On Supporting Strong and Default Negation in Answer-Set Program Updates

Martin Slota[1], Martin Baláž[2], and João Leite[1](✉)

[1] CENTRIA and Departamento de Informática, Universidade Nova de Lisboa, Lisbon, Portugal
[2] Faculty of Mathematics, Physics and Informatics, Comenius University, Bratislava, Slovakia
jleite@fct.unl.pt

Abstract. Existing semantics for answer-set program updates fall into two categories: either they consider only *strong negation* in heads of rules, or they primarily rely on *default negation* in heads of rules and optionally provide support for strong negation by means of a syntactic transformation.

In this paper we pinpoint the limitations of both these approaches and argue that both types of negation should be first-class citizens in the context of updates. We identify principles that plausibly constrain their interaction but are not simultaneously satisfied by any existing rule update semantics. Then we extend one of the most advanced semantics with direct support for strong negation and show that it satisfies the outlined principles as well as a variety of other desirable properties.

1 Introduction

The increasingly common use of rule-based knowledge representation languages in highly dynamic and information-rich contexts, such as the Semantic Web [2], requires standardised support for updates of knowledge represented by rules. Answer-set programming [3,4] forms the natural basis for investigation of rule updates, and various approaches to answer-set program updates have been explored throughout the last 15 years [5–18].

The most straightforward kind of conflict arising between an original rule and its update occurs when the original conclusion logically contradicts the newer one. Though the technical realisation and final result may differ significantly, depending on the particular rule update semantics, this kind of conflict is resolved by letting the newer rule prevail over the older one. Actually, under most semantics, this is also the *only* type of conflict that is subject to automatic resolution [5,7,8,11,12,15,16].

From this perspective, allowing for both *strong* and *default negation* to appear in heads of rules is essential for an expressive and universal rule update framework

J. Leite and M. Slota were partially supported by FCT under project PTDC/EIA-CCO/121823/2010. The collaboration between the co-authors resulted from the Slovak–Portuguese bilateral project supported by APVV agency under SK-PT-0028-10 and by FCT under FCT/2487/3/6/2011/S.

A preliminary version of this paper appeared as [1]. An extended version of this paper with all the proofs is available at http://arxiv.org/abs/1404.6784

A.L.C. Bazzan and K. Pichara (Eds.): IBERAMIA 2014, LNAI 8864, pp. 41–53, 2014.
DOI: 10.1007/978-3-319-12027-0_4

[9]. While strong negation is the natural candidate here, used to express that an atom *becomes explicitly false*, default negation allows for more fine-grained control: the atom only *ceases to be true*, but its truth value may not be known after the update. The latter also makes it possible to move between any pair of epistemic states by means of updates, as illustrated in the following example:

Example 1 (Railway crossing [9]). Take the following logic program used to choose an action at a railway crossing (where $\neg$ (reps. $\sim$) denotes strong (resp. default) negation):

$$\mathsf{cross} \leftarrow \neg\mathsf{train}. \qquad \mathsf{wait} \leftarrow \mathsf{train}. \qquad \mathsf{listen} \leftarrow {\sim}\mathsf{train}, {\sim}\neg\mathsf{train}.$$

The intuitive meaning of these rules is as follows: one should cross if there is evidence that no train is approaching; wait if there is evidence that a train is approaching; listen if there is no such evidence. Consider a situation where a train is approaching, represented by the fact ($\mathsf{train}.$). After this train has passed by, we want to update our knowledge to an epistemic state where we lack evidence with regard to the approach of a train. If this was accomplished by updating with the fact ($\neg\mathsf{train}.$), we would cross the tracks at the subsequent state, risking being killed by another train that was approaching. Therefore, we need to express an update stating that all past evidence for an atom is to be removed, which can be accomplished by allowing default negation in heads of rules. In this scenario, the intended update can be expressed by the fact (${\sim}\mathsf{train}.$).

