Merging Logic Programs under Answer Set Semantics

James Delgrande[1], Torsten Schaub[2], Hans Tompits[3], and Stefan Woltran[3]

[1] Simon Fraser University, Burnaby, B.C., Canada V5A 1S6
[2] Universität Potsdam, August-Bebel-Str. 89, D–14482 Potsdam, Germany
[3] Technische Universität Wien, Favoritenstraße 9–11, A–1040 Vienna, Austria

Abstract. This paper considers a semantic approach for merging logic programs under answer set semantics. Given logic programs $P_1, \ldots, P_n$, the goal is to provide characterisations of the merging of these programs. Our formal techniques are based on notions of relative distance between the underlying SE models of the logic programs. Two approaches are examined. The first informally selects those models of the programs that vary the least from the 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 $P_1, \ldots, P_n$. P_0 can be thought of as analogous to a set of database integrity constraints. We examine formal properties of these operators. Moreover, we give encodings for computing the mergings of a multiset of logic programs, within the same logic programming framework. As a by-product, we provide a complexity analysis revealing that our operators do not increase the complexity of the base formalism.

1 Introduction

Answer set programming [1] is an appealing approach for representing problems in knowledge representation and reasoning: It has a conceptually simple theoretical foundation, while at the same time it has found application in a wide range of practical problems. As well, there are now efficient and well-studied implementations. 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, or in some other fashion modifying the knowledge represented in the program.

In the past, research on the evolution of logic programs mostly focussed on *updating logic programs* [2–7]. In such approaches, the issue was to characterise the answer sets of a sequence of programs $\langle P_1, \ldots, P_n \rangle$, where for $j > i$, program P_j has higher priority, in some sense, over P_i. However, seemingly the nonmonotonic nature of extended logic programs makes the problem of belief change intrinsically harder compared to a monotonic setting, often leading to subtle effects, which hampered research in this direction. In previous work [8], we addressed this challenge by defining an approach for revising logic programs under answer set semantics based on the notion of an *SE model* [9]. The key point of this undertaking is that SE models provide a *monotonic semantic foundation* of answer set programs. More specifically, SE models derive from models in the logic of here-and-there, which is intermediate between classical logic and intuitionistic logic, representing the logical underpinning of strong equivalence [10]. Indeed, the latter notion can be seen as the logic programming analogue of

ordinary equivalence in classical logic, in the sense that both equivalence notions adhere to a substitution principle. With our revision approach for logic programs based on SE models we thus phrased the problem of belief revision in logic programs in terms analogous to those of revision in classical logic. Additionally, the approach possesses appealing features as it adheres to all but one of the established postulates for belief revision [11].

In this paper, we employ these techniques to address the *merging* of logic programs. The problem of merging multiple, potentially conflicting bodies of information arises in 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 get a consistent set of merged beliefs. Curiously, the problem of merging logic programs has, as far as we are aware, not previously been addressed, perhaps as a consequence of the lack of agreement concerning the revising and updating nonmonotonic logic programs.

In characterising the merging of logic programs, the central idea is that the SE models of the merged program are those that are in some sense "closest" to the SE models of the programs to be merged. However, as with merging knowledge bases expressed in classical logic, there is no one 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 *arbitration* [12]. Basically (SE) models are selected from among the SE models of the programs to be merged; in a sense this operator is a natural extension of our belief revision operator, presented in [8]. The second merging operator can be regarded as an instance of Konieczny and Pino Pérez's *merging* operator [13]. 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.

2 Background

2.1 Answer Set Programming

A (*generalised*) *logic program*[4] (GLP) over an alphabet $\mathcal{A}$ is a finite set of rules of the form

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

where $a_i, b_j, c_k, d_l \in \mathcal{A}$ are *atoms*, for $1 \leq i \leq m \leq j \leq n \leq k \leq o \leq l \leq p$. Operators ';' and ',' express disjunctive and conjunctive connectives. A *default literal* is an atom a or its (default) negation ${\sim}a$. A rule r as in (1) is called a *fact* if $p = 1$, *normal* if $n = 1$, *disjunctive* if $m = n$, and an *integrity constraint* if $n = 0$, yielding

[4] Such programs were first considered by Lifschitz and Woo [14] and coined *generalised disjunctive logic programs* by Inoue and Sakama [15].

an empty disjunction denoted by $\bot$. Accordingly, a program is called *disjunctive*, or a DLP, if it consists of disjunctive rules only. Likewise, a program is *normal* if it contains normal rules only. We furthermore define $H(r) = \{a_1, \ldots, a_m, \sim b_{m+1}, \ldots, \sim b_n\}$ as the *head* of r and $B(r) = \{c_{n+1}, \ldots, c_o, \sim d_{o+1}, \ldots, \sim d_p\}$ as the *body* of r, for r as in (1). Moreover, given a set X of literals, $X^+ = \{a \in \mathcal{A} \mid a \in X\}$, $X^- = \{a \in \mathcal{A} \mid \sim a \in X\}$, and $\sim X = \{\sim a \mid a \in X \cap \mathcal{A}\}$. For simplicity, we sometimes use a set-based notation, expressing r as in (1) as $H(r)^+; \sim H(r)^- \leftarrow B(r)^+, \sim B(r)^-$.

