, and so can be taken as *necessarily* holding. Konieczny and Pino Pérez [2002] call P_0 a set of *integrity constraints*, though this usage of the term differs from its usage in logic programs. Note that in the case where $SE(P_0)$ is the set of all SE models, the two approaches (of this section and the

Table II. Examples of Basic Merging

P_1	P_2	$SE(\Delta(\langle\emptyset, P_1, P_2\rangle))$
$\{p \leftarrow\}$	$\{q \leftarrow\}$	$\{(pq, pq)\}$
$\{p \leftarrow\}$	$\{\leftarrow p\}$	$\{(p,p), (\emptyset,\emptyset)\} \cup \{(p,\emptyset)\}$
$\{p \leftarrow \sim p\}$	$\{\leftarrow p\}$	$\{(\emptyset,p), (p,p), (\emptyset,\emptyset)\}$
$\{p \leftarrow, q \leftarrow\}$	$\{\leftarrow p, q\}$	$\{(pq,pq), (p,p), (q,q)\} \cup \{(p,pq), (q,pq)\}$
$\{\bot \leftarrow \sim p, \bot \leftarrow \sim q\}$	$\{\leftarrow p, q\}$	$\{S \in SE(\emptyset) \mid S \neq (\emptyset,\emptyset)\}$
$\{\bot \leftarrow p, \bot \leftarrow q\}$	$\{p; q \leftarrow\}$	$\{(\emptyset,\emptyset), (p,p), (q,q)\} \cup \{(p,\emptyset), (q,\emptyset)\}$

previous section) do not coincide, and that merging is generally a weaker operator than arbitration.

Definition 4.4. Let $\Psi = \langle P_0, \ldots, P_n\rangle$ be a belief profile and let $\overline{S}, \overline{T}$ be two SE models of Ψ (or two classical models of Ψ).

Then, define $\overline{S} \leq_b \overline{T}$, if $S_0 \ominus S_j \subseteq T_0 \ominus T_j$ for every $1 \leq j \leq n$.

As in the case of arbitration merging, $\leq_b$ is a partial preorder. Accordingly, let $Min_b(N)$ be the set of all minimal elements of a set N of tuples relative to $\leq_b$. In extending our notation for referring to components of tuples, we furthermore define $N_0 = \{S_0 \mid \overline{S} \in N\}$. We thus can state our definition for basic merging as follows:

Definition 4.5. Let $\Psi = \langle P_1, \ldots, P_n\rangle$ be a belief profile. Then, the *basic merging*, or simply *merging*, of Ψ, is a logic program $\Delta(\Psi)$ such that

$$SE(\Delta(\Psi)) = \{(X,Y) \mid Y \in Min_b(Mod(\Psi))_0, X \subseteq Y, \\ \text{and if } X \subset Y \text{ then } (X,Y) \in Min_b(SE(\Psi))_0\}\ ,$$

providing Ψ is satisfiable, otherwise, if P_i is unsatisfiable for some $1 \leq i \leq n$, define $\Delta(\Psi) = \Delta(\langle P_0, \ldots, P_{i-1}, P_{i+1}, \ldots, P_n\rangle)$.

Let us reconsider Programs P_1 and P_2 from (2) in the context of basic merging. To this end, we consider the belief profile $\langle\emptyset,\ \{p \leftarrow, u \leftarrow\},\ \{\leftarrow p, v \leftarrow\}\rangle$. We are now faced with 27 SE models for $SE(\langle\emptyset, P_1, P_2\rangle)$. Among them, we get the following $\leq_b$-minimal SE models

$$Min_b(SE(\langle\emptyset, P_1, P_2\rangle)) = \{\langle(uv,uv), (puv,puv), (uv,uv)\rangle, \\ \langle(uv,puv), (puv,puv), (uv,uv)\rangle, \langle(puv,puv), (puv,puv), (uv,uv)\rangle\}$$

along with $Min_b(Mod(\langle\emptyset, P_1, P_2\rangle)) = \{\langle uv, puv, uv\rangle, \langle puv, puv, uv\rangle\}$. We get:

$$\begin{aligned} Min_b(Mod(\langle\emptyset, P_1, P_2\rangle))_0 &= \{puv, uv\}, \\ Min_b(SE(\langle\emptyset, P_1, P_2\rangle))_0 &= \{(uv,uv), (uv,puv), (puv,puv)\}, \text{ and} \\ SE(\Delta(\langle\emptyset, P_1, P_2\rangle)) &= Min_b(SE(\langle\emptyset, P_1, P_2\rangle))_0\ . \end{aligned}$$

While arbitration resulted in $\nabla(\langle P_1, P_2\rangle) = \{p; \sim p \leftarrow, u \leftarrow, v \leftarrow\}$, the more conservative approach of basic merging yields $\Delta(\langle\emptyset, P_1, P_2\rangle) = \{u \leftarrow, v \leftarrow\}$.

We have just seen that basic merging adds "intermediate" SE models, viz. (uv, puv), to the ones obtained in arbitration merging. This can also be observed on the examples given in Table I, where every second merging is weakened by the addition of such intermediate SE models. This is made precise in Theorem 4.7 below. We summarize the results in Table II but omit the programs $\Delta(\langle\emptyset, P_1, P_2\rangle)$ because they are obtained from $\nabla(\langle P_1, P_2\rangle)$ in Table I by simply dropping all rules of form $p; \sim p \leftarrow$ and $q; \sim q \leftarrow$, respectively.

THEOREM 4.6. *Let Ψ be a belief profile, P_0 a program representing global constraints, and Δ as given in Definition 4.5. Then, Δ satisfies the following versions of the postulates of Definition 2.5:*

(IC0′) $\Delta(\langle P_0, \Psi\rangle) \models_s P_0$.
(IC1′) If P_0 and Ψ are satisfiable then $\Delta(\langle P_0, \Psi\rangle)$ is satisfiable.
(IC2′) If $\sqcap(P_0, \Psi)$ is satisfiable then $\Delta(\langle P_0, \Psi\rangle) \equiv_s P_0 \sqcap (\sqcap(\Psi))$.
(IC3′) If $P_0 \equiv_s P_0'$ and $\Psi \equiv_s \Psi'$ then $\Delta(\langle P_0, \Psi\rangle) \equiv_s \Delta(\langle P_0', \Psi'\rangle)$.
(IC4′) If $P_1 \models_s P_0$ and $P_2 \models_s P_0$ then:
if $\Delta(\langle P_0, P_1, P_2\rangle) \sqcap P_1$ is satisfiable, then $\Delta(\langle P_0, P_1, P_2\rangle) \sqcap P_2$ is satisfiable.
(IC5′) $\Delta(\langle P_0, \Psi\rangle) \sqcap \Delta(\langle P_0, \Psi'\rangle) \models_s \Delta(\langle P_0, \Psi \circ \Psi'\rangle)$.
(IC7′) $\Delta(\langle P_0, \Psi\rangle) \sqcap P_1 \models_s \Delta(\langle P_0 \sqcap P_1, \Psi\rangle)$.
(IC9′) Let Ψ' be a permutation of Ψ. Then, $\Delta(\langle P_0, \Psi\rangle) \equiv_s \Delta(\langle P_0, \Psi'\rangle)$.

We also obtain that arbitration merging is stronger than (basic) merging in the case of tautologous constraints in P_0.

THEOREM 4.7. *Let Ψ be a belief profile. Then $\nabla(\Psi) \models_s \Delta(\langle \emptyset, \Psi\rangle)$.*

As well, for belief profile $\Psi = \langle P_1, P_2\rangle$, we can express our merging operators in terms of the revision operator defined in Section 3.2.

THEOREM 4.8. *Let $\langle P_1, P_2\rangle$ be a belief profile.*

(1) $\nabla(\langle P_1, P_2\rangle) = (P_1 * P_2) \sqcup (P_2 * P_1)$.
(2) $\Delta(\langle P_1, P_2\rangle) = P_2 * P_1$.

Note that in the second part of the preceding result, P_1 is regarded as a set of constraints (usually with name P_0) according to our convention for basic merging. We note also that analogous results are obtained in the corresponding merging operators in propositional logic [Konieczny and Pino Pérez 2002; Liberatore and Schaerf 1998].

The final example further illustrates the difference between arbitration and basic merging. Take $P_1 = \{p \leftarrow, q \leftarrow\}$ and $P_2 = \{\sim p \leftarrow, \sim q \leftarrow\}$. Then, we have that $SE(\nabla(\langle P_1, P_2\rangle)) = \{(pq, pq), (\emptyset, \emptyset)\}$ and $SE(\Delta(\langle \emptyset, P_1, P_2\rangle)) = SE(\emptyset)$. That is, in terms of programs, we obtain

$$\nabla(\langle P_1, P_2\rangle) \;=\; \{p; \sim p \leftarrow,\; q; \sim q \leftarrow,\; \leftarrow p, \sim q,\; \leftarrow \sim p, q\} \text{ and } \Delta(\langle \emptyset, P_1, P_2\rangle) = \emptyset\,.$$

Thus in the arbitration, one essentially obtains that both p and q hold, or neither do. Thus, very informally, in merging P_1 and P_2 one obtains that the information in P_1 holds or that in P_2 does. This is also reflected in Theorem 4.8, where it can be observed that the SE models of the merging are drawn from those of P_1 and P_2. In basic merging this is not the case, and in merging the various possible combinations of truth values for p and q may hold, hence yielding a program with models given by $SE(\emptyset)$.

5. COMPUTING BELIEF CHANGE VIA ANSWER SET PROGRAMMING

In this section, we discuss computational aspects of our approach. More specifically, we provide encodings for our belief change operators in terms of fixed nonground ASP programs and give a general complexity analysis of the underlying reasoning tasks. We recall that nonground programs are defined over predicates of arbitrary arity which have either variables (denoted by upper-case letters) or constants (denoted by lower-case letters) as arguments. Such nonground programs can be seen as a compact representation of large programs without variables (and thus as propositional programs),

by considering the *grounding* of a program.[14] The nonground programs we define in this section can be seen as queries which take the (propositional) programs subject to revision or merging as an input database. Thus, we follow here the tradition of *meta-programming* (see, e.g., the works of Delgrande et al. [2003], Eiter et al. [2003], and Gebser et al. [2008]).