Concerning the support of negation in rule heads, existing rule update semantics fall into two categories: those that only allow for strong negation, and those that primarily consider default negation. As illustrated above, the former are unsatisfactory as they render many belief states unreachable by updates. As for the latter, they optionally provide support for strong negation by means of a syntactic transformation. Two such transformations are known, both based on the principle of coherence: if an atom p is true, its strong negation $\neg p$ cannot be true simultaneously, so ${\sim}\neg p$ must be true, and also vice versa, if $\neg p$ is true, then so is ${\sim}p$. The first transformation, introduced in [19], encodes this principle directly by adding, to both the original program and its update, the following two rules for every atom p: ${\sim}\neg p \leftarrow p.$ and ${\sim}p \leftarrow \neg p.$ This way, every conflict between an atom p and its strong negation $\neg p$ directly translates into two conflicts between the objective literals p, $\neg p$ and their default negations. However, the added rules lead to undesired side effects that stand in direct opposition with basic principles underlying updates. Specifically, despite the fact that the empty program does not encode any change in the modelled world, the stable models assigned to a program may change after an update by the empty program. This undesired behaviour is addressed in an alternative transformation from [9] that encodes the coherence principle more carefully. Nevertheless, this transformation also leads to undesired consequences, as demonstrated in the following example:

Example 2 (Faulty sensor). Suppose that we collect data from sensors and multiple sensors are used to supply information about the critical fluent p. In case of a malfunction of one of the sensors, we may end up with an inconsistent logic program consisting of the following two facts: $p.$ and $\neg p.$ At this point, no stable model of the program exists. If a problem is found in the sensor that supplied the first fact ($p.$), after the sensor is repaired, this information needs to be reset by updating the program with the fact (${\sim}p.$). Following the common pattern in rule updates, where recovery from conflicting states is always possible, this update should be sufficient to assign a stable model to the updated program. However, the transformational semantics for strong negation of [9] still does not provide any stable model – we remain without a valid epistemic state when one should exist.

In this paper we address the combination of strong and default negation in the context of rule updates. We formulate a generic desirable principle that is violated by the existing approaches. Then we show how two distinct definitions of one of the most well-behaved rule update semantics [11,12] can be equivalently extended with support for strong negation while satisfying the formulated principle and retaining the formal and computational properties of the original semantics. Our main contributions are as follows: *a*) based on Example 2, we introduce the *early recovery principle* that captures circumstances under which a stable model after a rule update should exist; *b*) we extend the *well-supported semantics for rule updates* [12] with direct support for strong negation; *c*) we define a fixpoint characterisation of the new semantics, based on the *refined dynamic stable model* semantics for rule updates [11]; *d*) we show that the defined semantics enjoy the early recovery principle as well as a range of desirable properties for rule updates known from the literature.

This paper is organised as follows: In Sect. 2 we present logic programs, generalise the well-supported semantics from the class of normal programs to extended ones and define the rule update semantics from [11,12]. In Sect. 3, we establish the early recovery principle, define the new rule update semantics for strong negation and show that it satisfies the principle. In Sect. 4 we introduce other established rule update principles and show that the proposed semantics satisfies them. We discuss our findings and conclude in Sect. 5.

2 Background

Logic Programs. We assume that a countable set of propositional atoms $\mathcal{A}$ is given and fixed. An *objective literal* is an atom $p \in \mathcal{A}$ or its strong negation $\neg p$. We denote the set of all objective literals by $\mathcal{L}$. A *default literal* is an objective literal preceded by $\sim$ denoting default negation. A *literal* is either an objective or a default literal. We denote the set of all literals by $\mathcal{L}^*$. As a convention, double negation is absorbed, so that $\neg\neg p$ denotes the atom p and $\sim\sim l$ denotes the objective literal l. Given a set of literals S, we introduce the following notation: $S^+ = \{ l \in \mathcal{L} \mid l \in S \}$, $S^- = \{ l \in \mathcal{L} \mid \sim l \in S \}$, $\sim S = \{ \sim L \mid L \in S \}$. An *extended rule* is a pair $\pi = (\mathsf{H}_\pi, \mathsf{B}_\pi)$ where H_π is a literal, referred to as the *head of* π, and B_π is a finite set of literals, referred to as the *body of* π. Usually we write π as $(\mathsf{H}_\pi \leftarrow \mathsf{B}_\pi^+, \sim\mathsf{B}_\pi^-.)$. A *generalised rule* is an extended rule that contains no occurrence of $\neg$, i.e., its head and body consist only of atoms and their default negations. A *normal rule* is a generalised rule that has an atom in the head. A *fact* is an extended rule whose body is empty and a *tautology* is any extended rule π such that $\mathsf{H}_\pi \in \mathsf{B}_\pi$. An *extended (generalised, normal) program* is a set of extended (generalised, normal) rules. An *interpretation* is a consistent subset of the set of objective literals, i.e., a subset of $\mathcal{L}$ not containing both p and $\neg p$ for any atom p. The satisfaction of an objective literal l, default literal $\sim l$, set of literals S, extended rule π and extended program P in an interpretation J is defined as usual: $J \models l$ iff $l \in J$; $J \models {\sim}l$ iff $l \notin J$; $J \models S$ iff $J \models L$ for all $L \in S$; $J \models \pi$ iff $J \models \mathsf{B}_\pi$ implies $J \models \mathsf{H}_\pi$; $J \models P$ iff $J \models \pi$ for all $\pi \in P$. Also, J is a *model of P* if $J \models P$, and P is *consistent* if it has a model.