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

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

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

As defined by Turner [9], an *SE interpretation* is a pair (X, Y) of interpretations such that $X \subseteq Y \subseteq \mathcal{A}$. An SE interpretation is an *SE model* of a program P if $Y \models P$ and $X \models P^Y$. The set of all SE models of a program P is denoted by $SE(P)$. Note that Y is an answer set of P iff $(Y, Y) \in SE(P)$ and no $(X, Y) \in SE(P)$ with $X \subset Y$ exists. Also, we have $(Y, Y) \in SE(P)$ iff $Y \in Mod(P)$.

A program P is *satisfiable* just if $SE(P) \neq \emptyset$. Two programs P and Q are *strongly equivalent*, symbolically $P \equiv_s Q$, iff $SE(P) = SE(Q)$. Alternatively, $P \equiv_s Q$ holds iff $AS(P \cup R) = AS(Q \cup R)$, for every program R [10]. 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, e.g., (a, ab) instead of $(\{a\}, \{a, b\})$.

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

We have the following properties: (i) for each GLP P, $SE(P)$ is well-defined; (ii) for each DLP P, $SE(P)$ is complete. Furthermore, for each well-defined set S of SE interpretations, there exists a GLP P such that $SE(P) = S$, and for each complete set S of SE interpretations, there exists a DLP P such that $SE(P) = S$. Programs meeting these conditions can be constructed thus [16, 17]: 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 $\mathcal{A} = \{p,q\}$.[5] Note that S is not complete. The canonical GLP is as follows:

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

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

$$r_\emptyset : \bot \leftarrow \sim p, \sim q; \qquad r_{\emptyset,q} : q \leftarrow \sim p.$$

One feature of SE models is that they contain "more information" than answer sets, which makes them an appealing candidate for problems where programs are examined with respect to further extension (in fact, this is what strong equivalence is about). We illustrate this point with the following well-known example, involving programs $P = \{p; q \leftarrow\}$ and $Q = \{p \leftarrow \sim q,\ q \leftarrow \sim p\}$. Here, we have $AS(P) = AS(Q) = \{\{p\},\{q\}\}$. However, the SE models (we list them for $\mathcal{A} = \{p,q\}$) differ:

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

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.

2.2 Belief Merging

This section reviews previous work in belief merging. We focus on two approaches that arguably cover the most intuitive means (in a semantic sense) of merging beliefs. We first briefly survey representative related work in the belief merging literature.

Earlier work on merging operators includes approaches by Baral et al. [18] and Revesz [19]. 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 (see below) that, intuitively selects from among the models of the belief sets being merged. Lin and Mendelzon [20] 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 [12] address arbitration in general, while Konieczny and Pino Pérez [13] considers 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, Lang, and Marquis [21] describe a very general framework in which a family of merging operators is parameterised by a distance between interpretations and aggregating functions. More or less concurrently, Meyer [22] proposed a general approach to formulating merging functions, based on ordinal conditional functions [23].

[5] We assume henceforth that the alphabet in an example consists of just the mentioned atoms.

Roughly, epistemic states are associated with a mapping from possible worlds onto the set of ordinal numbers. Various merging operators then can be defined by considering the ways in which two epistemic states can be resolved into a single ordinal conditional function. Booth [24] also considers the problem of an agent merging information from different sources, via what is called *social contraction*. In a manner analogous to the Levi Identity for belief revision, information from the various sources is weakened to the extent that it can be consistently added to the agent's belief base. Last, much work has been carried out in merging possibilistic knowledge bases; see for example [25].

We next describe the approaches by Liberatore and Schaerf [12] and by Konieczny and Pino Pérez [13], since we use the intuitions underlying these approaches as the basis for our approaches for merging logic programs. First, Liberatore and Schaerf [12] consider merging two belief bases based 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 an *arbitration* operator (Konieczny and Pino Pérez [13] call it a *commutative revision operator*). They consider a propositional languages over a finite set of atoms; consequently their merging operator can be expressed as a binary operator on formulas. The following postulates characterise this operator:

Definition 1. $\diamond$ *is an* arbitration operator (*or* commutative revision operator) *if* $\diamond$ *satisfies the following postulates.*