For encoding the cardinality-based revision operator, we make use well-known minimization statements [Gebser et al.; Simons et al. 2002], although similar optimization constructs like weak constraints [Leone et al. 2006] could be used as well, while for the set-based revision operator and the merging operators we need an inclusion-based account of minimization [Gebser et al. 2011], requiring the elevated complexity of disjunctive programs. Our goal in the encodings is to provide programs such that their answer sets characterize the SE models of the result of the encoded revision or merging problem. With these SE models at hand, corresponding programs can be obtained via the construction of canonical programs.

Before we start with the ASP encodings, we have to fix how programs subject to revision and merging are represented. For the sake of uniformity, we use belief profiles $\Psi = \langle P_\alpha, \ldots, P_n \rangle$, where

— for revision problems, we have $\alpha = 1$ and $n = 2$ (and so $\langle P_1, P_2 \rangle$ here represents revision problem $P_1 * P_2$),
— for arbitration problems, we have $\alpha = 1$ and $n \geq 2$, and
— for basic merging, we use $\alpha = 0$ and $n \geq 2$.

Moreover, we assume in this section that every P_i is satisfiable, that is, $SE(P_i) \neq \emptyset$.

Given a belief profile $\Psi = \langle P_\alpha, \ldots, P_n \rangle$, we use four ternary predicates, *phead*, *nhead*, *pbody*, and *nbody* to represent Ψ. For each predicate, the first argument i indices the program (i.e., i is a number between α and n), the second argument contains the rule identifier $\#r$ of a rule $r \in P_i$, and the third argument is an atom, indicating that this atom occurs in the positive or negative head or the positive or negative body of rule $r \in P_i$, respectively. For example, let $\Psi = \langle P_1, P_2 \rangle$ with $P_1 = \{\leftarrow \sim p, \ \leftarrow \sim q\}$ and $P_2 = \{p; q \leftarrow, \ \leftarrow p, q\}$. We obtain the *relational representation of* Ψ by[15]

$$\begin{aligned}[\Psi] \ = \ \{&nbody(1,1,p),\ nbody(1,2,q),\\ &phead(2,1,p),\ phead(2,1,q),\ pbody(2,2,p),\ pbody(2,2,q)\}.\end{aligned}$$

Here, we just use numbers as rule identifiers, that is, $\#(\leftarrow \sim p) = \#(p; q \leftarrow) = 1$ and $\#(\leftarrow \sim q) = \#(\leftarrow p, q) = 2$. The only necessary requirement is that different rules are assigned to different identifiers, that is, $r \neq r'$ implies $\#r \neq \#r'$.

In general, we define the relational representation of a belief profile as follows.

Definition 5.1. Let $\Psi = \langle P_\alpha, \ldots, P_n \rangle$ be a belief profile. Then, the *relational representation* of Ψ is given by

$$\begin{aligned}[\Psi] \ = \ \bigcup_{i=\alpha}^{n} \bigcup_{r \in P_i} \Big(&\{phead(i, \#r, a) \mid a \in H(r)^+\} \cup \{nhead(i, \#r, a) \mid a \in H(r)^-\} \cup \\ &\{pbody(i, \#r, a) \mid a \in B(r)^+\} \cup \{nbody(i, \#r, a) \mid a \in B(r)^-\}\Big).\end{aligned}$$

[14] Recall that the grounding of a program P is given by the union of the groundings of its rules, and the grounding of a rule $r \in P$ is the set obtained by all possible substitutions of variables in r by constants occurring in P; cf. Dantsin et al. [2001] for a more thorough exposition.

[15] Since we have here rules which are all simple facts, we omit the "$\leftarrow$"-symbol for rules.

We assume here that all i and $\#r$ are given as numbers. Following datalog notation, we write, for a program P and a belief profile Ψ, $P[\Psi]$ instead of $P \cup [\Psi]$.

We provide our encodings in a modular way. That is, we introduce various sets of rules which implement different aspects required to solve the respective problem. We start with some basic modules, which are used in most of the encodings. Then, we provide our results for revision and conclude with the encodings for merging.

5.1. Basic Modules

We start with a simple fragment which contains some domain predicates and fixes some designated identifiers.

Definition 5.2.

$$\begin{aligned} P_{domain} \;=\; & \{prog\_rule(P,R) \leftarrow \eta(P,R,A),\ dom(A) \leftarrow \eta(P,R,A) \mid \\ & \qquad \eta \in \{phead, pbody, nhead, nbody\}\} \cup \\ & \{prog(P) \leftarrow prog\_rule(P,R), \\ & \ model(c) \leftarrow,\ model(t) \leftarrow,\ model(h) \leftarrow, \\ & \ prog\_model(c) \leftarrow,\ prog\_model(t) \leftarrow\}. \end{aligned}$$

Predicates $prog\_rule(\cdot,\cdot)$, $dom(\cdot)$, and $prog(\cdot)$ are used to gather information from a conjoined input $[\Psi]$; the designated constants c, t, h are used later on to distinguish between different guesses for models. Specifically, c refers to classical models while h and t refer to the first and second part of SE models, respectively. Thus, c and t indicate models of the programs (hence the use of $prog\_model$), while h indicates models of a program reduct.

The following code guesses such models for each program P in the belief profile Ψ. The guess is accomplished in Rules (3) and (4) below which assign each atom A in the domain to be $in(\cdot)$ or $out(\cdot)$.[16]

Definition 5.3.

$$\begin{aligned} P_{models} \;=\; & \{in(P,A,M) \leftarrow {\sim}out(P,A,M), prog(P), dom(A), model(M), && (3)\\ & \ out(P,A,M) \leftarrow {\sim}in(P,A,M), prog(P), dom(A), model(M), && (4)\\ & \ \leftarrow in(P,A,h), out(P,A,t), && (5)\\ & \ diff(P,Q,A,M) \leftarrow in(P,A,M), out(Q,A,M), && (6)\\ & \ diff(P,Q,A,M) \leftarrow out(P,A,M), in(Q,A,M), && (7)\\ & \ ok(P,R,M) \leftarrow in(P,A,M), phead(P,R,A), model(M), && (8)\\ & \ ok(P,R,M) \leftarrow out(P,A,M), pbody(P,R,A), model(M), && (9)\\ & \ ok(P,R,M) \leftarrow in(P,A,M), nbody(P,R,A), prog\_model(M), && (10)\\ & \ ok(P,R,M) \leftarrow out(P,A,M), nhead(P,R,A), prog\_model(M), && (11)\\ & \ ok(P,R,h) \leftarrow in(P,A,t), nbody(P,R,A), && (12)\\ & \ ok(P,R,h) \leftarrow out(P,A,t), nhead(P,R,A), && (13)\\ & \ \leftarrow {\sim}ok(P,R,M), prog\_rule(P,R), model(M)\}. && (14) \end{aligned}$$

This allows us to draw a one-to-one correspondence between answer sets of the encoding and models (resp., SE models) of the programs in the belief profile. Note that Rule (5) excludes guesses where the corresponding SE model (X,Y) would not satisfy

[16]Note that we present our encodings in accord with the language introduced in Section 2. Unlike this, our implementation uses common choice rules, and thus avoids auxiliary predicates such as $out(\cdot)$.

$X \subseteq Y$. To make this intuition a bit more precise, let us define the following (projection) operators for a set S of ground atoms and a number i:

$$\pi^i_{Mod}(S) = \{a \mid in(i,a,c) \in S\};$$
$$\pi^i_{SE}(S) = \big(\{a \mid in(i,a,h) \in S\}, \{b \mid in(i,b,t) \in S\}\big).$$

The next Rules (6)–(7) indicate whether atom A is assigned differently (via predicate $diff(\cdot,\cdot,\cdot,\cdot)$) for two programs. This predicate is useful later and we will capture its idea formally below. Rules (8)–(13) tell us which rules (in which programs) are satisfied by the respective guess.

We observe that the answer sets of the program $P[\Psi]$ where $P = P_{domain} \cup P_{models}$ are in a one-to-one correspondence with the models and SE models of belief profile Ψ. We summarize our observations formally as follows.

LEMMA 5.4. *Given* $\Psi = \langle P_\alpha, \ldots, P_n \rangle$, *then*

$$\{(M,N) \mid M \in Mod(\Psi), N \in SE(\Psi)\} =$$
$$\{(\langle \pi^\alpha_{Mod}(S), \ldots, \pi^n_{Mod}(S)\rangle, \langle \pi^\alpha_{SE}(S), \ldots, \pi^n_{SE}(S)\rangle) \mid S \in AS(P[\Psi])\}.$$

We moreover observe the following relations concerning the *diff* predicate in view of Lemma 5.4.

LEMMA 5.5. *For* $\Psi = \langle P_\alpha, \ldots, P_n \rangle$, $1 \le i,j \le n$, *and* $S \in AS(P[\Psi])$,

$$\pi^i_{Mod}(S) \ominus \pi^j_{Mod}(S) = \{a \mid diff(i,j,a,c) \in S\};$$
$$\pi^i_{SE}(S) \ominus \pi^j_{SE}(S) = \big(\{a \mid diff(i,j,a,h) \in S\}, \{b \mid diff(i,j,b,t) \in S\}\big).$$

Finally, we define a module which takes the models and SE models, respectively, of some selected program (this is done via the *selector* predicate; in the case of revision, it is program P_2, thus $selector(2)$ is specified for revision problems) and copies them into a designated predicate.

Definition 5.6.

$$\begin{aligned} P_{result} = \{ & total \leftarrow {\sim}nontotal, \\ & nontotal \leftarrow {\sim}total, \\ & \leftarrow nontotal, selector(S), in(S,A,t), out(S,A,c), \\ & \leftarrow nontotal, selector(S), out(S,A,t), in(S,A,c), \\ & resultH(A) \leftarrow selector(S), in(S,A,h), nontotal, \\ & resultH(A) \leftarrow selector(S), in(S,A,c), total, \\ & resultT(A) \leftarrow selector(S), in(S,A,c)\}. \end{aligned}$$

The intuition for the module is as follows: we either generate a total SE model (Y,Y) or a nontotal SE model (X,Y) with $X \subset Y$. Thus, the guess between predicates *total* and *nontotal*. In case we want to derive a nontotal SE model (X,Y), we have to make sure that Y coincides with the classical model we guessed.[17] This is done by the two constraints. The remaining lines fill the predicates *resultH* and *resultT* accordingly,

[17] One might ask why we use the different concepts of t- and c-models. The reason is that there might be a minimal difference between (X_1,Y_1) and (X_2,Y_2) although there is no minimal difference between Y_1 and Y_2. But then we still need those interpretations Y in order to compute the corresponding interpretations X. On the other hand, there might be a minimal distance between Y_1 and Y_2 but not between any (X_1,Y_1) and (X_2,Y_2). Still, we then want (Y_2,Y_2) in the result.

where atoms in *resultH* yield the X of SE model (X, Y) and atoms in *resultT* yield the Y of the SE model.

