On Semantic Update Operators for Answer-Set Programs

Martin Slota and **João Leite**[1]

Abstract. Logic programs under the stable models semantics, or answer-set programs, provide an expressive rule based knowledge representation framework, featuring formal, declarative and well-understood semantics. However, handling the evolution of rule bases is still a largely open problem. The AGM framework for belief change was shown to give inappropriate results when directly applied to logic programs under a nonmonotonic semantics such as the stable models. Most approaches to address this issue, developed so far, proposed update operators based on syntactic conditions for rule rejection.

More recently, AGM revision has been successfully applied to a significantly more expressive semantic characterisation of logic programs based on SE models. This is an important step, as it changes the focus from the evolution of a syntactic representation of a rule base to the evolution of its semantic content.

In this paper, we borrow results from the area of belief update to tackle the problem of updating (instead of revising) logic programs. We prove a representation theorem which makes it possible to constructively define any operator satisfying a set of postulates derived from Katsuno and Mendelzon's postulates for belief update. We define a specific operator based on this theorem and compare the behaviour of this operator with syntactic update operators defined in the literature. Perhaps surprisingly, we uncover a very serious drawback in a large class of semantic update operators to which it belongs.

1 Introduction

Answer-Set Programming (ASP) [11, 5] is now widely recognised as a valuable approach for knowledge representation and reasoning, mostly due to its simple and well understood declarative semantics, its rich expressive power, and the existence of efficient implementations.

However, the dynamic character of many applications that can benefit from ASP calls for the development of ways to deal with the evolution of answer-set programs and the inconsistencies that may arise.

The problems associated with knowledge evolution have been extensively studied, over the years, by researchers in different research communities, namely in the context of Classical Logic, and in the context of Logic Programming.

In the context of Classical Logic, the seminal work by Alchourrón, Gärdenfors and Makinson (AGM) [1] proposed a set of desirable properties of belief change operators, now called *AGM postulates*. Subsequently, Katsuno and Mendelzon distinguished *update* and *revision* as two very related but ultimately different belief change operations [13]. While revision deals with incorporating new information about a *static* world, update takes place when changes occurring in a *dynamic* world are recorded. Katsuno and Mendelzon also formulated a separate set of postulates for updates, now known as *KM postulates*. One of the specific update operators that satisfies these postulates is Winslett's minimal change update operator [21].

In the context of Logic Programming, earlier approaches based on literal inertia [17] proved not sufficiently expressive for dealing with rule updates, leading to the development of semantics for rule updates based on different intuitions, principles and constructions, when compared to their classical counterparts. For example, the introduction of the *causal rejection principle* [15] lead to several approaches to rule updates [3, 14, 10, 2], all of them with a strong syntactic flavour. Though useful in practical applications, it turned out that most of these semantics have undesirable behaviour in certain situations [2, 19]. For example, in most of these semantics ([2] being an exception), tautological updates may influence the original rule base, a behaviour that is highly undesirable when considering knowledge update operations[2]. But more important, the syntactic nature of these formalisms makes any analysis of their semantic properties a daunting task. The postulates set forth in the context of Classical Logic (AGM and KM) were also studied in the context of Logic Programming, only to find that their formulations based on a nonmonotonic semantics such as the stable models semantics were inappropriate [10].

Recently, in [7], AGM based revision was reformulated in the context of Logic Programming in a manner analogous to belief revision in classical propositional logic, and specific revision operators for logic programs were investigated. Central to this novel approach are *SE models* [20] which provide a monotonic semantic characterisation of logic programs that is strictly more expressive than the answer-set semantics. Furthermore, two programs have the same set of SE models if and only if they are strongly equivalent [16], which means that programs $\mathcal{P}, \mathcal{Q}$ with the same set of SE models can be modularly replaced by each other with respect to the answer-set semantics, because strong equivalence guarantees that $\mathcal{P} \cup \mathcal{R}$ has the same answer sets as $\mathcal{Q} \cup \mathcal{R}$ for any program $\mathcal{R}$.

Indeed, the results of [7] constitute an important breakthrough in the research of answer-set program evolution, as they change the focus from the syntactic representation of a program to its semantic content.

In this paper, we follow a similar path, but to tackle the problem of answer-set program *updates*, instead of *revision* as in [7].

Using *SE models*, we adapt the KM postulates to answer-set program updates and show a representation theorem which provides a constructive characterisation of program update operators satisfying the postulates, making it possible to define and compute any operator satisfying the postulates using an intuitive construction. We also show how this constructive characterisation can be used by defining

---

[1] CENTRIA & Departamento de Informática, Universidade Nova de Lisboa, Email: {`mslota,jleite`}`@fct.unl.pt`

[2] This behaviour is also exhibited by [18, 22], where change operators were defined, somehow intermixing update and revision.

a specific answer-set program update operator which is a counterpart of Winslett's belief update operator.

However, while investigating the operator's properties, we uncover a serious drawback which, as it turns out, extends to all meaningful answer-set program update operators based on SE models and the Katsuno and Mendelzon's approach to updates. This finding is very important as it guides the research on updates of answer-set programs away from the semantic approach materialised in Katsuno and Mendelzon's postulates or, alternatively, to the development of semantic characterisations of answer-set programs, richer than SE models, that are appropriate to describe their dynamic behaviour.

The remainder of this paper is structured as follows: In Sect. 2 we introduce the notions that are necessary throughout the rest of the paper. Section 3 contains the reformulation of KM postulates for logic program updates and the representation theorem that establishes a general constructive characterisation of program update operators obeying the postulates. In Sect. 3 we also show how this theorem can be used by defining a specific program update operator. In Sect. 4 we analyse the operator defined in Sect. 3 and establish that a large class of semantic update operators to which it belongs exhibits an undesired behaviour. We then conclude in Sect. 5.