$(LS1)$ $\vdash \alpha \diamond \beta \equiv \beta \diamond \alpha$.
$(LS2)$ $\vdash \alpha \wedge \beta \supset \alpha \diamond \beta$.
$(LS3)$ *If* $\alpha \wedge \beta$ *is satisfiable then* $\vdash \alpha \diamond \beta \supset \alpha \wedge \beta$.
$(LS4)$ $\alpha \diamond \beta$ *is unsatisfiable iff* α *is unsatisfiable and* β *is unsatisfiable.*
$(LS5)$ *If* $\vdash \alpha_1 \equiv \alpha_2$ *and* $\vdash \beta_1 \equiv \beta_2$ *then* $\vdash \alpha_1 \diamond \beta_1 \equiv \alpha_2 \diamond \beta_2$.
$(LS6)$ $\alpha \diamond (\beta_1 \vee \beta_2) = \begin{cases} \alpha \diamond \beta_1 & \textit{or} \\ \alpha \diamond \beta_2 & \textit{or} \\ (\alpha \diamond \beta_1) \vee (\alpha \diamond \beta_2). \end{cases}$
$(LS7)$ $\vdash (\alpha \diamond \beta) \supset (\alpha \vee \beta)$.
$(LS8)$ *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 Peréz [13] 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. Multiset union is denoted by $\cup$. Following Konieczny and Pino Peréz [13], let $\Delta^{\mu}(\Psi)$ denote the result of merging the multi-set 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:

Definition 2 ([13]). *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* $\mu \nvdash \bot$ *then* $\Delta^{\mu}(\Psi) \nvdash \bot$.
$(IC2)$ *If* $\bigwedge \Psi \nvdash \neg\mu$ *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:* $\Delta^{\mu}(\phi \cup \phi') \wedge \phi \nvdash \bot$ *implies* $\Delta^{\mu}(\phi \cup \phi') \wedge \phi' \nvdash \bot$.
$(IC5)$ $\Delta^{\mu}(\Psi_1) \wedge \Delta^{\mu}(\Psi_2) \vdash \Delta^{\mu}(\Psi_1 \cup \Psi_2)$.
$(IC6)$ *If* $\Delta^{\mu}(\Psi_1) \wedge \Delta^{\mu}(\Psi_2) \nvdash \bot$ *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* $\Delta^{\mu_1}(\Psi) \wedge \mu_2 \nvdash \bot$ *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. $(IC4)$ is a *fairness* postulate, that when two belief bases disagree, merging doesn't 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, but with respect to the integrity constraints.

3 Merging Logic Programs

We denote (generalised) logic programs by P_1, P_2, ..., reserving P_0 for a program representing global constraints, as described later. For logic programs P_1, 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 $\langle P_1, \ldots, P_n \rangle$ of (generalised) 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, 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 $\overline{S}_i$ to denote the i^{th} component S_i of $\overline{S}$. Thus, we have $\overline{S}_i \in SE(P_i)$. 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 $\overline{X}_i$ to denote the i^{th} component of $\overline{X} \in Mod(\Psi)$.

Let $\ominus$ denote the symmetric difference operator between sets, i.e., $X \ominus Y = (X \setminus Y) \cup (Y \setminus X)$ for every set X, Y. We extend $\ominus$ so that it can be used with SE interpretations as follows: 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, $(X_1, X_2) \subseteq (Y_1, Y_2)$ iff $X_1 \subseteq Y_1$ and $X_2 \subseteq Y_2$, and, moreover, $(X_1, X_2) \subset (Y_1, Y_2)$ iff $(X_1, X_2) \subseteq (Y_1, Y_2)$ and either $X_1 \subset Y_1$ or $X_2 \subset Y_2$.

3.1 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 $\overline{S}_i = \overline{S}_j$ for every $1 \leq i, j \leq n$; and merging is the same as simply unioning 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 3. *Let $\Psi = \langle P_1, \ldots, P_n \rangle$ be a satisfiable belief profile and let $\overline{S}, \overline{T}$ be two SE models of Ψ (or two classical models of Ψ).*

Then, define $\overline{S} \leq_a \overline{T}$, if $\overline{S}_i \ominus \overline{S}_j \ \subseteq \ \overline{T}_i \ominus \overline{T}_j$ for every $1 \leq i < j \leq n$.

Clearly, $\leq_a$ is a partial pre-order. In what follows, let $Min_a(N)$ denote the set of all minimal elements of a set N of tuples relative to $\leq_a$, i.e.,

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

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

$$\cup N = \{S \mid S = \overline{S}_i \text{ for some } \overline{S} \in N \text{ and some } i \in \{1, \ldots, n\}\}.$$

Definition 4. *Let Ψ be a belief profile. The* arbitration merging*, or simply* arbitration *of Ψ, is a logic program $\nabla(\Psi)$ such that*

$$\begin{aligned} SE(\nabla(\Psi)) = \{(X, Y) \mid\ & Y \in \cup Min_a(Mod(\Psi)), X \subseteq Y, \\ & \textit{and if } X \subset Y \textit{ then } (X, Y) \in \cup Min_a(SE(\Psi))\} \, , \end{aligned}$$

provided $\Psi = \langle P_1, \ldots, P_n \rangle$ 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

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

Given that $SE(P_1) = \{(pu, pu), (pu, puv), (puv, puv)\}$ 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

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

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

Table 1. Examples on Arbitration Merging.

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 1.