The following straightforward observation paves the way for all our subsequent encoding.

LEMMA 5.7. *Given* $\Psi = \langle P_\alpha, \ldots, P_n \rangle$ *and* $P[\Psi]$, *where*

$$P = P_{domain} \cup P_{models} \cup P_{result} \cup \{selector(i)\} \qquad \text{for some } i \geq \alpha \geq 0.$$

For any set S *of ground atoms, define*

$$\rho(S) = (\{a \mid resultH(a) \in S\}, \{b \mid resultT(b) \in S\}).$$

Then, we have $SE(P_i) = \{\rho(S) \mid S \in AS(P[\Psi])\}$.

5.2. Encodings for Revision

The idea of our ASP encodings is based on the observation that all change operations we have considered in this article select distinguished (SE) models. For instance, all SE models of $P * Q$ are among $\{(X, X) \mid X \in Mod(Q)\} \cup SE(Q)$. In view of Lemma 5.7, it is then enough to select the appropriate SE models among those in $SE(P_i)$ for a given i. In ASP, the selection of models can be accomplished by optimization statements, expressing objective functions. This proceeding is best illustrated by cardinality-based revision, detailed next.

5.2.1. Cardinality-Based Revision. For implementing a cardinality-based preference criterion, we make use of optimization statements: A *#minimize* statement is of the form[18]

$$\#minimize[\ell_1@L_1, \ldots, \ell_k@L_k]\,.$$

Besides literals ℓ_j for $1 \leq j \leq k$, a *#minimize* statement includes integers L_j providing priority levels. The *#minimize* statements in a program P distinguish optimal answer sets of P in the following way. For any set X of atoms and integer L, let Σ_L^X denote the number of literals ℓ_j such that $\ell_j@L$ occurs in some *#minimize* statement in P and ℓ_j holds with respect to X. We also call Σ_L^X the utility of X at priority level L. An answer set X of P is dominated if there is an answer set Y of P such that $\Sigma_L^Y < \Sigma_L^X$ and $\Sigma_{L'}^Y = \Sigma_{L'}^X$ for all $L' > L$, and optimal otherwise. Note that greater priority levels are more significant than smaller ones, which allows for representing sequences of several optimization criteria.

The selection of the SE models defined in Definition 3.10 can then be accomplished by means of the optimization statements

$$\#minimize[diff(1, 2, A, h)@2]\,, \tag{15}$$

$$\#minimize[diff(1, 2, A, c)@1, diff(1, 2, A, t)@1]\,, \tag{16}$$

where A ranges over the considered set of atoms. Denoting the selection function on answer sets implemented by (15) and (16) as $min_{||}$, we obtain the following result.

THEOREM 5.8. *Let* $\Psi = \langle P_1, P_2 \rangle$ *be a belief profile and*

$$P = P_{domain} \cup P_{models} \cup P_{result} \cup \{selector(2)\}.$$

Then, we have

$$SE(P_1 *_c P_2) = \{\rho(S) \mid S \in min_{||}(AS(P[\Psi]))\}.$$

[18]Minimize statements also contain weights, which are omitted for simplicity.

5.2.2. Set-Based Revision. For addressing set-based revision, we can proceed analogously to the above, once we replace the cardinality-based account of minimization by an inclusion-based account. To this end, we follow the approach of Gebser et al. [2011] and redefine the utility of minimization statements: For any set X of atoms and integer L, let Π_L^X denote the set of literals such that $\ell_j@L$ occurs in some #*minimize* statement in P and ℓ_j holds with respect to X. An answer set X of P is dominated if there is an answer set Y of P such that $\Pi_L^Y \subset \Pi_L^X$ and $\Pi_{L'}^Y = \Pi_{L'}^X$ for all $L' > L$, and optimal otherwise.

Interpreting the minimization statements in (15) and (16) under this inclusion-based semantics provides us with an implementation of set-based revision. To this end, let $min_{\subseteq}$ denote the selection function on answer sets implemented by (15) and (16) under the inclusion-based semantics.

THEOREM 5.9. *Let* $\Psi = \langle P_1, P_2 \rangle$ *a belief profile and*

$$P = P_{domain} \cup P_{models} \cup P_{result} \cup \{selector(2)\}.$$

Then, we have

$$SE(P_1 * P_2) = \{\rho(S) \mid S \in min_{\subseteq}(AS(P[\Psi]))\}.$$

5.3. Encodings for Merging

5.3.1. Basic Merging. We continue with the problem of basic merging. Suppose that belief profile $\Psi = \langle P_0, \ldots, P_n \rangle$ is given for some arbitrary n. Also recall that P_0 plays a special role in basic merging. In particular, the SE models of the result of the merging are taken from the SE models of P_0.

As before, in view of Lemma 5.7, it is sufficient to define the appropriate selection function on answer sets using

$$\#minimize[diff(0,P,A,h)@2 : P > 0],$$
$$\#minimize[diff(0,P,A,c)@1 : P > 0, diff(0,P,A,t)@1 : P > 0],$$

where A and P range over the considered sets of atoms and programs, respectively. The qualification $P > 0$ restricts the instantiation of P.

In analogy to the above, let min_{basic} denote the selection function on answer sets implemented by the last two optimization statements under the inclusion-based semantics. Then, our result is forthcoming as in the previous sections.

THEOREM 5.10. *Let* $\Psi = \langle P_0, \ldots, P_n \rangle$ *be a belief profile and*

$$P = P_{domain} \cup P_{models} \cup P_{result} \cup \{selector(0), prog(0)\}$$

Then,

$$SE(\Delta(\Psi)) = \{\rho(S) \mid S \in min_{basic}(AS(P[\Psi]))\}.$$

Note that the fact $prog(0)$ is added for the case that no integrity constraints are specified by the input and recall that $selector(0)$ is used to select P_0 as the program which takes care of the result predicates.

5.3.2. Arbitration Merging. Our final encoding is the one for arbitration merging, which again does not require further modules, except that we need a somewhat more complicated program to prepare the *resultH* and *resultT* predicates, since arbitration merging collects SE models from *all* programs of the belief profile rather than from a single program (which has been the case in the approaches we encoded so far). We thus do not use a *selector* predicate here but instead provide a new result module below. Also recall that belief profiles for arbitration merging are of the form $\langle P_1, \ldots, P_n \rangle$.

Here is the new result module.

Definition 5.11.

$$\begin{aligned} P'_{result} = \{ & tout(I); tout(J) \leftarrow prog(I), prog(J), I \neq J, \\ & tselect(I) \leftarrow {\sim}tout(I), prog(I), \\ & total \leftarrow {\sim}nontotal, \\ & nontotal \leftarrow {\sim}total, \\ & resultT(A) \leftarrow in(M,A,c), tselect(M), \\ & resultH(A) \leftarrow in(M,A,c), total, tselect(M), \\ & hout(I); hout(J) \leftarrow prog(I), prog(J), I \neq J, nontotal, \\ & hselect(I) \leftarrow {\sim}hout(I), prog(I), nontotal, \\ & \leftarrow nontotal, in(I,A,t), out(J,A,c), tselect(J), hselect(I), nontotal, \\ & \leftarrow nontotal, out(I,A,t), in(J,A,c), tselect(J), hselect(I), nontotal, \\ & resultH(A) \leftarrow in(I,A,h), hselect(I), nontotal\}. \end{aligned}$$

Roughly speaking, the first two rules select exactly one program P_i from the belief profile. We then guess whether we build a total or a nontotal SE model (as we did in P_{result}). Then, we copy the model from the guessed program into the there-part of the result, and in case we are constructing a total SE model, also in the here-part. If we construct a nontotal SE model, we guess a second program P_j from the belief profile and check whether the there-part of the current SE model of P_j coincides with the classical model of P_i (this is done by the two constraints). If this check is passed, we copy the here-part of the SE model of P_j into the here-part of the resulting SE model.

The minimize statement is as follows:

$$\begin{aligned} & \#minimize[diff(Q,P,A,h) : P \neq Q], \\ & \#minimize[diff(Q,P,A,c) : P \neq Q, diff(Q,P,A,t) : P \neq Q], \end{aligned}$$

where A and P, Q range over the considered sets of atoms and programs, respectively; and $P \neq Q$ restricts the instantiation of P and Q.

Defining min_{arb} accordingly under the inclusion-based semantics, provides us with the following result.

THEOREM 5.12. *Let* $\Psi = \langle P_1, \ldots, P_n \rangle$ *be a belief profile and define* $P = P_{domain} \cup P_{models} \cup P'_{result}$. *Then,*

$$SE(\nabla(\Psi)) = \{\rho(S) \mid S \in min_{arb}(AS(P[\Psi]))\}.$$

5.4. Computational Complexity

We next address the computational complexity of our approach to revision and merging. As it turns out, our method of using ASP itself to compute revision and merging is adequate from a complexity point of view.

First, we recapitulate the complexity classes relevant in what follows. As usual, for any complexity class C, by co-C we understand the class of all problems which are complementary to the problems in C. Furthermore, for C as before and complexity class A, the notation C^A stands for the *relativized version* of C, consisting of all problems which can be decided by Turing machines of the same sort and time bound as in C, only that the machines have access to an oracle for problems in A.

Four complexity classes are relevant here, viz. NP, Σ_2^P, Π_2^P, and Θ_2^P, which are defined thus:

— NP consists of all decision problems which can be solved with a nondeterministic Turing machine working in polynomial time;

— $\Sigma_2^P = \mathrm{NP}^{\mathrm{NP}}$;
— $\Pi_2^P = \text{co-}\Sigma_2^P$; and
— Θ_2^P is the class of all problems solvable on a deterministic Turing machine in polynomial time asking on input x a total of $O(\log |x|)$ many oracle calls to NP (thus, Θ_2^P is also denoted by $\mathrm{P}^{\mathrm{NP}[\log n]}$).