2 Preliminaries

We consider a propositional language over a finite set of propositional variables $\mathcal{L}$ and the usual set of propositional connectives to form propositional formulae. A (propositional) interpretation is any $I \subseteq \mathcal{L}$ and the set of all interpretations is $\mathcal{I} = 2^{\mathcal{L}}$. We use the standard semantics for propositional formulae and for a formula ϕ we denote by $[\![\phi]\!]$ the set of its models. We say a formula ϕ is *satisfiable* if $[\![\phi]\!]$ is nonempty and that it is *complete* if $[\![\phi]\!]$ is a singleton set. For formulae ϕ, ψ we say ϕ *implies* ψ, denoted by $\phi \models \psi$, if $[\![\phi]\!] \subseteq [\![\psi]\!]$ and that ϕ is *equivalent to* ψ, denoted by $\phi \equiv \psi$, if $[\![\phi]\!] = [\![\psi]\!]$. As we are dealing with the finite case, every knowledge base can be expressed by a single formula.

2.1 Belief Update

Update is a belief change operation which captures changes occurring in the modelled environment [13]. An update operator is simply any function that takes two formulae as arguments, representing the original knowledge base and its update, respectively, and returns a formula representing the updated knowledge base.

Katsuno and Mendelzon, in [13], proposed the following eight desirable properties of a belief update operator $\diamond$, that should hold for all propositional formulae $\phi, \phi_1, \phi_2, \psi, \psi_1, \psi_2$:

(KM 1) $\phi \diamond \psi \models \psi$.
(KM 2)   If $\phi \models \psi$, then $\phi \diamond \psi \equiv \phi$.
(KM 3)   If both ϕ and ψ are satisfiable, then $\phi \diamond \psi$ is satisfiable.
(KM 4)   If $\phi_1 \equiv \phi_2$ and $\psi_1 \equiv \psi_2$, then $\phi_1 \diamond \psi_1 \equiv \phi_2 \diamond \psi_2$.
(KM 5) $(\phi \diamond \psi) \wedge \chi \models \phi \diamond (\psi \wedge \chi)$.
(KM 6)   If $\phi \diamond \psi_1 \models \psi_2$ and $\phi \diamond \psi_2 \models \psi_1$, then $\phi \diamond \psi_1 \equiv \phi \diamond \psi_2$.
(KM 7) $(\phi \diamond \psi_1) \wedge (\phi \diamond \psi_2) \models \phi \diamond (\psi_1 \vee \psi_2)$ if ϕ is complete.
(KM 8) $(\phi_1 \vee \phi_2) \diamond \psi \equiv (\phi_1 \diamond \psi) \vee (\phi_2 \diamond \psi)$.

They also proved an important representation theorem which makes it possible to define and compute any operator satisfying these postulates using an intuitive construction. It is based on treating the models of a knowledge base as possible real states of the modelled environment. An update of the original knowledge base ϕ is performed by modifying each of its models as little as possible to make

it consistent with the new information in the update ψ, obtaining a new set of interpretations – the models of the updated knowledge base. More formally,

$$[\![\phi \diamond \psi]\!] = \bigcup_{I \in [\![\phi]\!]} \mathsf{incorporate}([\![\psi]\!], I) \ ,$$

where $\mathsf{incorporate}(S, I)$ returns the members of S closer to I. A natural way of defining $\mathsf{incorporate}(S, I)$ is by assigning a preorder $\leq_I$ to each interpretation I and taking the minima of S w.r.t. $\leq_I$, i.e.

$$\mathsf{incorporate}(S, I) = \min(S, \leq_I)$$

Hence, we define the notion of an *order assignment* as follows:

Definition 1. *A preorder over S is a reflexive and transitive binary relation over S. A* strict preorder over S *is an irreflexive and transitive binary relation over S. The strict preorder $<$ induced by a preorder $\leq$ is defined for all $a, b \in S$ as follows: $a < b$ if and only if $a \leq b$ and not $b \leq a$.*

A partial order over S is any preorder over S that is antisymmetric.

The set of minimal elements of $T \subseteq S$ w.r.t. a preorder $\leq$ over S is defined as $\min(T, \leq) = \{\, x \in T \mid (\not\exists y \in T)(y < x) \,\}$ where $<$ is the strict preorder induced by $\leq$.

Definition 2 (Preorder assignment, partial order assignment)**.** *A preorder assignment is any function ω which assigns a preorder $\leq_I^\omega$ over $\mathcal{I}$ to each interpretation $I \in \mathcal{I}$. By $<_I^\omega$ we denote the strict preorder induced by $\leq_I^\omega$.*

A partial order assignment is any preorder assignment ω such that $\leq_I^\omega$ is a partial order over $\mathcal{I}$ for every interpretation $I \in \mathcal{I}$.

A natural condition on the assigned preorders is that every interpretation be the closest to itself. This is captured by the notion of a *faithful order assignment*:

Definition 3 (Faithful assignment)**.** *A preorder assignment ω is faithful if for every interpretation I the following condition is satisfied: For every $J \in \mathcal{I}$ with $J \neq I$ it holds that $I <_I^\omega J$.*

The mentioned representation theorem relates the above described construction with the postulates:

Theorem 4 ([13])**.** *Let $\diamond$ be a belief update operator. Then the following conditions are equivalent:*

1. *The operator $\diamond$ satisfies conditions* (KM 1) – (KM 8)*.*
2. *There exists a faithful preorder assignment ω such that for all formulae ϕ, ψ*

$$[\![\phi \diamond \psi]\!] = \bigcup_{I \in [\![\phi]\!]} \min([\![\psi]\!], \leq_I^\omega) \ .$$

3. *There exists a faithful partial order assignment ω such that for all formulae ϕ, ψ*

$$[\![\phi \diamond \psi]\!] = \bigcup_{I \in [\![\phi]\!]} \min([\![\psi]\!], \leq_I^\omega) \ .$$

The most commonly used and studied belief update operator satisfying Katsuno and Mendelzon's postulates is Winslett's minimal change update operator [21]. It can be defined using the above presented construction as follows: Let the preorder assignment W be defined for any interpretations I, J, K as follows:

$$J \leq_I^W K \text{ if and only if } (J \div I) \subseteq (K \div I)$$

where $\div$ denotes set-theoretic symmetric difference. Winslett's operator $\diamond^W$ is then characterised by

$$[\![\phi \diamond^W \psi]\!] = \bigcup_{I \in [\![\phi]\!]} \min([\![\psi]\!], \leq_I^W) \ .$$

2.2 Logic Programming

We define the syntax and semantics of logic programs, borrowing some notation used in [7].