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

Moreover, we have the following general result.

Theorem 1. *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 1.*

$(LS1')$ $P_1 \diamond P_2 \equiv_s P_2 \diamond P_1$.
$(LS2')$ $P_1 \sqcap P_2 \models_s P_1 \diamond P_2$.
$(LS3')$ *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 satisfiable iff P_1 is satisfiable and P_2 is satisfiable.*
$(LS5')$ *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 is satisfiable then $P_1 \sqcap (P_1 \diamond P_2)$ is satisfiable.*

3.2 Basic Merging

For the second approach to merging, programs $P_1, \ldots, P_n$ are merged with a target logic program P_0 so that the SE models in the merging will drawn from models of P_0. This operator will be 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 [13] 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 do not coincide, and that merging is generally a weaker operator than arbitration. As well, it can be observed again that if the set of programs is jointly consistent then the merging of the programs corresponds to their union.

Definition 5. *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 $\overline{S}_0 \ominus \overline{S}_j \ \subseteq\ \overline{T}_0 \ominus \overline{T}_j$ for every $1 \leq j \leq n$.

As in the case of arbitration merging, $\leq_b$ is a partial pre-order. 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

$$N_0 = \{\overline{S}_0 \mid \overline{S} \in N\} .$$

We thus can state our central definition for basic merging as follows:

Definition 6. *Let Ψ be a belief profile. The* basic merging*, or simply* merging *of Ψ, is a logic program $\Delta(\Psi)$ such that*

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

provided $\Psi = \langle P_1, \ldots, P_n\rangle$ 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 Program 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 twenty-seven 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 $Min_b(Mod(\langle \emptyset, P_1, P_2\rangle)) = \{\langle uv, puv, uv\rangle, \langle puv, puv, uv\rangle\}$. Accordingly, we get

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

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 1, where every second merging is weakened by the addition of such intermediate SE models. This is made precise in Theorem 3 below. We summarise the results in Table 2 (but omit programs due to limited space). In fact, the programs $\Delta(\langle \emptyset, P_1, P_2\rangle)$ are obtained from $\nabla(\langle P_1, P_2\rangle)$ in Table 1 by simply dropping all rules of form $p; {\sim}p \leftarrow$ and $q; {\sim}q \leftarrow$, respectively.

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

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

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

Table 2. Examples on Basic Merging.

Theorem 2. *Let Ψ be a belief profile, P_0 a program representing global constraints, and Δ as given in Definition 6. Then, Δ satisfies the following versions of the postulates of Definition 2:*

$(IC0')$ $\Delta(\langle P_0, \Psi\rangle) \models_s P_0$.
$(IC1')$ *If P_0 is satisfiable then $\Delta(\langle P_0, \Psi\rangle)$ is satisfiable.*
$(IC2')$ *If If $\sqcap(\Psi)$ 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.
$(IC5')$ $\Delta(\langle P_0, \Psi\rangle) \sqcap \Delta(\langle P_0, \Psi'\rangle) \models_s \Delta(\langle P_0, \Psi \circ \Psi'\rangle)$.
$(IC7')$ $\Delta(\langle P_0, \Psi\rangle) \sqcap P_0' \models_s \Delta(\langle P_0 \sqcap P_0', \Psi\rangle)$.
$(IC9')$ *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 3. *Let Ψ_a and $\Psi_b = \langle\emptyset, \Psi\rangle$ be belief profiles. Then $\nabla(\Psi_a) \models_s \Delta(\Psi_b)$.*

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 previous work [8].

Theorem 4. *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), as according to our convention for basic merging.

4 Computational Issues

In this section, we provide encodings for arbitration and basic merging. Since our encodings can be computed efficiently from a given belief profile, we are able to provide complexity results for decision problems, typically associated to merging operators. We start, however, with the formal machinery required for the encodings.

In what follows, for basic merging we consider the program representing integrity constraints to be part of a belief profile, and conventionally have it as the first element of the belief profile. Thus, we write $\Psi = \langle P_\alpha, \ldots, P_n \rangle$, and depending on the merging operator, we have $\alpha = 0$ or $\alpha = 1$. Moreover, we restrict ourselves to *satisfiable* belief profiles here. In fact, a generalisation of the subsequent encodings to the general case is possible but requires some further technical efforts, which we omit in order to provide a more succinct presentation of the basic ideas.

We let A be the set of all atoms occurring in Ψ and require mutually disjoint atoms

$$\{a_h^i, a_t^i, a_m^i, \hat{a}_h^i, \hat{a}_t^i, \hat{a}_m^i \mid a \in A, \alpha \leq i \leq n\}, \tag{3}$$

which are used as follows: Atoms a_h^i, a_t^i ($1 \leq i \leq n$) characterise $SE(P_i)$, and likewise, a_m^i is used to characterise $Mod(P_i)$. In other words, an assignment to the atoms $\{a_h^i, a_t^i \mid a \in A, 1 \leq i \leq n\}$ represents a candidate for $\overline{S} \in SE(\Psi)$ and an assignment to the atoms $\{a_m^i \mid a \in A, 1 \leq i \leq n\}$ represents a candidate for $\overline{X} \in Mod(\Psi)$. The atoms $\hat{a}_h^i, \hat{a}_t^i, \hat{a}_m^i$ play analogous roles and are used to range over further SE models (resp., classical models) $\overline{T}$ of Ψ. In particular, we compare $\overline{T}$ to $\overline{S}$ (resp., to $\overline{X}$) to make the necessary checks for the merging operators. We give the formal details below.