Definition 1 (Stable model). *Let P be an extended program. The set* $[\![P]\!]_{\mathsf{SM}}$ *of* stable models of P *consists of all interpretations J such that* $J^* = \mathsf{least}(P \cup \mathsf{def}(J))$ *where* $\mathsf{def}(J) = \{\, {\sim}l. \mid l \in \mathcal{L} \setminus J \,\}$, $J^* = J \cup {\sim}(\mathcal{L} \setminus J)$ *and* $\mathsf{least}(\cdot)$ *denotes the least model of the argument program with all literals treated as propositional atoms.*

A *level mapping* ℓ is a function that maps every atom to a natural number. Also, for any default literal ${\sim}p$, where $p \in \mathcal{A}$, and finite set of atoms and their default negations S, $\ell({\sim}p) = \ell(p)$, $\ell^{\downarrow}(S) = \min\{\, \ell(L) \mid L \in S \,\}$ and $\ell^{\uparrow}(S) = \max\{\, \ell(L) \mid L \in S \,\}$.

Definition 2 (Well-supported model of a normal program). *Let P be a normal program and ℓ a level mapping. An interpretation $J \subseteq \mathcal{A}$ is a* well-supported model of P w.r.t. ℓ *if the following conditions are satisfied: 1. J is a model of P and 2. For every atom $p \in J$ there exists a rule $\pi \in P$ such that* $\mathsf{H}_\pi = p \wedge J \models \mathsf{B}_\pi \wedge \ell(\mathsf{H}_\pi) > \ell^{\uparrow}(\mathsf{B}_\pi)$. *The set* $[\![P]\!]_{\mathsf{WS}}$ *of* well-supported models of P *consists of all interpretations $J \subseteq \mathcal{A}$ such that J is a well-supported model of P w.r.t. some level mapping.*

Proposition 1 ([20]). *Let P be a normal program. Then,* $[\![P]\!]_{\mathsf{WS}} = [\![P]\!]_{\mathsf{SM}}$.

Well-Supported Models for Extended Programs. The well-supported models for normal logic programs can be generalised in a straightforward manner to deal with strong negation while maintaining their tight relationship with stable models (c.f. Proposition 1). This will come useful when we discuss adding support for strong negation to semantics for rule updates. We extend level mappings from atoms and their default negations to all literals: An *(extended) level mapping* ℓ maps every objective literal to a natural number. Also, for any default literal ${\sim}l$ and finite set of literals S, $\ell({\sim}l) = \ell(p)$, $\ell^{\downarrow}(S) = \min\{\, \ell(L) \mid L \in S \,\}$ and $\ell^{\uparrow}(S) = \max\{\, \ell(L) \mid L \in S \,\}$.