An *atom* is any $p \in \mathcal{L}$. A *literal* is an atom p or its default negation $\sim p$. Given a set of literals X, we define $X^{+} = \{ p \in \mathcal{L} \mid p \in X \}$, $X^{-} = \{ p \in \mathcal{L} \mid \sim p \in X \}$ and $\sim X = \{ \sim p \mid p \in X \cap \mathcal{L} \}$. A *rule* is any expression of the form

$$p_1; \ldots; p_m; \sim q_1; \ldots; \sim q_n \leftarrow r_1, \ldots, r_o, \sim s_1, \ldots, \sim s_p \quad (1)$$

where m, n, o, p are natural numbers such that $m + n \geq 1$ and $p_i, q_j, r_k, s_l \in \mathcal{L}$ for $i \in \{1, 2, \ldots, m\}$, $j \in \{1, 2, \ldots, n\}$, $k \in \{1, 2, \ldots, o\}$, $l \in \{1, 2, \ldots, p\}$. Operators ';' and ',' express disjunctive and conjunctive connectives, respectively. We also define

$$H(r) = \{ p_1, \ldots, p_m, \sim q_1, \ldots, \sim q_n \} \ ,$$
$$B(r) = \{ r_1, \ldots, r_o, \sim s_1, \ldots, \sim s_p \} \ .$$

$H(r)$ is dubbed the *head of r* and $B(r)$ the *body of r*. For simplicity, we sometimes use a set-based notation, expressing a rule (1) as

$$H(r)^{+}; \sim H(r)^{-} \leftarrow B(r)^{+}, \sim B(r)^{-} \ .$$

A *program* is a set of rules. A rule as in (1) is called *normal* if $m = 1$ and $n = 0$; *definite* if it is normal and $p = 0$; *fact* if $o = p = 0$ and $m + n = 1$. A program is *normal* if all its rules are normal and it is *definite* if all its rules are definite.

Turning to the semantics, we need to define answer sets and SE models of a logic program. We start by defining the more basic notion of a *(classical) model* of a logic program. A *model* of a program $\mathcal{P}$ is an interpretation in which all rules from $\mathcal{P}$ are true according to the standard definition of truth in propositional logic, and where default negation is treated as classical negation. We write $I \models \mathcal{P}$ if and only if I is a classical model of $\mathcal{P}$.

An interpretation I is an *answer set* of a program $\mathcal{P}$ if and only if I is a subset-minimal model of

$$\mathcal{P}^{I} = \big\{ H(r)^{+} \leftarrow B(r)^{+} \mid r \in \mathcal{P} \land H(r)^{-} \subseteq I$$
$$\land B(r)^{-} \cap I = \emptyset \big\} \ .$$

An SE interpretation [20] has a richer structure than a propositional interpretation. It consists of a pair of propositional interpretations, the first representing the true atoms and the second the atoms that are not false. Formally:

Definition 5 (SE Interpretation). *An* SE interpretation *is a pair* $X = \langle I, J \rangle$ *of interpretations such that* $I \subseteq J \subseteq \mathcal{L}$. *A set S of SE interpretations is* well-defined *if, for each* $\langle I, J \rangle \in S$, *also* $\langle J, J \rangle \in S$. *The set of all SE interpretations is denoted by* $\mathcal{I}^{\mathsf{SE}}$. *For every SE interpretation* $X = \langle I, J \rangle$ *we denote by* X^{*} *the SE interpretation* $\langle J, J \rangle$.

We are now ready to define the notion of an SE model. SE models provide a monotonic characterisation of logic programs that we later use to adapt KM postulates for program updates, similarly as was done for revision in [7].

Definition 6 (SE Models). *An SE interpretation* $\langle I, J \rangle$ *is an* SE model *of a program* $\mathcal{P}$ *if* $J \models \mathcal{P}$ *and* $I \models \mathcal{P}^{J}$. *The set of all SE models of a program* $\mathcal{P}$ *is denoted by* $\llbracket \mathcal{P} \rrbracket^{\mathsf{SE}}$.

We say that a program $\mathcal{P}$ *is* satisfiable *if* $\llbracket \mathcal{P} \rrbracket^{\mathsf{SE}}$ *is nonempty and that it is* basic *if* $\llbracket \mathcal{P} \rrbracket^{\mathsf{SE}}$ *is either a singleton set, or has two elements* X, Y *such that* $Y = X^{*}$. *For programs* $\mathcal{P}, \mathcal{Q}$ *we say* $\mathcal{P}$ implies $\mathcal{Q}$, *denoted by* $\mathcal{P} \models_s \mathcal{Q}$, *if* $\llbracket \mathcal{P} \rrbracket^{\mathsf{SE}} \subseteq \llbracket \mathcal{Q} \rrbracket^{\mathsf{SE}}$, *and that* $\mathcal{P}$ *is* strongly equivalent *to* $\mathcal{Q}$, *denoted by* $\mathcal{P} \equiv_s \mathcal{Q}$, *if* $\llbracket \mathcal{P} \rrbracket^{\mathsf{SE}} = \llbracket \mathcal{Q} \rrbracket^{\mathsf{SE}}$.