To "guess" assignments, we need each atom a from the set (3) also in a "negated way", $\tilde{a}$. Moreover, we use further atoms $O = \{a_h^o, a_t^o \mid a \in A\}$ to carry our final result, $SE(\nabla(\Psi))$ (resp., $SE(\Delta(\Psi))$), and atoms H for particular technical programming issues, which we introduce as we go along. For the moment, we just have to assume that our encodings are given over an alphabet A^Ψ which contains each atom from the set (3) and its negation, the output atoms O and further atoms H.

We use sub- and superscripts also as renaming functions: Given a set $Y \subseteq A$ of atoms, $x \in \{h, t, m\}$, and an index i, Y_x^i denotes the set $\{y_x^i \mid y \in Y\}$, $\hat{Y}_x^i$ denotes the set $\{\hat{y}_x^i \mid y \in Y\}$, etc. Likewise for a rule r, r_x^i denotes the rule r after replacing each of its atom y by y_x^i; $\hat{r}_x^i$ denotes the rule r after replacing each atom y by $\hat{y}_x^i$, etc.

We are now able to formally associate an interpretation $I \subseteq A^\Psi$ to several SE and classical interpretations over A as follows: Let $I \subseteq A^\Psi$ and i an index. Then,

$$\begin{aligned}
\sigma^i(I) &= \{(X, Y) \mid X, Y \subseteq A, X_h^i = I \cap A_h^i, Y_t^i = I \cap A_t^i\}, \\
\pi^i(I) &= \{X \mid X \subseteq A, X_m^i = I \cap A_m^i\}.
\end{aligned}$$

Moreover, let $\sigma(I) = \langle \sigma^\alpha(I), \ldots, \sigma^n(I) \rangle$ and $\pi(I) = \langle \pi^\alpha(I), \ldots, \pi^n(I) \rangle$. Likewise, for a set $\mathcal{I}$ of interpretations over A^Ψ, we define $\Sigma^i(\mathcal{I}) = \bigcup_{I \in \mathcal{I}} \sigma^i(I)$, $\Pi^i(\mathcal{I}) = \bigcup_{I \in \mathcal{I}} \pi^i(I)$, $\Sigma(\mathcal{I}) = \Sigma^\alpha(\mathcal{I}) \times \cdots \times \Sigma^n(\mathcal{I})$, and $\Pi(\mathcal{I}) = \Pi^\alpha(\mathcal{I}) \times \cdots \times \Pi^n(\mathcal{I})$.

We define the following module for an index i:

$$\begin{aligned}
G[i] = \{&a_x^i; \tilde{a}_x^i \leftarrow, \quad \bot \leftarrow a_x^i, \tilde{a}_x^i \mid a \in A, x \in \{h, t, m\}\} \cup \\
&\{\bot \leftarrow a_h^i, \tilde{a}_t^i \mid a \in A\} \cup \\
&\{\bot \leftarrow H^+(\tilde{r}_y^i), H^-(r_y^i), B^+(r_y^i), B^-(\tilde{r}_y^i) \mid r \in P_i, y \in \{t, m\}\} \cup \\
&\{\bot \leftarrow H^+(\tilde{r}_h^i), H^-(r_t^i), B^+(r_h^i), B^-(\tilde{r}_t^i) \mid r \in P_i\}.
\end{aligned}$$

We note that $\Sigma^i(AS(G[i])) = SE(P_i)$ and $\Pi^i(AS(G[i])) = Mod(P_i)$). Consequently, $\Sigma(AS(G[\alpha] \cup \cdots \cup G[n])) = SE(\Psi)$ and $\Pi(AS(G[\alpha] \cup \cdots \cup G[n])) = Mod(\Psi)$.