Definition 3 (Well-supported model of an extended program). *Let P be an extended program and ℓ a level mapping. An interpretation J is a* well-supported model of P w.r.t. ℓ *if the following conditions are satisfied: 1. J is a model of P and 2. For every objective literal $l \in J$ there exists a rule $\pi \in P$ such that* $\mathsf{H}_\pi = l \wedge J \models \mathsf{B}_\pi \wedge \ell(\mathsf{H}_\pi) > \ell^{\uparrow}(\mathsf{B}_\pi)$. *The set* $[\![P]\!]_{\mathsf{WS}}$ *of* well-supported models of P *consists of all interpretations J such that J is a well-supported model of P w.r.t. some level mapping.*

Proposition 2. *Let P be an extended program. Then,* $[\![P]\!]_{\mathsf{WS}} = [\![P]\!]_{\mathsf{SM}}$.

Rule Updates. Rule update semantics assign stable models to a pair or sequence of programs where each component represents an update of the preceding ones. Formally, a *dynamic logic program* (DLP) is a finite sequence of extended programs and by $\mathsf{all}(\boldsymbol{P})$ we denote the multiset of all rules in the components of $\boldsymbol{P}$. A rule update semantics S assigns a *set of S-models*, denoted by $[\![\boldsymbol{P}]\!]_{\mathsf{S}}$, to $\boldsymbol{P}$.

We focus on semantics based on the causal rejection principle [5,7–9,11,12,16] which states that a rule is *rejected* if it is in a direct conflict with a more recent rule. The basic conflict between rules π and σ occurs when their heads are complementary, i.e. when $\mathsf{H}_\pi = {\sim}\mathsf{H}_\sigma$. Based on such conflicts and on a stable model candidate, a *set of rejected rules* can be determined and it can be verified that the candidate is indeed stable w.r.t. the remaining rules.

We define the most mature of these semantics, providing two equivalent definitions: the *refined dynamic stable models* [11], or *RD-semantics*, defined using a fixpoint equation, and the *well-supported models* [12], or *WS-semantics*, based on level mappings.

Definition 4 (RD-semantics [11]). *Let $\boldsymbol{P} = \langle P_i \rangle_{i<n}$ be a DLP without strong negation. Given an interpretation J, the multisets of rejected rules $\mathsf{rej}_{\geq}(\boldsymbol{P}, J)$ and of default assumptions $\mathsf{def}(\boldsymbol{P}, J)$ are defined as follows:*

$$\mathsf{rej}_{\geq}(\boldsymbol{P}, J) = \{\, \pi \in P_i \mid i < n \wedge \exists j \geq i\, \exists \sigma \in P_j : \mathsf{H}_\pi = {\sim}\mathsf{H}_\sigma \wedge J \models \mathsf{B}_\sigma \,\} \quad ,$$
$$\mathsf{def}(\boldsymbol{P}, J) = \{\, ({\sim}l.) \mid l \in \mathcal{L} \wedge \neg(\exists \pi \in \mathsf{all}(\boldsymbol{P}) : \mathsf{H}_\pi = l \wedge J \models \mathsf{B}_\pi) \,\} \quad .$$

Let J^* and $\mathsf{least}(\cdot)$ be defined as before. The set $[\![\boldsymbol{P}]\!]_{\mathsf{RD}}$ of RD-models of $\boldsymbol{P}$ *consists of all interpretations J such that $J^* = \mathsf{least}\left([\mathsf{all}(\boldsymbol{P}) \setminus \mathsf{rej}_{\geq}(\boldsymbol{P}, J)] \cup \mathsf{def}(\boldsymbol{P}, J)\right)$.*

Definition 5 (WS-semantics [12]). *Let $\boldsymbol{P} = \langle P_i \rangle_{i<n}$ be a DLP without strong negation. Given an interpretation J and a level mapping ℓ, the multiset of rejected rules $\mathsf{rej}_{\ell}(\boldsymbol{P}, J)$ is defined as follows:*

$$\mathsf{rej}_{\ell}(\boldsymbol{P}, J) = \{\, \pi \in P_i \mid i < n \wedge \exists j > i\, \exists \sigma \in P_j : \mathsf{H}_\pi = {\sim}\mathsf{H}_\sigma \wedge J \models \mathsf{B}_\sigma \wedge \ell(\mathsf{H}_\sigma) > \ell^{\uparrow}(\mathsf{B}_\sigma) \,\} \quad .$$