Observe that NP, Σ_2^P, and Π_2^P are part of the polynomial hierarchy, which is given by the following sequence of objects: the initial elements are

$$\Delta_0^P = \Sigma_0^P = \Pi_0^P = \mathrm{P};$$

and, for $i > 0$,

$$\Delta_i^P = \mathrm{P}^{\Sigma_{i-1}^P}; \qquad \Sigma_i^P = \mathrm{NP}^{\Sigma_{i-1}^P}; \qquad \Pi_i^P = \text{co-}\mathrm{NP}^{\Sigma_{i-1}^P}.$$

Here, P is the class of all problems solvable on a deterministic Turing machine in polynomial time. It holds that $\Sigma_1^P = \mathrm{NP}$, $\Sigma_2^P = \mathrm{NP}^{\mathrm{NP}}$, and $\Pi_2^P = \text{co-}\mathrm{NP}^{\mathrm{NP}}$. A problem is said to be at the *k-th level* of the polynomial hierarchy iff it is in Δ_{k+1}^P and either Σ_k^P-hard or Π_k^P-hard.

We first consider the worst-case complexity of our approach to set-containment based revision. The standard decision problem for revision in classical logic is:

Given formulas P, Q, and 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 5.13. *Deciding whether $P * Q \models_s R$ holds, for given GLPs P, Q, and 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 result holds for the cautious revision operator $*_w$ as well.

For cardinality-based revision, we obtain the following result, again mirroring a similar behavior for the classical case.

THEOREM 5.14. *Deciding whether $P *_c Q \models_s R$ holds, for given GLPs P, Q, and R, is in Θ_2^P.*

Concerning merging, we have the following result.

THEOREM 5.15. *Given a belief profile Ψ and a program R, deciding $\nabla(\Psi) \models_s R$ (resp., $\Delta(\Psi) \models_s R$) is Π_2^P-complete.*

5.5. Technical Remarks

All encodings presented in the previous subsections are publicly available and can be downloaded via the following URL: `http://www.cs.uni-potsdam.de/~torsten/ASPChange.tgz`. In contrast to the description of the encodings as given above, which follow the definition of our logic-programming language from Section 2, the programs available for download are somewhat simpler as they make use of extended language constructs. In particular, they rely on so-called *choice rules* [Simons et al. 2002],

avoiding the use of cyclic rules with auxiliary predicates. As an example, consider the first two rules in P_{result} in Definition 5.6. The pair of rules

$$total \leftarrow \sim nontotal \quad \text{and} \quad nontotal \leftarrow \sim total$$

uses the auxiliary atom *nontotal* to indicate that *total* may not belong to an answer set. This can be equivalently expressed by using instead the single choice rule

$$\{total\} \leftarrow,$$

making the auxiliary atom *nontotal* unnecessary. In addition, one must then replace every occurrence of *nontotal* by $\sim total$ in the rest of the program. The same can be done with rules (3) and (4) of Definition 5.3. In fact, we may analogously treat the disjunctive rules in P'_{result}. The reason for taking pairs of rules in the first place is motivated by complexity considerations given that disjunctions give rise to an elevated complexity (which is of no concern in P'_{result} in view of the already elevated complexity of arbitration merging).

In fact, from an encoding perspective, the elevated complexity of set-based revision and both merging operations is reflected by the usage of inclusion-based preference criteria. To accommodate this, we rely upon the approach of Gebser et al. [2011] that allows for interpreting #*minimize* statement in various ways. This approach takes advantage of recent advances in ASP grounding technology, admitting an easy use of meta-modeling techniques. The idea is to reinterpret existing optimization statements in order to express complex preferences among answer sets. While, for instance, in the ASP solver `smodels` [Simons et al. 2002], the meaning of #*minimize* is to compute answer sets incurring minimum costs, we may alternatively use it for selecting inclusion-minimal answer sets.

Furthermore, our encodings are given via certain language fragments of nonground ASP such that their respective data complexity matches the complexity of the encoded task. Recall that data complexity addresses problems over programs $P \cup D$ where a nonground program P is fixed, while the input database D (a set of ground atoms) is the input of the decision problem. As is well known, the data complexity of the problem whether atom a is contained in all answer sets of $P \cup D$ is Π_2^P-complete for disjunctive programs [Dantsin et al. 2001] and Δ_2^P-complete for normal programs with optimization constructs [Simons et al. 2002]. Moreover, it is Θ_2^P-complete whenever there is only a single optimization construct used.

6. DISCUSSION

We have addressed the problem of belief change in logic programming under the answer set semantics. Our overall approach is based on a monotonic characterization of logic programs, given in terms of the set of SE models of a program. Based on the latter, we first defined and examined operators for logic program expansion and revision. Both subset-based revision and cardinality-based revision were considered. As well as giving properties of these operators, we also considered their complexity. This work is novel, in that it addresses belief change in terms familiar to researchers in belief revision: expansion is characterized in terms of intersections of models, and revision is characterized in terms of minimal distance between models.

We also addressed the problem of merging logic programs under the answer set semantics. Again, the approaches are based on a monotonic characterization of logic programs, given in terms of the set of SE models of a sequence of programs. We defined and examined two operators for logic program merging, the first following intuitions from arbitration [Liberatore and Schaerf 1998], the second being closer to

IC merging [Konieczny and Pino Pérez 2002]. As well as giving properties of these operators, we also considered complexity questions.

Last, we provided encodings for computing the revision and merging operators described previously. These encodings were carried out within the same logic programming framework. This allows for a direct implementation of our approach in terms of available answer set solvers. As well, these encodings also pragmatically demonstrate the fact that our change operators do not increase the complexity of the base formalism.

We note that previous work on logic program belief change was formulated at the level of the individual program, and not in terms of an abstract characterization (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 are lacking. (On the other hand, perhaps this criticism is not totally warranted, since we have also claimed that such work is best regarded as addressing *belief base change*; and in classical belief revision, it is typically difficult in general to obtain desirable properties in such approaches.) The main point of departure for the current approach then is to lift the problem of logic program revision or merging from the program (or syntactic) level to an abstract (or semantic) level. Notably, since all our operators are defined via semantic characterizations, the results of revision, expansion, and merging are independent of the particular syntactic expression of a logic program.

A continuation of our method was recently taken up by Slota and Leite [2010], who, inspired by our preliminary work [Delgrande et al. 2008, 2009], discussed answer set program update based on SE models using modified adaptions of the update postulates by Katsuno and Mendelzon [1992]. They define a set-based update operator and show that it satisfies their modified update postulates. However, it is also shown that update operators satisfying certain conditions Slota and Leite consider as reasonable, do not respect support, that is, that atoms true in an answer set may not be the consequence of a rule whose body is true.

Finally, we noted at the outset that strong equivalence coincides with equivalence in the logic of *here and there* (HT), a logic that can be seen as being intermediate between intuitionistic logic and classical logic. Moreover, equivalence between programs in HT corresponds to equality between sets of SE models. Hence, the results reported here on the revision and merging of logic programs can also be viewed as addressing the same concerns in the logic of HT. Consequently, the present approach may provide an appropriate starting point for developing a more general belief change formalism in intuitionistic and other related nonclassical logics, thus complementing existing research in this direction [Gabbay et al. 2008].

A. APPENDIX

A.1. Proof of Theorem 3.2

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

(1) We show that Definition 3.1 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 $P \models_s Q$, 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. □

A.2. Proof of Theorem 3.5

(*RA*1) This postulate follows immediately from Definition 3.4. Note that $(X, Y) \in SE(P * Q)$ only if $Y \in \sigma(Mod(Q), Mod(P))$, and therefore $(Y, Y) \in \sigma(SE(Q), SE(P))$. So, $SE(P * Q)$ is well-defined.
(*RA*2) If $P + Q$ is satisfiable, then we have that both $\sigma(Mod(Q), Mod(P)) \neq \emptyset$ and $\sigma(SE(Q), SE(P)) \neq \emptyset$. Further, for $Y \in Mod(Q)$ (or $(X, Y) \in SE(Q)$), there is some $Y' \in Mod(P)$ (resp., $(X', Y') \in SE(P)$) such that $Y \ominus Y' = \emptyset$ (resp., $(X, Y) \ominus (X', Y') = \emptyset$), from which our result follows.
(*RA*3) From Definition 3.4 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 3.3). Hence, $SE(P * Q) \neq \emptyset$.
(*RA*4) Immediate from Definition 3.4.
(*RA*5) If $SE(P) = \emptyset$, then the result follows immediately from the first part of Definition 3.4.
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 (*RA*1) 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 then $(X, Y) = (Y, Y)$, and $(Y, Y) \in SE(P * Q)$, we have that $Y \in \sigma(Mod(Q), Mod(P))$. Hence, from Definition 3.3, $Y \in Mod(Q)$ 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 $Y_1 \ominus Y_2 \subset Y \ominus Y'$.
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 $Y_1 \ominus Y_2 \subset Y \ominus Y'$. 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 $Y_1 \ominus Y_2 \subset Y \ominus Y'$.
Thus, from Definition 3.3, 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 3.3, 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 $(U,V) \ominus (U',V') \subset (X,Y) \ominus (X',Y')$.
Therefore, there is no $(U,V) \in SE(Q+R)$ and no $(U',V') \in SE(P)$ such that $(U,V) \ominus (U',V') \subset (X,Y) \ominus (X',Y')$.
We previously showed that $(X,Y) \in SE(Q+R)$. Consequently, from Definition 3.4, 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.

A.3. Proof of Theorem 3.6

For initialization, idempotency, and tautology, in the left-hand side of the given equivalence, revision corresponds with expansion via (*RA*2), 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 3.2, 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. □

A.4. Proof of Theorem 3.8

We need to show that $SE(P * Q) \subseteq SE(P *_w Q)$. 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 $X \subset Y$. Then, $(X,Y) \in \sigma(SE(P), SE(Q))$ by Definition 3.4, and $(X,Y) \in SE(P *_w Q)$ by Definition 3.7.
(2) $(Y,Y) \in SE(P * Q)$. From Definition 3.4, 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 3.7, $(Y,Y) \in SE(P *_w Q)$.

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

A.5. Proof of Theorem 3.11

Before giving the proof, we first present a lemma that is key for postulates (*RA*5) and (*RA*6).

LEMMA A.1. *Let E_1, E_2, and E_3 be SE interpretations. If $\sigma_{||}(E_1,E_2) \cap E_3 \neq \emptyset$, then $\sigma_{||}(E_1,E_2) \cap E_3 = \sigma_{||}(E_1 \cap E_3, E_2)$.*

PROOF. Assume that $\sigma_{||}(E_1,E_2) \cap E_3 \neq \emptyset$.

For showing $\subseteq$ in the equality, let $(X,Y) \in \sigma_{||}(E_1,E_2) \cap E_3$ and, toward a contradiction, assume that $(X,Y) \notin \sigma_{||}(E_1 \cap E_3, E_2)$.