The next module guesses the remaining atoms which are used to check minimality of the guess above. However, we use now a spoiling technique rather than constraints, to exclude (SE) interpretations which are not (SE) models of the respective program. The new atom z indicates whether we have to spoil. Moreover, this spoiling is also activated below where we compare the new guess with the guess from above.

$$\begin{aligned} H[i] = \{&\hat{a}^i_x; \tilde{\hat{a}}^i_x \leftarrow, \quad z \leftarrow \hat{a}^i_x, \tilde{\hat{a}}^i_x, \quad \hat{a}^i_x \leftarrow z, \quad \tilde{\hat{a}}^i_x \leftarrow z \mid a \in A, x \in \{h,t,m\}\} \cup \\ &\{z \leftarrow \hat{a}^i_h, \tilde{\hat{a}}^i_t \mid a \in A\} \cup \\ &\{z \leftarrow H^+(\tilde{\hat{r}}^i_y), H^-(\hat{r}^i_y), B^+(\hat{r}^i_y), B^-(\tilde{\hat{r}}^i_y) \mid r \in P_i, y \in \{t,m\}\} \cup \\ &\{z \leftarrow H^+(\tilde{\hat{r}}^i_h), H^-(\hat{r}^i_t), B^+(\hat{r}^i_h), B^-(\tilde{\hat{r}}^i_t) \mid r \in P_i\}. \end{aligned}$$

For the moment, we can assume that the $H[i]$ modules act in the same way as the $G[i]$ modules. In particular, assuming that each P_i has at least one SE model, there exists a situation where z is not derived. Below, on the one hand, we derive z to indicate the outcome of several checks, and finally force z to be included in an answer set. However, for the moment, we can use the operators $\hat{\sigma}$, $\hat{\pi}$, $\hat{\Sigma}$, $\hat{\Pi}$ in an analogous way as above. Hence, for instance, given $I \subseteq A^\Psi$ and an index i, we have $\hat{\pi}^i(I) = \{X \mid X \subseteq A, \hat{X}^i_h = I \cap \hat{A}^i_h\}$, and so on.

Next, we want to compare different models, e.g., $\Sigma(I)$ with $\hat{\Sigma}(I)$, for some $I \subseteq A^\Psi$. By the considerations above, this allows us to compare two SE models $\overline{S}, \overline{T}$ of Ψ.

We require the following property:

Lemma 1. *For $\Psi = \langle P_\alpha, \ldots, P_n\rangle$ a belief profile, we have $\overline{S} \in Min_a(SE(\Psi))$ iff*

(*i*) *for each $\overline{T} \in SE(\Psi)$, $\overline{S} \leq_a \overline{T}$, and*
(*ii*) *there exist $\alpha \leq i < j \leq n$ such that $\overline{S}_i \ominus \overline{S}_j \neq \overline{T}_i \ominus \overline{T}_j$.*

An analogous results holds for $Mod(\Psi)$ instead of $SE(\Psi)$.

Exploiting a somewhat dual method to this lemma, the following module derives, for given i, j,

- the atom z iff $\overline{S}_i \ominus \overline{S}_j \not\subseteq \overline{T}_i \ominus \overline{T}_j$, and
- the atom $z_{i,j}$ iff $\overline{S}_i \ominus \overline{S}_j = \overline{T}_i \ominus \overline{T}_j$.

For the latter, we require further new atoms $a^{i,j}_{x,\delta}$, for $x \in \{h,t,m\}$. Indeed, the compared models $\overline{S}$ and $\overline{T}$ are characterised via $I \subseteq A^\Psi$ by $\Sigma(I) = \overline{S}$ and $\hat{\Sigma}(I) = \overline{T}$, resp., $\Pi(I) = \overline{S}$ and $\hat{\Pi}(I) = \overline{T}$.

We define

$$\begin{aligned} C[i,j] = \{ \; & z \leftarrow a^i_x, \tilde{a}^j_x, \hat{a}^i_x, \hat{a}^j_x, && z \leftarrow a^i_x, \tilde{a}^j_x, \tilde{\hat{a}}^i_x, \tilde{\hat{a}}^j_x, \\ & z \leftarrow \tilde{a}^i_x, a^j_x, \hat{a}^i_x, \hat{a}^j_x, && z \leftarrow \tilde{a}^i_x, a^j_x, \tilde{\hat{a}}^i_x, \tilde{\hat{a}}^j_x, \\ & a^{i,j}_{x,\delta} \leftarrow a^i_x, \tilde{a}^j_x, \hat{a}^i_x, \tilde{\hat{a}}^j_x, && a^{i,j}_{x,\delta} \leftarrow a^i_x, \tilde{a}^j_x, \tilde{\hat{a}}^i_x, \hat{a}^j_x, \\ & a^{i,j}_{x,\delta} \leftarrow \tilde{a}^i_x, a^j_x, \hat{a}^i_x, \tilde{\hat{a}}^j_x, && a^{i,j}_{x,\delta} \leftarrow \tilde{a}^i_x, a^j_x, \tilde{\hat{a}}^i_x, \hat{a}^j_x, \\ & a^{i,j}_{x,\delta} \leftarrow a^i_x, a^j_x, \hat{a}^i_x, \hat{a}^j_x, && a^{i,j}_{x,\delta} \leftarrow a^i_x, a^j_x, \tilde{\hat{a}}^i_x, \tilde{\hat{a}}^j_x, \\ & a^{i,j}_{x,\delta} \leftarrow \tilde{a}^i_x, \tilde{a}^j_x, \hat{a}^i_x, \hat{a}^j_x, && a^{i,j}_{x,\delta} \leftarrow \tilde{a}^i_x, \tilde{a}^j_x, \tilde{\hat{a}}^i_x, \tilde{\hat{a}}^j_x \mid a \in A, x \in \{h,t,m\}\} \cup \\ & \{ \; z_{i,j} \leftarrow A^{i,j}_{h,\delta} \cup A^{i,j}_{t,\delta}, && z_{i,j} \leftarrow A^{i,j}_{m,\delta}\}. \end{aligned}$$