Note that J is an answer set of $\mathcal{P}$ if and only if $\langle J, J \rangle \in \llbracket \mathcal{P} \rrbracket^{\mathsf{SE}}$ and no $\langle I, J \rangle \in \llbracket \mathcal{P} \rrbracket^{\mathsf{SE}}$ with $I \subsetneq J$ exists. Also, we have $\langle J, J \rangle \in \llbracket \mathcal{P} \rrbracket^{\mathsf{SE}}$ if and only if J is a classical model of $\mathcal{P}$.

The following result pinpoints the fact that the set of SE models of a logic program is always well-defined.

Proposition 7 ([7]). *A set of SE interpretations S is well-defined if and only if* $S = \llbracket \mathcal{P} \rrbracket^{\mathsf{SE}}$ *for some program* $\mathcal{P}$.

3 Semantic Program Update Operators

In this section we study semantic program update operators based on Katsuno and Mendelzon's update postulates and SE models. Since SE models provide a monotonic characterisation of logic programs, the analysis provided in [10], which showed KM postulates not appropriate to be used with nonmonotonic semantics, no longer applies. We are then able to re-define KM postulates in the context of Logic Programming and SE models, and show a representation theorem which provides a constructive characterisation of update operators satisfying the postulates. These operators are semantic in nature, in contrast with those defined in [3, 14, 10, 2], and are in line with KM postulates, in contrast with those defined in [18, 22].

We start by defining a program update operator as a function that takes two programs, the original program and the updating program, as arguments and returns the updated program:

Definition 8 (Program update operator). *A program update operator is any function that assigns a program to each pair of programs.*

In order to reformulate postulates (KM 1) to (KM 8) for logic programs under the SE models semantics, we first need to specify what a conjunction and disjunction of logic programs is. To this end, we introduce program conjunction and disjunction operators. These are required to assign, to each pair of programs, a program whose set of SE models is an intersection and union, respectively, of the sets of SE models of the argument programs. Formally:

Definition 9 (Program conjunction and disjunction). *A binary operator* $\dot{\land}$ *on the set of all programs is a* program conjunction operator *if an only if* $\llbracket \mathcal{P} \dot{\land} \mathcal{Q} \rrbracket^{\mathsf{SE}} = \llbracket \mathcal{P} \rrbracket^{\mathsf{SE}} \cap \llbracket \mathcal{Q} \rrbracket^{\mathsf{SE}}$ *for all programs* $\mathcal{P}, \mathcal{Q}$.
A binary operator $\dot{\lor}$ *on the set of all programs is a* program disjunction operator *if an only if* $\llbracket \mathcal{P} \dot{\lor} \mathcal{Q} \rrbracket^{\mathsf{SE}} = \llbracket \mathcal{P} \rrbracket^{\mathsf{SE}} \cup \llbracket \mathcal{Q} \rrbracket^{\mathsf{SE}}$ *for all programs* $\mathcal{P}, \mathcal{Q}$.

In the following we assume that some program conjunction and disjunction operators $\dot{\land}, \dot{\lor}$ are given. Also note that the program conjunction operator is the same as the *expansion operator* defined in [7]. A syntactic operator for program disjunction operator can be defined by translating the argument programs into the logic of here and there, taking their disjunction and transforming the resulting formula back into a logic program (using results from [6]).

Finally, we need to substitute the notion of a complete formula used in (KM 7) with a suitable class of logic programs. It turns out that the notion of a *basic program* is a natural candidate for this purpose because the set of SE models of a program is always well-defined. We recall that a program is called *basic* if and only if it either has a single SE model $\langle J, J \rangle$ or a pair of SE models $\langle I, J \rangle, \langle J, J \rangle$.

The following are the reformulated postulates of a program update operator $\oplus$ that should hold for all programs $\mathcal{P}, \mathcal{P}_1, \mathcal{P}_2, \mathcal{Q}, \mathcal{Q}_1, \mathcal{Q}_2$:

(PU 1) $\mathcal{P} \oplus \mathcal{Q} \models_s \mathcal{Q}$.
(PU 2)   If $\mathcal{P} \models_s \mathcal{Q}$, then $\mathcal{P} \oplus \mathcal{Q} \equiv_s \mathcal{P}$.
(PU 3)   If both $\mathcal{P}$ and $\mathcal{Q}$ are satisfiable, then $\mathcal{P} \oplus \mathcal{Q}$ is satisfiable.

(PU 4)   If $\mathcal{P}_1 \equiv_s \mathcal{P}_2$ and $\mathcal{Q}_1 \equiv_s \mathcal{Q}_2$, then $\mathcal{P}_1 \oplus \mathcal{Q}_1 \equiv_s \mathcal{P}_2 \oplus \mathcal{Q}_2$.
(PU 5) $(\mathcal{P} \oplus \mathcal{Q}) \mathbin{\dot\wedge} \mathcal{R} \models_s \mathcal{P} \oplus (\mathcal{Q} \mathbin{\dot\wedge} \mathcal{R})$.
(PU 6)   If $\mathcal{P} \oplus \mathcal{Q}_1 \models_s \mathcal{Q}_2$ and $\mathcal{P} \oplus \mathcal{Q}_2 \models_s \mathcal{Q}_1$,
          then $\mathcal{P} \oplus \mathcal{Q}_1 \equiv_s \mathcal{P} \oplus \mathcal{Q}_2$.
(PU 7) $(\mathcal{P} \oplus \mathcal{Q}_1) \mathbin{\dot\wedge} (\mathcal{P} \oplus \mathcal{Q}_2) \models_s \mathcal{P} \oplus (\mathcal{Q}_1 \mathbin{\dot\vee} \mathcal{Q}_2)$ if $\mathcal{P}$ is basic.
(PU 8) $(\mathcal{P}_1 \mathbin{\dot\vee} \mathcal{P}_2) \oplus \mathcal{Q} \equiv_s (\mathcal{P}_1 \oplus \mathcal{Q}) \mathbin{\dot\vee} (\mathcal{P}_2 \oplus \mathcal{Q})$.