Since $(X,Y) \in \sigma_{||}(E_1,E_2)$, so $(X,Y) \in E_1$; as well, $(X,Y) \in E_3$, so $(X,Y) \in E_1 \cap E_3$. Since $(X,Y) \notin \sigma_{||}(E_1 \cap E_3, E_2)$ we have that there is some $(X',Y') \in E_1 \cap E_3$ and some $(U',V') \in E_2$ such that for every $(U,V) \in E_2$, $|(X',Y') \ominus (U',V')| < |(X,Y) \ominus (U,V)|$. But this contradicts the assumption that $(X,Y) \in \sigma_{||}(E_1,E_2)$. Hence, the assumption that $(X,Y) \notin \sigma_{||}(E_1 \cap E_3, E_2)$ cannot hold, that is, $(X,Y) \in \sigma_{||}(E_1 \cap E_3, E_2)$, establishing that $\sigma_{||}(E_1,E_2) \cap E_3 \subseteq \sigma_{||}(E_1 \cap E_3, E_2)$.

To show $\supseteq$ in the equality, let $(X,Y) \in \sigma_{||}(E_1 \cap E_3, E_2)$ and, toward a contradiction, assume that $(X,Y) \notin \sigma_{||}(E_1,E_2) \cap E_3$.

Since $(X,Y) \in \sigma_{||}(E_1 \cap E_3, E_2)$, we get that $(X,Y) \in E_3$. Hence, $(X,Y) \notin \sigma_{||}(E_1,E_2)$ (via the assumption that $(X,Y) \notin \sigma_{||}(E_1,E_2) \cap E_3$).

We also have by assumption that $\sigma_{||}(E_1, E_2) \cap E_3 \neq \emptyset$, and so let $(X', Y') \in \sigma_{||}(E_1, E_2) \cap E_3$. Then, from the first part above, we have that $(X, Y) \in \sigma_{||}(E_1 \cap E_3, E_2)$. Thus, we have both that $(X, Y) \in \sigma_{||}(E_1 \cap E_3, E_2)$ and $(X', Y') \in \sigma_{||}(E_1 \cap E_3, E_2)$. Consequently, we obtain that

$$min(\{|(X, Y) \ominus (U, V)| \mid (U, V) \in E_2\}) = min(\{|(X', Y') \ominus (U, V)| (U, V) \in E_2\}).$$

Therefore, since $(X', Y') \in \sigma_{||}(E_1, E_2)$, so also $(X, Y) \in \sigma_{||}(E_1, E_2)$. But this together with $(X, Y) \in E_3$ contradicts our assumption that $(X, Y) \notin \sigma_{||}(E_1, E_2) \cap E_3$; that is, we have $(X, Y) \in \sigma_{||}(E_1, E_2) \cap E_3$, establishing that $\sigma_{||}(E_1, E_2) \cap E_3 \supseteq \sigma_{||}(E_1 \cap E_3, E_2)$. □

We now move on to the proof of Theorem 3.11.

($RA1$): This follows immediately from Definition 3.10. Note that $(X, Y) \in SE(P *_c Q)$ only if $Y \in \sigma_{||}(Mod(Q), Mod(P))$, and therefore $(Y, Y) \in \sigma_{||}(SE(Q), SE(P))$. So, $SE(P *_c Q)$ is well-defined.

($RA2$): If $P + Q$ is satisfiable, then we have that both $\sigma_{||}(Mod(Q), Mod(P)) \neq \emptyset$ 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 some $Y' \in Mod(P)$ (resp., $(X', Y') \in SE(P)$) such that $Y \ominus Y' = \emptyset$ ($(X, Y) \ominus (X', Y') = \emptyset$), from which our result follows.

($RA3$): From Definition 3.10 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 3.9). Hence, $SE(P * Q) \neq \emptyset$.

($RA4$): This is immediate from Definition 3.10.

($RA5$), ($RA6$): For $P *_c Q$, if $SE(P) = \emptyset$, we have that $(P *_c Q) + R = Q + R = (P *_c (Q + R))$.

So, assume that $SE(P) \neq \emptyset$. We show that $SE(P *_c Q) + SE(R) = SE(P *_c (Q + R))$, thus establishing both postulates.

For $\subseteq$, assume that $(X, Y) \in SE(P *_c Q) + R$. Thus, $(X, Y) \in SE(P *_c Q)$ and $(X, Y) \in R$.

For $X \subseteq Y$, we have that $Y \in \sigma_{||}(Mod(Q), Mod(P))$ and as well $Y \in Mod(R)$. We get that $Y \in \sigma_{||}(Mod(Q+R), Mod(P))$ by the analogous proof in propositional logic for cardinality-based revision.

For $X \subset Y$, we have that $(X, Y) \in \sigma_{||}(SE(Q), SE(P))$ and as well $(X, Y) \in SE(R)$. By Lemma A.1 we get that $(X, Y) \in \sigma_{||}(SE(Q + R), SE(P))$.

This establishes one direction of the set equality. For $\supseteq$, the argument is essentially the same, though in the reverse direction, and again appealing to Lemma A.1.

A.6. Proof of Theorem 3.12

The proof is the same as for Theorem 3.6.

A.7. Proof of Theorem 4.3

The definitions for arbitration and basic merging (Definitions 4.2 and 4.5) are essentially composed of two parts (as are the definitions for revision): there is a phrase to deal with classical propositional models (or SE models of form (Y, Y)) and then general SE models. For brevity, and because the case for propositional models follows immediately from the case of general SE models, we consider general SE models in the proofs here.

($LS1'$)–($LS7'$). These all follow trivially or straightforwardly from the definition of $P_1 \diamond P_2$.

($LS8'$). Assume that P_1 and P_2 are satisfiable. It follows that $SE(\langle P_1, P_2 \rangle) \neq \emptyset$ and so $Min_a(SE(\langle P_1, P_2 \rangle)) \neq \emptyset$. Let $\langle S_1, S_2 \rangle \in Min_a(SE(\langle P_1, P_2 \rangle))$, and so $S_1, S_2 \in$

$SE(P_1 \diamond P_2)$. Since $S_1 \in SE(P_1)$ we get that $S_1 \in SE(P_1) \cap SE(P_1 \diamond P_2)$ and so $S_1 \in SE(P_1 \sqcap (P_1 \diamond P_2))$. Thus, $P_1 \sqcap (P_1 \diamond P_2)$ is satisfiable.

A.8. Proof of Theorem 4.6

Let Ψ be a belief profile, P_0 a program representing global constraints, and Δ as given in Definition 4.5.

($IC0'$)–($IC3'$), ($IC9'$). These follow trivially or straightforwardly from the definition of $\Delta(\langle P_0, \Psi\rangle)$.

($IC4'$). Assume that $P_1 \models_s P_0$ and $P_2 \models_s P_0$. If $SE(P_1) \cap SE(P_2) \neq \emptyset$ then by ($IC2'$) we have that $\Delta(\langle P_0, P_1, P_2\rangle) = P_0 \sqcap P_1 \sqcap P_2$ from which our result follows immediately.

Consequently, assume that $SE(P_1) \cap SE(P_2) = \emptyset$. As well, assume the antecedent condition of the postulate that $\Delta(\langle P_0, P_1, P_2\rangle) \sqcap P_1$ is satisfiable. Let $\Psi = \langle P_0, P_1, P_2\rangle$. Thus, we have for some (X, Y) that $(X, Y) \in SE(\Delta(\Psi) \sqcap P_1)$, and so $(X, Y) \in SE(P_0) \cap SE(P_1)$, where $(X, Y) \in Min_b(SE(\Psi))_0$.

$(X, Y) \in Min_b(SE(\Psi))_0$ implies that there is some $(X', Y') \in SE(P_2)$ such that $\overline{S} = \langle (X, Y), (X, Y), (X', Y')\rangle \in Min_b(SE(\Psi))$.

We claim that $\overline{S}' = \langle (X', Y'), (X, Y), (X', Y')\rangle \in Min_b(SE(\Psi))$. This is sufficient to prove our result, since $\overline{S}' \in Min_b(SE(\Psi))$ yields that $(X', Y') \in \Delta(\Psi)$ and $(X', Y') \in SE(P_2)$. That is to say, $\Delta(\Psi) \sqcap P_2$ is satisfiable.

Proof of claim: Since $\overline{S} \in Min_b(SE(\Psi))$, this means that for every $\overline{T} \in SE(\Psi)$ we have that $\overline{T} \leq_b \overline{S}$ implies that $\overline{S} \leq_b \overline{T}$.

Consider $\overline{T} = \langle (U_0, V_0), (U_1, V_1), (U_2, V_2)\rangle$. If $\overline{T} \leq_b \overline{S}$ then we have that $(U_0, V_0) \ominus (U_1, V_1) \subseteq (X, Y) \ominus (X, Y) = (\emptyset, \emptyset)$. That is, $U_0 = U_1$ and $V_0 = V_1$, and so $\overline{T} = \langle (U_0, V_0), (U_0, V_0), (U_2, V_2)\rangle$. As well, from $\overline{T} \leq_b \overline{S}$, we get that $(U_0, V_0) \ominus (U_2, V_2) \subseteq (X, Y) \ominus (X', Y')$. Since $\overline{T} \leq_b \overline{S}$ implies $\overline{S} \leq_b \overline{T}$, this means that $(X, Y) \ominus (X', Y') \subseteq (U_0, V_0) \ominus (U_2, V_2)$.

We will use this later, and so summarize the result here.

(α) (X, Y) and (X', Y') are such that for every $(U_0, V_0) \in SE(P_1)$ and $(U_2, V_2) \in SE(P_2)$ if $(U_0, V_0) \ominus (U_2, V_2) \subseteq (X, Y) \ominus (X', Y')$ then $(X, Y) \ominus (X', Y') \subseteq (U_0, V_0) \ominus (U_2, V_2)$.

We must show for $\overline{S}' = \langle (X', Y'), (X, Y), (X', Y')\rangle$, that $\overline{T} \leq_b \overline{S}'$ implies $\overline{S}' \leq_b \overline{T}$. Let $\overline{T} = \langle (U_0', V_0'), (U_1', V_1'), (U_2', V_2')\rangle$ and assume that $\overline{T} \leq_b \overline{S}'$. Then, by definition of $\leq_b$, we have that $(U_0', V_0') \ominus (U_1', V_1') \subseteq (X', Y') \ominus (X, Y)$. As well, we have that $(U_0', V_0') \ominus (U_2', V_2') \subseteq (X', Y') \ominus (X', Y') = (\emptyset, \emptyset)$. Hence, we must have that $U_0' = U_2'$ and $V_0' = V_2'$. Thus, we can write $\overline{T} = \langle (U_0', V_0'), (U_1', V_1'), (U_0', V_0')\rangle$.