The set $[\![\boldsymbol{P}]\!]_{\mathsf{WS}}$ of WS-models of $\boldsymbol{P}$ *consists of all interpretations J such that for some level mapping ℓ, the following conditions are satisfied: 1. J is a model of $\mathsf{all}(\boldsymbol{P}) \setminus \mathsf{rej}_{\ell}(\boldsymbol{P}, J)$ and 2. For every $l \in J$ there exists some rule $\pi \in \mathsf{all}(\boldsymbol{P}) \setminus \mathsf{rej}_{\ell}(\boldsymbol{P}, J)$ such that $\mathsf{H}_\pi = l \wedge J \models \mathsf{B}_\pi \wedge \ell(\mathsf{H}_\pi) > \ell^{\uparrow}(\mathsf{B}_\pi)$.*

Unlike most other rule update semantics, these semantics can properly deal with tautological and other irrelevant updates, as illustrated in the following example:

Example 3 (Irrelevant updates). Consider the DLP $\boldsymbol{P} = \langle P, U \rangle$ where programs P, U are as follows: $P = \{\, \mathsf{day} \leftarrow {\sim}\mathsf{night}., \mathsf{night} \leftarrow {\sim}\mathsf{day}., \mathsf{stars} \leftarrow \mathsf{night}, {\sim}\mathsf{cloudy}., {\sim}\mathsf{stars}. \,\}$ and $U = \{\, \mathsf{stars} \leftarrow \mathsf{stars}. \,\}$. Program P has the single stable model $J_1 = \{\, \mathsf{day} \,\}$ and U contains a single tautological rule, i.e. it does not encode any change in the modelled domain. Thus, we expect that $\boldsymbol{P}$ also has the single stable model J_1. However, many rule update semantics, such as those introduced in [5,7–10,13,15,16,18], are sensitive to this or other tautological updates, introducing or eliminating models of the original program. In this case, the unwanted model candidate is $J_2 = \{\, \mathsf{night}, \mathsf{stars} \,\}$ and it is neither an RD- nor a WS-model of $\boldsymbol{P}$, though the reasons for this are technically different under these two semantics. It is not difficult to verify that, given an arbitrary level mapping ℓ, the set of default assumptions and the respective sets of rejected rules are as follows: $\mathsf{def}(\boldsymbol{P}, J_2) = \{\, ({\sim}\mathsf{cloudy}.), ({\sim}\mathsf{day}.) \,\}$, $\mathsf{rej}_{\geq}(\boldsymbol{P}, J_2) = \{(\mathsf{stars} \leftarrow \mathsf{night}, {\sim}\mathsf{cloudy}.), ({\sim}\mathsf{stars}.)\}$, and $\mathsf{rej}_{\ell}(\boldsymbol{P}, J_2) = \emptyset$. Note that $\mathsf{rej}_{\ell}(\boldsymbol{P}, J_2)$ is empty because, independently of ℓ, no rule π in U satisfies the condition $\ell(\mathsf{H}_\pi) > \ell^{\uparrow}(\mathsf{B}_\pi)$, so there is no rule that could reject another rule. Thus, the atom stars belongs to J_2^* but does not belong to $\mathsf{least}([\mathsf{all}(\boldsymbol{P}) \setminus \mathsf{rej}_{\geq}(\boldsymbol{P}, J_2)] \cup \mathsf{def}(\boldsymbol{P}, J_2))$, so J_2 is not an RD-model of $\boldsymbol{P}$. Furthermore, no model of $\mathsf{all}(\boldsymbol{P}) \setminus \mathsf{rej}_{\ell}(\boldsymbol{P}, J_2)$ contains stars, so J_2 cannot be a WS-model of $\boldsymbol{P}$. Furthermore, the resilience of RD- and WS-semantics is not limited to empty and tautological updates, but extends to other irrelevant updates as well [11,12]. For example, consider the DLP $\boldsymbol{P}' = \langle P, U' \rangle$ where $U' = \{ (\mathsf{stars} \leftarrow \mathsf{venus}.), (\mathsf{venus} \leftarrow \mathsf{stars}.) \}$. Though the updating program contains non-tautological rules, it does not provide a bottom-up justification of any model other than J_1 and, indeed, J_1 is the only RD- and WS-model of $\boldsymbol{P}'$.