We now turn to a constructive characterisation of program update operators satisfying postulates (PU 1) – (PU 8). Similarly as for belief update, it is based on the equation

$$\llbracket \mathcal{P} \oplus \mathcal{Q} \rrbracket^{\mathsf{SE}} = \bigcup_{Z \in \llbracket \mathcal{P} \rrbracket^{\mathsf{SE}}} \min(\llbracket \mathcal{Q} \rrbracket^{\mathsf{SE}}, \leq_Z) \ ,$$

where $\leq_Z$ is a preorder over $\mathcal{I}^{\mathsf{SE}}$ assigned to Z using some *SE preorder assignment*:

Definition 10 (SE preorder assignment, SE partial order assignment). *An* SE binary relation assignment *is any function ω which assigns a binary relation R_Z^ω over $\mathcal{I}^{\mathsf{SE}}$ to every SE interpretation Z.*

An SE preorder assignment *is any SE binary relation assignment ω for which R_Z^ω is a preorder over $\mathcal{I}^{\mathsf{SE}}$. In this case we denote R_Z^ω by $\leq_Z^\omega$ and use infix notation instead of prefix notation. By $<_Z^\omega$ we denote the strict preorder induced by $\leq_Z^\omega$. Given an SE preorder assignment ω we denote by $\mathsf{op}(\omega)$ the set of program update operators $\oplus$ such that for all programs $\mathcal{P}, \mathcal{Q}$ it holds that*

$$\llbracket \mathcal{P} \oplus \mathcal{Q} \rrbracket^{\mathsf{SE}} = \bigcup_{Z \in \llbracket \mathcal{P} \rrbracket^{\mathsf{SE}}} \min(\llbracket \mathcal{Q} \rrbracket^{\mathsf{SE}}, \leq_Z^\omega) \ .$$

We say that ω is well-defined *if $\mathsf{op}(\omega)$ is nonempty and for each $\oplus \in \mathsf{op}(\omega)$ we say that ω characterises $\oplus$.*

An SE partial order assignment *is any SE preorder assignment ω such that $\leq_Z^\omega$ is a partial order over $\mathcal{I}^{\mathsf{SE}}$ for every $Z \in \mathcal{I}^{\mathsf{SE}}$.*

Similarly as before, we will require the preorder assignment to be faithful, i.e. to consider each SE interpretation the closest to itself.

Definition 11 (Faithful SE preorder assignment). *An SE preorder assignment ω is* faithful *if for every SE interpretation Z the following condition is satisfied: For every $X \in \mathcal{I}^{\mathsf{SE}}$ with $X \neq Z$ it holds that $Z <_Z^\omega X$.*

Interestingly, faithful assignments generate the same class of operators as the larger class of semi-faithful assignments, defined as:

Definition 12 (Semi-faithful SE preorder assignment). *An SE preorder assignment ω is* semi-faithful *if for every SE interpretation Z the following conditions are satisfied:*

1. *For every $X \in \mathcal{I}^{\mathsf{SE}}$ with $X \neq Z$ and $X \neq Z^*$ either $Z <_Z^\omega X$ or $Z^* <_Z^\omega X$.*
2. *There is no $X \in \mathcal{I}^{\mathsf{SE}}$ such that $X <_Z^\omega Z$.*

Finally, we require the SE preorder assignment to satisfy one further condition which is again related to the well-definedness of sets of SE models of a program. It can naturally be seen as the semantic counterpart of (PU 7).

Definition 13 (Organised SE preorder assignment). *An SE preorder assignment ω is* organised *if for any SE interpretations X, Z and any well-defined sets of SE interpretations S_1, S_2 the following condition is satisfied: If X is a minimum of S_1 w.r.t. either $\leq_Z^\omega$ or $\leq_{Z^*}^\omega$ and it is a minimum of S_2 w.r.t. either $\leq_Z^\omega$ or $\leq_{Z^*}^\omega$, then it is also a minimum of $S_1 \cup S_2$ w.r.t. either $\leq_Z^\omega$ or $\leq_{Z^*}^\omega$.*

We are now ready to formulate the main result of this section:

Theorem 14 (Representation theorem). *Let $\oplus$ be a program update operator. Then the following conditions are equivalent:*

1. *The operator $\oplus$ satisfies conditions* (PU 1) – (PU 8).
2. *There exists a semi-faithful and organised SE preorder assignment ω such that $\oplus \in \mathsf{op}(\omega)$, i.e. for all programs $\mathcal{P}, \mathcal{Q}$ it holds that*

$$\llbracket \mathcal{P} \oplus \mathcal{Q} \rrbracket^{\mathsf{SE}} = \bigcup_{Z \in \llbracket \mathcal{P} \rrbracket^{\mathsf{SE}}} \min \left(\llbracket \mathcal{Q} \rrbracket^{\mathsf{SE}}, \leq_Z^\omega \right) \ .$$

3. *There exists a faithful and organised SE partial order assignment ω such that $\oplus \in \mathsf{op}(\omega)$, i.e. for all programs $\mathcal{P}, \mathcal{Q}$ it holds that*

$$\llbracket \mathcal{P} \oplus \mathcal{Q} \rrbracket^{\mathsf{SE}} = \bigcup_{Z \in \llbracket \mathcal{P} \rrbracket^{\mathsf{SE}}} \min \left(\llbracket \mathcal{Q} \rrbracket^{\mathsf{SE}}, \leq_Z^\omega \right) \ .$$

This theorem provides a constructive characterisation of program update operators satisfying the defined postulates. It facilitates the analysis of their semantic properties as well as their computational complexity. Note also that it implies that as far as the defined operator is concerned, the larger class of semi-faithful and organised SE preorder assignments is equivalent to the smaller class of faithful and organised SE partial order assignments. Furthermore, it offers a strategy for defining operators satisfying the postulates that can be directly implemented. This strategy is also complete in the sense that, up to strong equivalence, all operators satisfying the postulates can be characterised and distinguished by applying this strategy.