Now, we will have $\overline{S}' \leq_b \overline{T}$ just if $(X', Y') \ominus (X, Y) \subseteq (U_0', V_0') \ominus (U_1', V_1')$ and $(X', Y') \ominus (X', Y') \subseteq (U_0', V_0') \ominus (U_2', V_2')$. The second condition is vacuously true. As for the first condition, we have that $(U_0', V_0') \in SE(P_2)$ and $(U_1', V_1') \in SE(P_1)$. Thus, via (α), we obtain that $(X', Y') \ominus (X, Y) \subseteq (U_1', V_1') \ominus (U_0', V_0')$. We conclude that $\overline{S}' \leq_b \overline{T}$.

This shows that $\overline{S}' \in Min_b(SE(\Psi))$, where $(X', Y') \in SE(P_0)$ and $(X', Y') \in SE(P_2)$. Consequently, $SE(\Delta(\langle P_0, P_1, P_2\rangle) \sqcap P_2)$ is satisfiable.

($IC5'$). Consider $(X, Y) \in SE(\Delta(\langle P_0, \Psi\rangle) \sqcap \Delta(\langle P_0, \Psi'\rangle))$, and so $(X, Y) \in SE(\Delta(\langle P_0, \Psi\rangle))$ and $(X, Y) \in SE(\Delta(\langle P_0, \Psi'\rangle))$. Thus, $(X, Y) \in Min_b(SE(\langle P_0, \Psi\rangle))$ and $(X, Y) \in Min_b(SE(\langle P_0, \Psi'\rangle))$. Hence, there is some $\langle (X, Y), \overline{S}\rangle \in SE(\langle P_0, \Psi\rangle)$

and some $\langle (X,Y), \overline{S}' \rangle \in SE(\langle P_0, \Psi' \rangle)$ such that $\langle (X,Y), \overline{S} \rangle \leq_b \overline{T}$ for every $\overline{T} \in SE(\langle P_0, \Psi \rangle)$ and $\langle (X,Y), \overline{S}' \rangle \leq_b \overline{T}'$ for every $\overline{T}' \in SE(\langle P_0, \Psi' \rangle)$. But this implies that $\langle (X,Y), \overline{S}, \overline{S}' \rangle \leq_b \langle (X,Y), \overline{T}'' \rangle$ for every $\overline{T}'' \in SE(\langle P_0, \Psi, \Psi' \rangle)$. Consequently, $(X,Y) \in SE(\Delta(\langle P_0, \Psi \circ \Psi' \rangle))$.
($IC7'$). If $\Delta(\langle P_0, \Psi \rangle) \sqcap P_1$ is unsatisfiable then the result is immediate.
So, assume that $\Delta(\langle P_0, \Psi \rangle) \sqcap P_1$ is satisfiable, and let $(X,Y) \in SE(\Delta(\langle P_0, \Psi \rangle) \sqcap P_1)$. That is, $(X,Y) \in SE(\Delta(\langle P_0, \Psi \rangle))$ and $(X,Y) \in SE(P_1)$. By definition we have that $(X,Y) \in Min_b(SE(\langle P_0, \Psi \rangle))_0$. Clearly, since $(X,Y) \in SE(P_1)$ we also obtain that $(X,Y) \in Min_b(SE(\langle P_0 \sqcap P_1, \Psi \rangle))_0$, from which we get $(X,Y) \in SE(\Delta(\langle P_0 \sqcap P_1, \Psi \rangle))$.

A.9. Proof of Theorem 4.7

We first prove a helpful lemma.

LEMMA A.2. *Let Ψ be a belief profile. If $\overline{X} \in Min_a(SE(\Psi))$ then for $X_i \in \overline{X}$ we have $\langle X_i, \overline{X} \rangle \in Min_b(SE(\langle \emptyset, \Psi \rangle))$.*

PROOF. Let Ψ be a belief profile, and let $\overline{X} \in Min_a(SE(\Psi))$. Hence, for every $\overline{Y} \in SE(\Psi)$ we have that $\overline{Y} \leq_a \overline{X}$ implies $\overline{X} \leq_a \overline{Y}$. Now, $\overline{Y} \leq_a \overline{X}$ means that $Y_i \ominus Y_j \subseteq X_i \ominus X_j$ for every $1 \leq i,j \leq |\Psi|$.
So, for fixed i we have that $Y_i \ominus Y_j \subseteq X_i \ominus X_j$ implies that $X_i \ominus X_j \subseteq Y_i \ominus Y_j$. Let $X_0 = X_i$ for that i. Thus, substituting we get that $Y_i \ominus Y_j \subseteq X_0 \ominus X_j$ implies that $X_0 \ominus X_j \subseteq Y_i \ominus Y_j$.
But this means that $\langle X_0, X \rangle \in Min_b(SE(\langle \emptyset, \Psi \rangle))$. □

For the proof of the theorem, we have:
Let $X \in SE(\nabla(\Psi))$. Then, $X \in \bigcup Min_a(SE(\Psi))$; that is, there is some $\overline{X}$ such that $X \in \overline{X}$ and $\overline{X} \in Min_a(SE(\Psi))$. But by Lemma A.2 we then have that $\langle X, \overline{X} \rangle \in Min_b(SE(\langle \emptyset, \Psi \rangle))$. Hence, $X \in Min_b(SE(\langle \emptyset, \Psi \rangle)_0)$ and so $X \in SE(\Delta(\langle \emptyset, \Psi \rangle))$.

A.10. Proof of Theorem 4.8

PROOF. Let $\langle P_1, P_2 \rangle$ be a belief profile. If $\langle P_1, P_2 \rangle$ is unsatisfiable, then both parts of the theorem follow immediately. Hence, assume in the following that $\langle P_1, P_2 \rangle$ is satisfiable.

(1) By definition,

$$SE(\nabla(\langle P_1, P_2 \rangle)) = \{(X,Y) \mid Y \in \cup Min_a(Mod(\langle P_1, P_2 \rangle)), X \subseteq Y, \text{ and if } X \subset Y \text{ then } (X,Y) \in \cup Min_a(SE(\langle P_1, P_2 \rangle))\}.$$

Define

$$f(P,Q) = \{(X,Y) \mid Y \in Min_a(Mod(\langle P, Q \rangle)), X \subseteq Y, \text{ and if } X \subset Y \text{ then } (X,Y) \in Min_a(SE(\langle P, Q \rangle))\}.$$

Then,

$$\begin{aligned} f(P,Q) &= \{(X,Y) \mid Y \in \sigma(Mod(Q), Mod(P)), X \subseteq Y, \text{ and if } X \subset Y \text{ then } (X,Y) \in \sigma(SE(Q), SE(P))\} \\ &= SE(P * Q). \end{aligned}$$

So,

$$SE(\nabla(\langle P_1, P_2 \rangle)) = f(P_1, P_2) \cup f(P_2, P_1).$$

Hence,

$$f(P_1, P_2) \cup f(P_2, P_1) = SE(P_1 * P_2) \cup SE(P_2 * P_1),$$

and so $\nabla(\langle P_1, P_2\rangle) = (P_1 * P_2) \sqcup (P_2 * P_1)$.

(2) By definition we have that

$$SE(\Delta(\langle P_1, P_2\rangle)) = \{(X,Y) \mid Y \in Min_b(Mod(\langle P_1, P_2\rangle))_0, X \subseteq Y, \\ \text{and if } X \subset Y \text{ then } (X,Y) \in Min_b(SE(\langle P_1, P_2\rangle))_0\},$$

From the definitions of Min_b and σ we have that $Y \in Min_b(Mod(\langle P_1, P_2\rangle))_0$ just if $Y \in \sigma(Mod(P_1), Mod(P_2))$ and $(X,Y) \in Min_b(SE(\langle P_1, P_2\rangle))_0$ just if $(X,Y) \in \sigma(SE(P_1), SE(P_2))$.

Thus we get that

$$\begin{aligned} SE(\Delta(\langle P_1, P_2\rangle)) &= \{(X,Y) \mid Y \in \sigma(Mod(P_1), Mod(P_2)), X \subseteq Y, \\ &\qquad \text{and if } X \subset Y \text{ then } (X,Y) \in \sigma(SE(P_1), SE(P_2))\} \\ &= SE(P_2 * P_1). \end{aligned}$$

□

A.11. Proof of Lemma 5.4

Observe that Rules (8) and (9) from Definition 5.3 are applied to any forms of models (i.e., h, t, and c) while (10) and (11) are only applied to t and c. Rules (12) and (13) finally take care of the fact that the first argument of an SE model has to be a model of the reduct. Therefore, we check whether the model given by the t-guess already eliminates rules. Note that such rules are satisfied by the h-guess in a trivial way. Rule (14) finally ensures that all rules of all programs are satisfied by our guesses.

A.12. Proof of Lemma 5.5

The proof follows from a direct argument from the construction of $P[\Psi]$ and is omitted.

A.13. Proof of Lemma 5.7

Omitted.

A.14. Proof of Theorem 5.8

Recall that

$$SE(P *_c \boldsymbol{Q}) = \{(X,Y) \mid Y \in \sigma_{||}(Mod(\boldsymbol{Q}), Mod(P)), X \subseteq Y, \\ \text{and if } X \subset Y \text{ then } \ (X,Y) \in \sigma_{||}(SE(\boldsymbol{Q}), SE(P))\}.$$

where

$$\sigma_{||}(E_1, E_2) = \{A \in E_1 \mid \exists B \in E_2 \ \text{ such that} \\ \forall A' \in E_1, \forall B' \in E_2, |A' \ominus B'| \not< |A \ominus B|\}.$$

By Lemma 5.4, we know that for each pair M_1, M_2 it holds that $M_1 \in Mod(P_1)$ and $M_2 \in Mod(P_2)$ iff $\langle M_1, M_2\rangle \in \{\langle \pi^1_{Mod}(S), \pi^2_{Mod}(S)\rangle \mid S \in AS(P[\Psi])\}$; and likewise, for each pair S_1, S_2 it holds that $S_1 \in SE(P_1)$ and $S_2 \in SE(P_2)$ iff $\langle S_1, S_2\rangle \in \{\langle \pi^1_{SE}(S), \pi^2_{SE}(S)\rangle \mid S \in AS(P[\Psi])\}$.