We also note that the two presented semantics for DLPs without strong negation provide the same result regardless of the particular DLP to which they are applied.

Proposition 3 ([12]). *Let $\boldsymbol{P}$ be a DLP without strong negation. Then,* $[\![\boldsymbol{P}]\!]_{\mathsf{WS}} = [\![\boldsymbol{P}]\!]_{\mathsf{RD}}$.

When dealing with a single program, strong negation can be reduced away by treating all objective literals as atoms and adding, for each atom p, the integrity constraint $(\leftarrow p, \neg p.)$ to the program [4]. However, this transformation does not serve its purpose when adding support for strong negation to causal rejection semantics for DLPs because integrity constraints have empty heads, so according to these rule update semantics, they cannot be used to reject any other rule. For example, a DLP such as $\langle \{ p., \neg p. \}, \{ p. \} \rangle$ would remain without a stable model even though the DLP $\langle \{ p., \sim p. \}, \{ p. \} \rangle$ does have a stable model. To capture the conflict between opposite objective literals l and $\neg l$ in a way that is compatible with causal rejection semantics, a slightly modified syntactic transformation can be performed, translating such conflicts into conflicts between objective literals and their default negations. Two such transformations have been suggested in the literature [9,19], both based on the principle of coherence. For any extended program P and DLP $\boldsymbol{P} = \langle P_i \rangle_{i<n}$ they are defined as follows:

$$P^{\dagger} = P \cup \{ \sim\neg l \leftarrow l. \mid l \in \mathcal{L} \}, \quad \boldsymbol{P}^{\dagger} = \langle P_i^{\dagger} \rangle_{i<n},$$
$$P^{\ddagger} = P \cup \{ \sim\neg \mathsf{H}_{\pi} \leftarrow \mathsf{B}_{\pi}. \mid \pi \in P \land \mathsf{H}_{\pi} \in \mathcal{L} \}, \quad \boldsymbol{P}^{\ddagger} = \langle P_i^{\ddagger} \rangle_{i<n}.$$

These transformations lead to four possibilities for defining the semantics of a DLP $\boldsymbol{P}$: $[\![\boldsymbol{P}^{\dagger}]\!]_{\mathsf{RD}}$, $[\![\boldsymbol{P}^{\ddagger}]\!]_{\mathsf{RD}}$, $[\![\boldsymbol{P}^{\dagger}]\!]_{\mathsf{WS}}$ and $[\![\boldsymbol{P}^{\ddagger}]\!]_{\mathsf{WS}}$. We discuss these in the subsequent section.

3 Direct Support for Strong Negation in Rule Updates

The problem with existing semantics for strong negation in rule updates is that those based on the first transformation ($\boldsymbol{P}^{\dagger}$) sometimes assign too many models, while those based on the second ($\boldsymbol{P}^{\ddagger}$) sometimes do not assign any model when one should exist.

Example 4 (Undesired side effects of the first transformation). Consider the DLP $\boldsymbol{P}_1 = \langle P, U \rangle$ where $P = \{ p., \neg p. \}$ and $U = \emptyset$. Since P has no stable model and U does not encode any change in the represented domain, it should follow that $\boldsymbol{P}_1$ has no stable model either. However, $[\![\boldsymbol{P}_1^{\dagger}]\!]_{\mathsf{RD}} = [\![\boldsymbol{P}_1^{\dagger}]\!]_{\mathsf{WS}} = \{ \{ p \}, \{ \neg p \} \}$, i.e. two models are assigned to $\boldsymbol{P}_1$ when using the first transformation to add support for strong negation. To verify this, observe that $\boldsymbol{P}_1^{\dagger} = \langle P^{\dagger}, U^{\dagger} \rangle$ where $P^{\dagger} = \{ p., \neg p., \sim p \leftarrow \neg p., \sim\neg p \leftarrow p. \}$ and $U^{\dagger} = \{ \sim p \leftarrow \neg p., \sim\neg p \leftarrow p. \}$. Consider $J_1 = \{ p \}$. Then, we have $\mathsf{rej}_{\geq}(\boldsymbol{P}_1^{\dagger}, J_1) = \{ \neg p., \sim\neg p \leftarrow p. \}$ and $\mathsf{def}(\boldsymbol{P}_1^{\dagger}, J_1) = \emptyset$, so it follows that $\mathit{least}([\mathsf{all}(\boldsymbol{P}_1^{\dagger}) \setminus \mathsf{rej}_{\geq}(\boldsymbol{P}_1^{\dagger}, J_1)] \cup$