In what follows, we define a specific update operator based on Winslett's minimal change update operator [21] defined in the previous section. As in [7], in case of the SE models semantics, the preorder needs to give preference to the second component of SE interpretations. We define the SE binary relation assignment W for any SE interpretations $X = \langle I_1, J_1 \rangle, Y = \langle I_2, J_2 \rangle, Z = \langle K, L \rangle$ as follows: $X \leq_Z^W Y$ if and only if the following conditions are satisfied:

1. $(J_1 \div L) \subseteq (J_2 \div L)$;
2. If $(J_1 \div L) = (J_2 \div L)$, then $(I_1 \div K) \setminus \Delta \subseteq (I_2 \div K) \setminus \Delta$ where $\Delta = J_1 \div L$.

Intuitively, first we compare the differences between the second components of X and Y with respect to Z. If they are equal, we compare the differences between the first components of X and Y with respect to Z, but now ignoring the differences of the second components. The following result shows that W indeed satisfies the necessary conditions to characterise a program update operator satisfying the postulates.

Proposition 15. *The assignment W is a well-defined, faithful and organised SE preorder assignment.*

Let us pick an arbitrary program update operator $\oplus^W$ from $\mathsf{op}(W)$ (note that $\mathsf{op}(W)$ is nonempty since W is well-defined). Then, as a consequence of the theorem and the above proposition, we obtain

Corollary 16. *The program update operator $\oplus^W$ satisfies conditions* (PU 1) – (PU 8).

4 Support in Semantic Program Updates

In this section we take a closer look at the behaviour of semantic program update operators.

One of the benefits of dealing with program updates on a semantic level is that semantic properties that are rather difficult to show for syntactic update operators are much easier to be analysed and proven. For example, one of the most widespread and counterintuitive side effects of syntactic updates is that they are sensitive to tautological updates, with the exception of [2]. In case of semantic update operators, such a behaviour is easily shown to be impossible given that the operator satisfies postulate (PU 2).

However, semantic update operators do not always behave the way we expect. Consider first an example using the update operator $\oplus^W$ defined in the previous section[3]:

Example 17. *Let the programs $\mathcal{P}_1, \mathcal{P}_2$ and $\mathcal{Q}$ be as follows:*

$$\mathcal{P}_1 : \quad p. \qquad \mathcal{P}_2 : \quad p \leftarrow q. \qquad \mathcal{Q} : \quad \sim q.$$
$$q. \qquad\qquad q.$$

It can be easily verified that[4]:

$$[\![\mathcal{P}_1 \oplus^W \mathcal{Q}]\!]^{\mathsf{SE}} = [\![\mathcal{P}_2 \oplus^W \mathcal{Q}]\!]^{\mathsf{SE}} = \{\, \langle p, p \rangle \,\} \quad .$$

Hence, both $\mathcal{P}_1 \oplus^W \mathcal{Q}$ and $\mathcal{P}_2 \oplus^W \mathcal{Q}$ have the single answer set $I = \{\, p \,\}$. In case of $\mathcal{P}_1 \oplus^W \mathcal{Q}$ this is indeed the expected result. But in case of $\mathcal{P}_2 \oplus \mathcal{Q}$ we can see that p is true in I even though there is no rule in $\mathcal{P}_2 \cup \mathcal{Q}$ justifying it, i.e. there is no rule with p in its head and its body true in I. Hence, the behaviour of $\oplus^W$ is in disaccord with intuitions underlying most Logic Programming semantics.

In the following we show that such counterintuitive behaviour is not specific to $\oplus^W$, but extends to a large class of semantic update operators for answer-set programs based on SE models and Katsuno and Mendelzon's postulates for update.

The property of support [4, 9] is one of the basic conditions that Logic Programming semantics are intuitively designed to satisfy. In the static case, this property can be formulated as follows:

Definition 18 (Support in the static case)**.** *Let $\mathcal{P}$ be a program, L a literal and I an interpretation. We say $\mathcal{P}$ supports L in I if and only if there is some rule $r \in \mathcal{P}$ such that $L \in H(r)$ and $I \models B(r)$.*

A Logic Programming semantics SEM is supported *if for each model I of a program $\mathcal{P}$ under SEM the following condition is satisfied: Every atom $p \in I$ is supported by $\mathcal{P}$ in I.*

Note that the widely accepted Logic Programming semantics, such as the answer-set and well-founded semantics, are supported (see [8, 9] for more on properties of Logic Programming semantics).

It is only natural to require that program update operators do not neglect this essential property which also gives rise to much of the intuitive appeal of Logic Programming systems. In particular, we conjecture that an update operator for answer-set programs should obey the following properties related to support:

---

[3] It has been shown that Winslett's update operator has some drawbacks, just as other update operators previously proposed in the context of Classical Logic do (see [12] for a survey). Nevertheless, we decided to choose Winslett's update operator as the basis to define a program update operator and illustrate its properties because it is one of the most extensively studied and understood update operators, and because the undesired behaviour illustrated in this example is shared by a much larger class of update operators based on KM postulates and SE models – as we shall see – and not a specific problem due to our choice of Winslett's update operator.

[4] For the sake of readability, we omit the curly braces when listing SE interpretations. For example, instead of $\langle \{\, p \,\}, \{\, p, q \,\} \rangle$ we simply write $\langle p, pq \rangle$.