By Lemma 5.5, we have, for each such pairs characterized by an answer set S, a direct handle to the sets $M_1 \ominus M_2$ and $S_1 \ominus S_2$, respectively, that is, $\pi^1_{Mod}(S) \ominus \pi^2_{Mod}(S) = \{a \mid diff(1,2,a,c) \in S\}$ and $\pi^1_{SE}(S) \ominus \pi^2_{SE}(S) = \big(\{a \mid diff(1,2,a,h) \in S\}, \{b \mid diff(1,2,b,t) \in S\}\big)$.

Now, $\#minimize[diff(1,2,A,c)@1, diff(1,2,A,t)@1]$ exactly selects those answer sets S such that the characterized models (viz., $\pi^1_{Mod}(S)$ and $\pi^2_{Mod}(S)$) possess a minimal cardinality difference among all such pairs.[19] In other words, we obtain

$$\sigma_{||}(Mod(P_2), Mod(P_1)) = \\ \sigma_{||}(\{\pi^2_{Mod}(S) \mid S \in AS(P[\Psi])\}, \{\pi^1_{Mod}(S) \mid S \in AS(P[\Psi])\}),$$

since the latter set equals $\{\pi^2_{Mod}(S) \mid S \in Opt(P[\Psi])\}$, where

$$Opt(P[\Psi]) = \{ S \in AS(P[\Psi]) \mid \forall S' \in AS(P[\Psi])) \\ |\pi^2_{Mod}(S) \ominus \pi^1_{Mod}(S)| \not< |\pi^2_{Mod}(S') \ominus \pi^1_{Mod}(S')|\}.$$

Similarly, we minimize SE models with lower priority via $\#minimize[diff(1,2,A,h)@2]$ thus following the two-phased definition of $SE(P *_c Q)$. Finally, Lemma 5.7 collects the atoms characterising $SE(P *_c Q)$ into the designated result atoms.

A.15. Proofs of Theorems 5.9, 5.10, and 5.12

The proofs of these results proceed similarly to the one for Theorem 5.8.

A.16. Proof of Theorem 5.13

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

LEMMA A.3. *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 $Z' \in Mod(P)$, $Y' \ominus Z' \not\subset Y \ominus Z$. We show that $x \in Z$ holds. Suppose this is not the case: Then, we have $x \in Y \ominus Z$, since $x \in Y$. Now, since $x \notin var(P)$, also $Z \cup \{x\} \in Mod(P)$. But then $x \notin Y \ominus (Z \cup \{x\})$ which yields $Y \ominus (Z \cup \{x\}) \subset Y \ominus Z$, a contradiction to our assumption. Hence, we can suppose $x \in Z$. Now, since $Y \in Mod(Q)$, obviously $Y \setminus \{x\} \in Mod(Q)$ as well. We obtain $Y \ominus Z = (Y \setminus \{x\}) \ominus (Z \setminus \{x\})$, thus $Y \setminus \{x\} \in \sigma(Mod(Q), Mod(P))$ holds. □

LEMMA A.4. *Let P, Q be programs, (X, Y) an SE interpretation, and $x \in Y \setminus var(P \cup Q)$. Then, $(X, Y) \in \sigma(SE(Q), SE(P))$ implies $(X \setminus \{x\}, Y \setminus \{x\}) \in \sigma(SE(Q), SE(P))$.*

PROOF. Since $(X, Y) \in \sigma(SE(Q), SE(P))$, $(X, Y) \in SE(Q)$ and there exists some $(U, Z) \in SE(P)$ such that for each $(X', Y') \in SE(Q)$ and each $(U', Z') \in SE(P)$, $(X', Y') \ominus (U', Z') \not\subset (X, Y) \ominus (U, Z)$. We show that the following relations hold: (i) $x \in Z$; and (ii) $x \in U$ iff $x \in X$. Towards a contradiction, first suppose $x \notin Z$. Then, we have $x \in Y \ominus Z$, 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 $x \notin Y \ominus (Z \cup \{x\})$ which yields $Y \ominus (Z \cup \{x\}) \subset Y \ominus Z$. Thus, $(X, Y) \ominus (U, Z \cup \{x\}) \subset (X, Y) \ominus (U, Z)$, which is a contradiction to the assumption. Hence, $x \in Z$ holds. If (ii) does not hold, we get $x \in X \ominus U$. Now, in case $x \in X$ and $x \notin U$, we have $(X, Y) \ominus (U \cup \{x\}, Z) \subset (X, Y) \ominus (U, Z)$. In case $x \in U$ and $x \notin X$, we have $(X, Y) \ominus (U \setminus \{x\}, Z) \subset (X, Y) \ominus (U, Z)$. 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 $(X, Y) \ominus (U, Z) = (X \setminus \{x\}, Y \setminus \{x\}) \ominus (U \setminus \{x\}, Z \setminus \{x\})$. $(X \setminus \{x\}, Y \setminus \{x\}) \in \sigma(SE(Q), SE(P))$ thus follows. □

[19] The $diff(1,2,A,t)@1$ directive is present for technical matters. Roughly speaking, each answer set S contains a classical model and an SE model of the two programs; since a classical model T corresponds to an SE model (T, T), a parallel minimization is required.

LEMMA A.5. *For any programs P, Q, and R, $P * Q \not\models_s R$ 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, assume $P * Q \not\models_s R$. Then, 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 $(X \cap V, Y \cap V) \in SE(P * Q)$. By definition, $(X,Y) \in SE(Q)$. If $SE(P) = \emptyset$, $SE(P * Q) = SE(Q)$, and since $(X,Y) \in SE(Q)$ obviously implies $(X \cap V, Y \cap V) \in SE(Q)$. $(X \cap V, Y \cap V) \in SE(P * Q)$ thus follows in this case. So suppose $SE(P) \neq \emptyset$. Then, $Y \in \sigma(Mod(Q), Mod(P))$. By iteratively applying Lemma A.3, we obtain that also $Y \cap V \in \sigma(Mod(Q), Mod(P))$. Analogously using Lemma A.4, $(X,Y) \in \sigma(SE(Q), SE(P))$ yields $(X \cap V, Y \cap V) \in \sigma(SE(Q), SE(P))$. By Definition 3.4, we get $(X \cap V, Y \cap V) \in SE(P * Q)$. Finally, it is clear that $(X,Y) \notin SE(R)$, implies that $(X \cap V, Y \cap V) \notin SE(R)$. □

We now proceed with the proof of Theorem 5.13.

We first show membership in Σ_2^P for the complementary problem. From Lemma A.5, 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 $W' \in Mod(P)$, $Y' \ominus W' \not\subset Y \ominus W$. Now, if $Y \subseteq V$, then there is also a $W \subseteq V$ satisfying above test (this is seen by the arguments used in the proof of Lemma A.3). A similar observation holds for $(X,Y) \in \sigma(SE(Q), SE(P))$.

Thus, an algorithm to decide $P * Q \not\models_s R$ 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 whether $(U,Z) \in SE(P)$ and $W \in Mod(P)$. Then, (i) given Y and W, we check whether for each $Y' \subseteq V$ and each $W' \subseteq V$ such that $Y' \in Mod(Q)$ and $W' \in Mod(P)$, $Y' \ominus W' \not\subset Y \ominus W$ 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, (ii) 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 $(X',Y') \in SE(Q)$ and $(U',W') \in SE(P)$, $(X',Y') \ominus (U',W') \not\subset (X,Y) \ominus (U,W)$. 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 3.4 which together with $(X,Y) \notin SE(R)$ solves the complementary problem in view of Lemma A.5.

We recall that model checking as well as SE model checking are in P. So most of the checks used above are in P (expect the already mentioned call to an NP-oracle) and it remains to settle the complexity of the checks (i) and (ii). 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 the problem of checking whether a given quantified Boolean formula of form $\Phi = \forall Y \exists X \varphi$, where φ is a propositional formula in conjunctive normal form over atoms $X \cup Y$, evaluates to true, which is Π_2^P-complete. For Φ as described, let, for each $z \in X \cup Y$, z' be a new atom. Additionally, 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

$z'_1, \ldots, z'_k, z_{k+1}, \ldots, z_m$. Finally, 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 $\{z' \mid z \in Z\}$):

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

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

Only-if direction. Suppose $P_\Phi * Q_\Phi \models_s R_\Phi$ does not hold. By Lemma A.5, 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 $(T, T) \ominus (V, V) \subset (S, T) \ominus (V, V)$, 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 $(U, U) \ominus (V, V) \subset (S, S) \ominus (V, V)$). 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 $w \notin S$, $P_\Phi * Q_\Phi \models_s R_\Phi$ does not hold.

This shows Π^P_2-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 $x \leftarrow \sim x', w$ and $x' \leftarrow \sim x, w$ by $x; x' \leftarrow w$. Due to the presence of the constraints $\bot \leftarrow y, y'$ and $\bot \leftarrow x, x'$, this modification does not change the SE models of these programs.

A.17. Proof of Theorem 5.14

By Theorem 5.8, the SE models of $P *_c Q$ are determined by a disjunction-free logic program making use of minimize statements. To encode the decision problem $P *_c Q \models_s R$ one just has to slightly extend the encoding in such a way that, given R, the answer sets of the extended encoding characterizes those SE models of $P *_c Q$ which are *not* SE-models of R. If no such answer set remains, the problem $P *_c Q \models_s R$ holds. Moreover, by methods described by Simons et al. [2002], the encoding can be written in such a way

that only a single minimize statement is used. Hence, we have a program S containing one minimize statement such that $P *_c Q \models_s R$ holds iff S has no answer set. Now, since checking whether a program with a single minimize statement has an answer set is in Θ_2^P [Buccafurri et al. 2000], and since co-$\Theta_2^P = \Theta_2^P$, the result follows.

A.18. Proof of Theorem 5.15

By Theorem 4.8, it can be shown that the Π_2^P-hardness result for the revision problem also applies to the respective problems in terms of merging. On the other hand, Π_2^P-membership can be obtained by a slight extension of the encodings for merging given in Section 5.3 such that these extensions possess an answer set iff the respective decision problem of checking whether $\nabla(\Psi) \models_s R$ or $\Delta(\Psi) \models_s R$, for a given profile Ψ and program R, does *not* hold. Since checking whether a program has at least one answer set is Σ_2^P-complete, and our (extended) encodings are polynomial in the size of the encoded problems, the desired membership results follow.

REFERENCES

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

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

Alferes, J. J., Banti, F., Brogi, A., and Leite, J. A. 2005. The refined extension principle for semantics of dynamic logic programming. *Studia Logica 79*, 1, 7–32.