In other words, if we have guessed $\overline{S}$ and $\overline{T}$ in such a way that $\overline{S}_i \ominus \overline{S}_j \not\subseteq \overline{T}_i \ominus \overline{T}_j$, then $\overline{S} \leq_a \overline{T}$ cannot hold and we derive the spoiling atom z. In case $\overline{S}_i \ominus \overline{S}_j = \overline{T}_i \ominus \overline{T}_j$, we store this result by deriving an intermediate spoiling atom $z_{i,j}$. Below, we spoil if all relevant $z_{i,j}$'s have been derived.

Arbitration Merging. For a belief profile $\Psi = \langle P_1, \ldots, P_n \rangle$, we put things together as follows, where Z is the set $\{z_{i,j} \mid 1 \leq i < j \leq n\}$:

$$\begin{aligned} E(\Psi) = {} & \textstyle\bigcup_{i=1}^{n}(G[i] \cup H[i]) \cup \bigcup_{i=1}^{n}\bigcup_{j=i+1}^{n} C[i,j] \cup \\ & \{z \leftarrow Z, \quad \bot \leftarrow {\sim}z\} \cup \\ & \{g_1 \vee \cdots \vee g_{2n} \leftarrow\} \cup \\ & \{a_t^o \leftarrow g_i, a_m^i, \quad a_h^o \leftarrow g_i, a_m^i \\ & \ a_t^o \leftarrow g_{n+i}, a_t^i, \quad a_h^o \leftarrow g_{n+i}, a_h^i \mid a \in A, 1 \leq i \leq n\} \cup \\ & \{f_j \leftarrow g_{n+i}, a_m^j, \tilde{a}_t^i, \quad f_j \leftarrow g_{n+i}, \tilde{a}_m^j, a_t^i \mid 1 \leq i, j \leq n, a \in A\} \cup \\ & \{\bot \leftarrow f_1, \ldots f_n\}. \end{aligned}$$

Roughly speaking, the guess via the g_i's selects from which P_i we now add a pair (X, Y) into $SE(\nabla(\Psi))$. More precisely, if a g_i is selected, with $1 \leq i \leq n$, we add $(\overline{X}_i, \overline{X}_i)$ for the currently guessed $\overline{X} \in Mod(\Psi)$. Otherwise, i.e., when g_{n+i} is selected ($1 \leq i \leq n$), we add $\overline{S}_i = (X, Y)$, where $\overline{S} \in SE(\Psi)$ is the current guess, provided that Y matches *some* $\overline{X}_j$.

Let now for a set $\mathcal{I}$ of interpretations over A^Ψ,

$$\Sigma^o = \bigcup_{I \in \mathcal{I}} \{(X, Y) \mid X, Y \subseteq A, X_h^o = I \cap A_h^o, Y_t^o = I \cap A_t^o\}.$$

We obtain the following result:

Theorem 5. *$SE(\nabla(\Psi)) = \Sigma^o(AS(E(\Psi)))$.*

Basic Merging. We now continue with the encoding for basic merging. We already have most ingredients at hand. In fact, for a belief profile $\Psi = \langle P_0, \ldots, P_n \rangle$ define

$$\begin{aligned} F(\Psi) = {} & \textstyle\bigcup_{i=0}^{n}(G[i] \cup H[i]) \cup \bigcup_{i=1}^{n} C[0,i] \cup \\ & \{z \leftarrow z_{0,1}, \ldots, z_{0,n}, \quad \bot \leftarrow {\sim}z\} \cup \\ & \{g_0 \vee g_1 \leftarrow\} \cup \\ & \{a_t^o \leftarrow g_0, a_m^0, \quad a_h^o \leftarrow g_0, a_m^0, \quad a_t^o \leftarrow g_1, a_t^0, \quad a_h^o \leftarrow g_1, a_h^0 \mid a \in A\} \cup \\ & \{\bot \leftarrow g_1, a_m^0, \tilde{a}_t^0, \quad \bot \leftarrow g_1, \tilde{a}_m^0, a_t^0\}. \end{aligned}$$

$F(\Psi)$ follows the same ideas as used in $E(\Psi)$ but significantly simplifies due to the special role of P_0 in basic merging. Note that we require much less comparisons $C[0,i]$ here. As well, we only have to select classical and SE models of P_0 to become output atoms. Our result is thus as follows:

Theorem 6. *$SE(\Delta(\Psi)) = \Sigma^o(AS(F(\Psi)))$.*

Complexity. In our previous work [8], the following decision problem has been studied with respect to the revision operator $*$:

Given GLPs P, Q, R. Does $P * Q \models_s R$ hold?