Definition 19 (Support for answer-set program update operators)**.**
We say a program update operator $\oplus$ respects support if the following conditions are satisfied for any programs $\mathcal{P}, \mathcal{Q}$, any answer set I of $\mathcal{P} \oplus \mathcal{Q}$ and any atom p:

1. *If $p \in I$, then $\mathcal{P} \cup \mathcal{Q}$ supports p in I.*
2. *If p appears only in the head of a single normal rule in $\mathcal{P}$ and nowhere in $\mathcal{Q}$, and p is supported by $\mathcal{P}$ in I, then $p \in I$.*

The first condition simply requires all atoms true in an answer set of $\mathcal{P} \oplus \mathcal{Q}$ to be supported by either $\mathcal{P}$ or $\mathcal{Q}$ in that same answer set. We believe this is an intuitive criterion for update operators as it reflects the static support property of most Logic Programming semantics.

The second condition requires every atom with only a single occurrence in a rule of $\mathcal{P}$ which also supports it in an answer set of $\mathcal{P} \oplus \mathcal{Q}$ to also be true in that answer set. Note that this second condition applies only to a few very specific cases. In certain scenarios, a stricter criterion may be desirable. However, the present condition captures a very basic case in which support should be inherited by inertia into the updated program.

The following theorem shows that reasonable update operators satisfying even only some of the most basic postulates *do not respect support*, i.e. they break at least one of the two above defined conditions. By *reasonable* we mean all those operators that always provide an answer set when updating a definite logic program by a consistent set of facts. This is indeed a natural condition to put on program update operators since definite logic programs are the most basic and best understood class of logic programs and they always have a unique least model (which coincides with the unique answer set), so an update by something as simple as a consistent set of facts should not cause the resulting program to be without an answer set.

Theorem 20. *Let $\oplus$ be a program update operator satisfying conditions (PU 1), (PU 3) and (PU 4) as well as the condition*

- *If $\mathcal{P}$ is a definite program and $\mathcal{Q}$ is a consistent set of facts, then $\mathcal{P} \oplus \mathcal{Q}$ has an answer set.*

Then $\oplus$ does not respect support.

Proof. Consider again the programs

$$\mathcal{P}_1 : \quad p. \qquad \mathcal{P}_2 : \quad p \leftarrow q. \qquad \mathcal{Q} : \quad \sim q.$$
$$q. \qquad\qquad q.$$

We have $[\![\mathcal{P}_1]\!]^{\mathsf{SE}} = [\![\mathcal{P}_2]\!]^{\mathsf{SE}} = \{\, \langle pq, pq \rangle \,\}$ and $[\![\mathcal{Q}]\!]^{\mathsf{SE}} = \{\, \langle \emptyset, \emptyset \rangle, \langle \emptyset, p \rangle, \langle p, p \rangle \,\}$. By (PU 4) we obtain that $\mathcal{P}_1 \oplus \mathcal{Q}$ is strongly equivalent to $\mathcal{P}_2 \oplus \mathcal{Q}$. Let $S = [\![\mathcal{P}_1 \oplus \mathcal{Q}]\!]^{\mathsf{SE}} = [\![\mathcal{P}_2 \oplus \mathcal{Q}]\!]^{\mathsf{SE}}$. S must be nonempty by (PU 3) and by (PU 1) it must be a subset of $[\![\mathcal{Q}]\!]^{\mathsf{SE}}$. Consequently, one of the following three cases must occur:

a) If $\langle \emptyset, \emptyset \rangle \in S$, then $\emptyset$ is an answer set of $\mathcal{P}_1 \oplus \mathcal{Q}$ in which p is false, though it has a single occurrence in $\mathcal{P}_1 \cup \mathcal{Q}$ and is supported by $\mathcal{P}_1$ in $\emptyset$. Thus, $\oplus$ does not respect support by the second condition of Def. 19.

b) If $S = \{\, \langle p, p \rangle \,\}$, then $\{\, p \,\}$ is an answer set of $\mathcal{P}_2 \oplus \mathcal{Q}$ in which p is true, though it is not supported by $\mathcal{P} \cup \mathcal{Q}$ in $\{\, p \,\}$. Consequently, $\oplus$ does not respect support by the first condition of Def. 19.

c) If $S = \{\, \langle \emptyset, p \rangle, \langle p, p \rangle \,\}$, then $\mathcal{P}_1 \oplus \mathcal{Q}$ has no answer set, contrary to the assumption of the theorem. □

The above theorem shows that a large class of reasonable answer-set program update operators based on SE models and Katsuno and

Mendelzon's approach to belief update will not respect support, as defined in Def. 19. We believe this is a major drawback of such operators, severely diminishing their applicability.

The problems we identified might be lifted if a richer semantic characterisation of logic programs was used instead of SE models. Such a characterisation would have to be able to distinguish between programs such as $\mathcal{P}_1 = \{\, p.\,, q.\, \}$ and $\mathcal{P}_2 = \{\, p \leftarrow q.\,, q.\, \}$ because they have different expected behaviour when subject to evolution.

Another alternative is to use the syntactic approaches to program updates [14, 2] that have matured over the years.

5 Conclusion

In this paper we revisited the problem of updates of answer-set programs, in an attempt to bring them in line with the more established class of belief updates that follow Katsuno and Mendelzon's postulates. Whereas until recently this was not possible since these postulates were simply not applicable (nor adaptable) when considering nonmonotonic Logic Programming semantics, as shown in [10], the introduction of *SE models* [20], which provide a monotonic characterisation of logic programs that is strictly more expressive than the answer-set semantics, provided a new opportunity to cast KM postulates into Logic Programming.

We adapted the KM postulates to be used for answer-set program updates and showed a representation theorem which provides a constructive characterisation of program update operators satisfying the postulates. This characterisation not only facilitates the investigation of these operators' properties, both semantic as well as computational, but it also provides an intuitive strategy for constructively defining these operators. This is one of the major contributions of the paper since it brings, for the first time, updates of answer-set programs in line with KM postulates. We illustrated this result with a definition of a specific answer-set program update operator which is a counterpart of Winslett's belief update operator.