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

Baral, C., Kraus, S., and Minker, J. 1991. Combining multiple knowledge bases. *IEEE Trans. Knowl. Data Eng. 3*, 2, 208–220.

Baral, C., Kraus, S., Minker, J., and Subrahmanian, V. 1992. Combining multiple knowledge bases consisting of first order theories. *Computational Intell. 8*, 1, 45–71.

Benferhat, S., Dubois, D., Kaci, S., and Prade, H. 2003. Possibilistic merging and distance-based fusion of propositional information. *Ann. Math. Artif. Intell. 34*, 1–3, 217–252.

Booth, R. 2002. Social contraction and belief negotiation. In *Proceedings of the 8th International Conference on the Principles of Knowledge Representation and Reasoning*. D. Fensel, F. Giunchiglia, D. McGuiness, and M. Williams Eds., Morgan Kaufmann, San Francisco, 375–384.

Buccafurri, F. and Gottlob, G. 2002. Multiagent compromises, joint fixpoints, and stable models. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski*, Part I., Springer, 561–585.

Buccafurri, F., Leone, N., and Rullo, P. 2000. Enhancing disjunctive datalog by constraints. *IEEE Trans. Knowl. Data Eng. 12*, 5, 845–860.

Cabalar, P. and Ferraris, P. 2007. Propositional theories are strongly equivalent to logic programs. *Theory Pract. Logic Program. 7*, 6, 745–759.

Dalal, M. 1988. Investigations into theory of knowledge base revision. In *Proceedings of the AAAI National Conference on Artificial Intelligence*. 449–479.

Dantsin, E., Eiter, T., Gottlob, G., and Voronkov, A. 2001. Complexity and expressive power of logic programming. *ACM Comput. Surv. 33*, 3, 374–425.

Delgrande, J., Schaub, T., and Tompits, H. 2003. A framework for compiling preferences in logic programs. *Theory Pract. Logic Program. 3*, 2, 129–187.

Delgrande, J., Schaub, T., and Tompits, H. 2007. A preference-based framework for updating logic programs. In *Proceedings of the 9th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR'07)*. C. Baral, G. Brewka, and J. Schlipf Eds., Lecture Notes in Artificial Intelligence, vol. 4483, Springer-Verlag, 71–83.

Delgrande, J., Schaub, T., Tompits, H., and Woltran, S. 2008. Belief revision of logic programs under answer set semantics. In *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR'08)*. G. Brewka and J. Lang Eds., AAAI Press, 411–421.

Delgrande, J., Schaub, T., Tompits, H., and Woltran, S. 2009. Merging logic programs under answer set semantics. In *Proceedings of the 25th International Conference on Logic Programming (ICLP'09)*. P. Hill and D. Warren Eds., Lecture Notes in Computer Science, vol. 5649. Springer, 160–174.

Eiter, T. and Gottlob, G. 1992. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artif. Intell. 57*, 2–3, 227–270.

Eiter, T. and Wang, K. 2008. Forgetting in answer set programming. *Artif. Intell. 172*, 14, 1644–1672.

Eiter, T., Fink, M., Sabbatini, G., and Tompits, H. 2002. On properties of update sequences based on causal rejection. *Theory Pract. Logic Program. 2*, 6, 711–767.

Eiter, T., Faber, W., Leone, N., and Pfeifer, G. 2003. Computing preferred answer sets by meta-interpretation in answer set programming. *Theory Pract. Logic Program. 3*, 4–5, 463–498.

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'05)*. 97–102.

Gabbay, D., Rodrigues, O., and Russo, A. 2008. Belief revision in non-classical logics. *Rev. Symb. Logic 1*, 3, 267–304.

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

Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T., and Thiele, S. A user's guide to Gringo, clasp, Clingo, and iClingo. http://potassco.sourceforge.net.

Gebser, M., Pührer, J., Schaub, T., and Tompits, H. 2008. A meta-programming technique for debugging answerset programs. In *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI'08)*. D. Fox and C. Gomes Eds., AAAI Press, 448–453.

Gebser, M., Kaminski, R., and Schaub, T. 2011. Complex optimization in answer set programming. *Theory Pract. Logic Program. 11*, 4–5, 821–839.

Gelfond, M. and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference and Symposium of Logic Programming (ICLP'88)*. R. Kowalski and K. Bowen Eds., MIT Press, 1070–1080.

Hansson, S. O. 1999. *A Textbook of Belief Dynamics*. Applied Logic Series. Kluwer Academic Publishers.

Inoue, K. and Sakama, C. 1995. Abductive framework for nonomonotonic theory change. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI'95)*. Morgan Kaufmann, 204–210.

Inoue, K. and Sakama, C. 1998. Negation as failure in the head. *J. Logic Program. 35*, 1, 39–78.

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

Konieczny, S. and Pino Pérez, R. 2002. Merging information under constraints: A logical framework. *J. Logic Comput. 12*, 5, 773–808.

Konieczny, S., Lang, J., and Marquis, P. 2002. Distance-based merging: A general framework and some complexity results. In *Proceedings of the 8th International Conference on the Principles of Knowledge Representation and Reasoning*. D. Fensel, F. Giunchiglia, D. McGuiness, and M. Williams Eds., Morgan Kaufmann, San Francisco, 97–108.

Kudo, Y. and Murai, T. 2004. A method of belief base revision for extended logic programs based on state transition diagrams. In *Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES'04)*, Part I. M. G. Negoita, R. J. Howlett, and L. C. Jain Eds., Lecture Notes in Computer Science, vol. 3213, Springer, 1079–1084.

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

Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., and Scarcello, F. 2006. The DLV system for knowledge representation and reasoning. *ACM Trans. Comput. Logic 7*, 3, 499–562.

Liberatore, P. and Schaerf, M. 1998. Arbitration (or how to merge knowledge bases). *IEEE Trans. Knowl. Data Eng. 10*, 1, 76–90.

Lifschitz, V. and Woo, T. 1992. Answer sets in general nonmonotonic reasoning (preliminary report). In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*. B. Nebel, C. Rich, and W. Swartout Eds., Morgan Kaufmann Publishers, 603–614.

Lifschitz, V., Pearce, D., and Valverde, A. 2001. Strongly equivalent logic programs. *ACM Trans. Comput. Logic 2*, 4, 526–541.

Lin, J. and Mendelzon, A. 1999. Knowledge base merging by majority. In *Dynamic Worlds: From the Frame Problem to Knowledge Management*, R. Pareschi and B. Fronhöfer Eds., Kluwer, 195–218.

Marek, V. W. and Truszczyński, M. 1998. Revision programming. *Theor. Comput. Sci. 190*, 241–277.

Meyer, T. 2001. On the semantics of combination operations. *J. Appl. Non-Classical Logics 11*, 1–2, 59–84.

Nelson, D. 1949. Constructible falsity. *J. Symb. Logic 14*, 2, 16–26.

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

Osorio, M. and Zacarías, F. 2004. On updates of logic programs: A properties-based approach. In *Proceedings of the 3rd International Symposium on Foundations of Information and Knowledge Systems (FoIKS'04)*. Lecture Notes in Computer Science, vol. 2942, Springer, 231–241.

Osorio, M. and Zepeda, C. 2003. Towards the use of semantics contents in ASP for planning and diagnostic in GIS. In *Proceedings of the 2nd International Workshop on Answer Set Programming (ASP'03)*. M. D. Vos and A. Provetti Eds., CEUR Workshop Proceedings, vol. 78, CEUR-WS.org.

Przymusinski, T. and Turner, H. 1997. Update by means of inference rules. *J. Logic Program. 30*, 2, 125–143.

Revesz, P. 1993. On the semantics of theory change: Arbitration between old and new information. In *Proceedings of the 12th ACM Symposium on Principles of Database Systems*. C. Beeri Ed., 71–82.

Sakama, C. and Inoue, K. 2003. An abductive framework for computing knowledge base updates. *Theory Pract. Logic Program. 3*, 6, 671–713.

Sakama, C. and Inoue, K. 2006. Combining answer sets of nonmonotonic logic programs. In *Proceedings of the 6th International Workshop on Computational Logic in Multi-Agent Systems*. Lecture Notes in Computer Science, vol. 3900, Springer, 320–339.

Sakama, C. and Inoue, K. 2007. Constructing consensus logic programs. In *Proceedings of the 16th International Symposium on Logic-Based Program Synthesis and Transformation (Lopstr'06)*. Revised Selected Papers, G. Puebla Ed., Lecture Notes in Computer Science, vol. 4407, Springer, 26–42.

Sakama, C. and Inoue, K. 2008. Coordination in answer set programming. *ACM Trans. Comput. Logic 9*, 2, 1–30.

Satoh, K. 1988. Nonmonotonic reasoning by minimal belief revision. In *Proceedings of the International Conference on 5th Generation Computer Systems*. 455–462.

Simons, P., Niemelä, I., and Soininen, T. 2002. Extending and implementing the stable model semantics. *Artif. Intell. 138*, 1–2, 181–234.

Slota, M. and Leite, J. 2010. On semantic update operators for answer-set programs. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI'10)*. H. Coelho, R. Studer, and M. Wooldridge Eds., IOS Press, 957–962.

Spohn, W. 1988. Ordinal conditional functions: A dynamic theory of epistemic states. In *Causation in Decision, Belief Change, and Statistics*, W. Harper and B. Skyrms Eds., Vol. II, Kluwer Academic Publishers, 105–134.

Turner, H. 2003. Strong equivalence made easy: Nested expressions and weight constraints. *Theory Pract. Logic Program. 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 *Proceedings of the 5th European Workshop on Logics in Artificial Intelligence (JELIA'94)*. Lecture Notes in Artificial Intelligence, vol. 838, Springer, 137–151.

Zacarías, F., Osorio, M., Acosta Guadarrama, J. C., and Dix, J. 2005. Updates in Answer Set Programming based on structural properties. In *Proceedings of the 7th International Symposium on Logical Formalizations of Commonsense Reasoning*. S. McIlraith, P. Peppas, and M. Thielscher Eds., Fakultät für Informatik, ISSN 1430-211X, 213–219.

Zhang, Y. and Foo, N. 1997. Towards generalized rule-based updates. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI'97)*. Vol. 1, Morgan Kaufmann, 82–88.

Zhang, Y. and Foo, N. 2006. Solving logic program conflict through strong and weak forgetting. *Artif. Intell. 170*, 739–778.

Zhang, Y. and Foo, N. Y. 1998. Updating logic programs. In *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI'98)*. 403–407.

Received December 2009; revised August 2011; accepted March 2012