This problem was shown to be Π_2^P-complete.

Accordingly, we give here results for the following problems:

Given a belief profile Ψ and a further program R. (1) Does $\nabla(\Psi) \models_s R$ hold? (2) Does $\Delta(\Psi) \models_s R$ hold?

By Theorem 4, it can be shown that the 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 above encodings such that these extensions possess an answer set iff the respective problem (1) or (2) does *not* hold. Since checking whether a program has at least one answer set is a problem on the second of layer of the polynomial hierarchy and our (extended) encodings are polynomial in the size of the encoded problems, the desired membership results follow.

Theorem 7. *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 Discussion

We have addressed the problem of merging of logic programs under the answer set semantics. Unlike related work in updating logic programs, but similar to our work in logic program revision [8], our approach is based on a monotonic characterisation 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 [12], the second being closer to [13]. Notably, since these merging operators are defined via a semantic characterisation, the results of merging are independent of the particular syntactic expression of a logic program. As well as giving properties of these operators, we also considered the complexity and an encoding scheme for both.

This work is original, given that it addresses merging in terms familiar to researchers in belief change. However, it applies these concepts in the context of logic programs. While we considered set-containment-based merging here, cardinality-based merging (which in fact would be closer to the specific operators proposed by Konieczny and Pino Pérez [13]) can also easily be defined.

References

1. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University (2003)
2. Zhang, Y., Foo, N.: Updating logic programs. In: ECAI'98, IOS (1998) 403–407
3. Alferes, J., Leite, J., Pereira, L., Przymusinska, H., Przymusinski, T.: Dynamic updates of non-monotonic knowledge bases. JLP **45**(1–3) (2000) 43–70
4. Leite, J.: Evolving Knowledge Bases: Specification and Semantics. IOS (2003)

5. Inoue, K., Sakama, C.: Updating extended logic programs through abduction. In: LPNMR'99, Springer (1999) 147–161
6. Eiter, T., Fink, M., Sabbatini, G., Tompits, H.: On properties of update sequences based on causal rejection. TPLP **2**(6) (2002) 711–767
7. Delgrande, J., Schaub, T., Tompits, H.: A preference-based framework for updating logic programs. In: LPNMR'07, Springer (2007) 71–83
8. Delgrande, J., Schaub, T., Tompits, H., Woltran, S.: Belief revision of logic programs under answer set semantics. In: KR'08, AAAI Press (2008) 411–421
9. Turner, H.: Strong equivalence made easy: nested expressions and weight constraints. TPLP **3**(4-5) (2003) 609–622
10. Lifschitz, V., Pearce, D., Valverde, A.: Strongly equivalent logic programs. ACM TOCL **2**(4) (2001) 526–541
11. Gärdenfors, P.: Knowledge in Flux MIT Press (1988)
12. Liberatore, P., Schaerf, M.: Arbitration (or how to merge knowledge bases). IEEE TKDE **10**(1) (1998) 76–90
13. Konieczny, S., Pino Pérez, R.: Merging information under constraints: A logical framework. JLC **12**(5) (2002) 773–808
14. Lifschitz, V., Woo, T.: Answer sets in general nonmonotonic reasoning (preliminary report). In: KR'92, Morgan Kaufmann (1992) 603–614
15. Inoue, K., Sakama, C.: Negation as failure in the head. JLP **35**(1) (1998) 39–78
16. Eiter, T., Tompits, H., Woltran, S.: On solution correspondences in answer set programming. In: IJCAI'05, Professional Book Center (2005) 97–102
17. Cabalar, P., Ferraris, P.: Propositional theories are strongly equivalent to logic programs. TPLP **7**(6) (2007) 745–759
18. Baral, C., Kraus, S., Minker, J., Subrahmanian, V.: Combining multiple knowledge bases consisting of first order theories. CI **8**(1) (1992) 45–71
19. Revesz, P.: On the semantics of theory change: Arbitration between old and new information. In: ACM Principles DBS (1993) 71–82
20. Lin, J., Mendelzon, A.: Knowledge base merging by majority. In: Dynamic Worlds: From the Frame Problem to Knowledge Management. Kluwer (1999) 195–218
21. Konieczny, S., Lang, J., Marquis, P.: Distance-based merging: a general framework and some complexity results. In: KR'02 (2002) 97–108
22. Meyer, T.: On the semantics of combination operations. JANCL **11**(1-2) (2001) 59–84
23. Spohn, W.: Ordinal conditional functions: A dynamic theory of epistemic states. In: Causation in Decision, Belief Change, and Statistics. Kluwer (1988) 105–134
24. Booth, R.: Social contraction and belief negotiation. In: KR'02 (2002) 375–384
25. Benferhat, S., Dubois, D., Kaci, S., Prade, H.: Possibilistic merging and distance-based fusion of propositional information. AMAI **34**(1-3) (2003) 217–252