The second important contribution of this paper is the uncovering of a serious drawback which extends to all meaningful answer-set program update operators based on SE models and the Katsuno and Mendelzon's approach to updates. All such operators violate a notion of *support*, which is a desirable property commonly enjoyed, in the static case, by all widely accepted Logic Programming semantics. This contribution is very important as it should guide further research on updates of answer-set programs either a) away from the semantic approach materialised in Katsuno and Mendelzon's postulates, or b) to the development of semantic characterisations of answer-set programs that are richer than SE models and appropriately capture their dynamic behaviour, or even c) turning back to the more syntactic approaches [14, 2] and see if they indeed offer a viable alternative.

Either way, updating answer-set programs is a very important theoretical and practical problem that is still waiting for a definite solution. This paper contains, we believe, a relevant contribution to its better understanding which will help guide future research.

ACKNOWLEDGEMENTS

REFERENCES

[1] Carlos E. Alchourrón, Peter Gärdenfors, and David Makinson, 'On the logic of theory change: Partial meet contraction and revision functions', *Journal of Symbolic Logic*, **50**(2), 510–530, (1985).

[2] José Júlio Alferes, Federico Banti, Antonio Brogi, and João Alexandre Leite, 'The refined extension principle for semantics of dynamic logic programming', *Studia Logica*, **79**(1), 7–32, (2005).

[3] José Júlio Alferes, João Alexandre Leite, Luís Moniz Pereira, Halina Przymusinska, and Teodor C. Przymusinski, 'Dynamic updates of nonmonotonic knowledge bases', *The Journal of Logic Programming*, **45**(1-3), 43–70, (2000).

[4] Krzysztof R. Apt, Howard A. Blair, and Adrian Walker, 'Towards a theory of declarative knowledge', in *Foundations of Deductive Databases and Logic Programming.*, 89–148, Morgan Kaufmann, (1988).

[5] Chitta Baral, *Knowledge Representation, Reasoning, and Declarative Problem Solving*, Cambridge University Press, 2003.

[6] Pedro Cabalar and Paolo Ferraris, 'Propositional theories are strongly equivalent to logic programs', *Theory and Practice of Logic Programming (TPLP)*, **7**(6), 745–759, (2007).

[7] James P. Delgrande, Torsten Schaub, Hans Tompits, and Stefan Woltran, 'Belief revision of logic programs under answer set semantics', in *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008)*, eds., Gerhard Brewka and Jérôme Lang, pp. 411–421, Sydney, Australia, (2008). AAAI Press.

[8] Jürgen Dix, 'A classification theory of semantics of normal logic programs: I. Strong properties', *Fundamenta Informaticae*, **22**(3), 227–255, (1995).

[9] Jürgen Dix, 'A classification theory of semantics of normal logic programs: II. Weak properties', *Fundamenta Informaticae*, **22**(3), 257–288, (1995).

[10] Thomas Eiter, Michael Fink, Giuliana Sabbatini, and Hans Tompits, 'On properties of update sequences based on causal rejection', *Theory and Practice of Logic Programming (TPLP)*, **2**(6), 721–777, (2002).

[11] Michael Gelfond and Vladimir Lifschitz, 'The stable model semantics for logic programming', in *Proceedings of the 5th International Conference and Symposium on Logic Programming (ICLP/SLP 1988)*, eds., Robert A. Kowalski and Kenneth A. Bowen, pp. 1070–1080, Seattle, Washington, (1988). MIT Press.

[12] Andreas Herzig and Omar Rifi, 'Propositional belief base update and minimal change', *Artificial Intelligence*, **115**(1), 107–138, (1999).

[13] Hirofumi Katsuno and Alberto O. Mendelzon, 'On the difference between updating a knowledge base and revising it', in *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91)*, eds., James F. Allen, Richard Fikes, and Erik Sandewall, pp. 387–394, Cambridge, MA, USA, (1991). Morgan Kaufmann Publishers.

[14] João Alexandre Leite, *Evolving Knowledge Bases*, volume 81 of *Frontiers of Artificial Intelligence and Applications, xviii + 307 p. Hardcover*, IOS Press, 2003.

[15] João Alexandre Leite and Luís Moniz Pereira, 'Generalizing updates: From models to programs', in *Proceedings of the 3rd International Workshop on Logic Programming and Knowledge Representation (LPKR '97)*, eds., Jürgen Dix, Luís Moniz Pereira, and Teodor C. Przymusinski, volume 1471 of *Lecture Notes in Computer Science*, pp. 224–246, Port Jefferson, New York, USA, (1997). Springer.

[16] Vladimir Lifschitz, David Pearce, and Agustín Valverde, 'Strongly equivalent logic programs', *ACM Transactions on Computational Logic (TOCL)*, **2**(4), 526–541, (2001).

[17] Viktor W. Marek and Miroslaw Truszczynski, 'Revision programming', *Theoretical Computer Science*, **190**(2), 241–277, (1998).

[18] Chiaki Sakama and Katsumi Inoue, 'An abductive framework for computing knowledge base updates', *Theory and Practice of Logic Programming (TPLP)*, **3**(6), 671713, (2003).

[19] Ján Šefránek, 'Irrelevant updates and nonmonotonic assumptions', in *Proceedings of the 10th European Conference on Logics in Artificial Intelligence (JELIA 2006)*, eds., Michael Fisher, Wiebe van der Hoek, Boris Konev, and Alexei Lisitsa, volume 4160 of *Lecture Notes in Computer Science*, pp. 426–438, Liverpool, UK, (2006). Springer.

[20] Hudson Turner, 'Strong equivalence made easy: nested expressions and weight constraints', *Theory and Practice of Logic Programming (TPLP)*, **3**(4-5), 609–622, (2003).

[21] Marianne Winslett, *Updating Logical Databases*, Cambridge University Press, New York, NY, USA, 1990.

[22] Yan Zhang and Norman Y. Foo, 'A unified framework for representing logic program updates', in *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, eds., Manuela M. Veloso and Subbarao Kambhampati, pp. 707–713, Pittsburgh, USA, (2005). AAAI.