$\mathsf{def}(\boldsymbol{P}_1^{\dagger}, J_1)) = \{\, p, \sim\neg p \,\} = J_1^*$. In other words, J_1 belongs to $[\![\boldsymbol{P}_1^{\dagger}]\!]_{\mathsf{RD}}$ and in an analogous fashion it can be verified that $J_2 = \{\, \neg p \,\}$ also belongs there. A similar situation occurs with $[\![\boldsymbol{P}_1^{\dagger}]\!]_{\mathsf{WS}}$ since the rules that were added to the more recent program can be used to reject facts in the older one.

Thus, the problem with the first transformation is that an update by an empty program, which does not express any change in the represented domain, may affect the original semantics. This behaviour goes against basic and intuitive principles underlying updates, grounded already in the classical belief update postulates [21,22] and satisfied by virtually all belief update operations [23] as well as by the vast majority of existing rule update semantics, including the original RD- and WS-semantics.

This undesired behaviour can be corrected by using the second transformation. The more technical reason is that it does not add any rules to a program in the sequence unless that program already contains some original rules. However, its use leads to another problem: sometimes *no model* is assigned when in fact a model should exist.

Example 5 (Undesired side effects of the second transformation). Consider Example 2, formalised as DLP $\boldsymbol{P}_2 = \langle P, V \rangle$ where $P = \{\, p., \neg p. \,\}$ and $V = \{\, \sim p. \,\}$. One expects that since V resolves the conflict present in P, a stable model should be assigned to $\boldsymbol{P}_2$. However, $[\![\boldsymbol{P}_2^{\ddagger}]\!]_{\mathsf{RD}} = [\![\boldsymbol{P}_2^{\ddagger}]\!]_{\mathsf{WS}} = \emptyset$. To verify this, observe that $\boldsymbol{P}_2^{\ddagger} = \langle P^{\ddagger}, V^{\ddagger} \rangle$ where $P^{\ddagger} = \{\, p., \neg p., \sim p., \sim\neg p. \,\}$ and $V^{\ddagger} = \{\, \sim p. \,\}$. Given an interpretation J and level mapping ℓ, we conclude that $\mathsf{rej}_{\ell}(\boldsymbol{P}_2^{\ddagger}, J) = \{\, p. \,\}$, so the facts $(\neg p.)$ and $(\sim\neg p.)$ both belong to the program $\mathsf{all}(\boldsymbol{P}_2^{\ddagger}) \setminus \mathsf{rej}_{\ell}(\boldsymbol{P}_2^{\ddagger}, J)$. Consequently, this program has no model and it follows that J cannot belong to $[\![\boldsymbol{P}_2^{\ddagger}]\!]_{\mathsf{WS}}$. Similarly it can be shown that $[\![\boldsymbol{P}_2^{\ddagger}]\!]_{\mathsf{RD}} = \emptyset$.

Based on this example, in the following we formulate a generic *early recovery principle* that formally identifies conditions under which *some* stable model should be assigned to a DLP. For the sake of simplicity, we concentrate on DLPs of length 2 which are composed of facts. We discuss a generalisation of the principle to DLPs of arbitrary length and containing other rules than just facts in Sect. 5. After introducing the principle, we define a semantics for rule updates which directly supports both strong and default negation and satisfies the principle.

We begin by defining, for every objective literal l, the sets of literals $\overline{l} = \{\, \sim l, \neg l \,\}$ and $\overline{\sim l} = \{\, l \,\}$. Intuitively, for every literal L, $\overline{L}$ denotes the set of literals that are in conflict with L. Furthermore, given two sets of facts P and U, we say that U *solves all conflicts in* P if for each pair of rules $\pi, \sigma \in P$ such that $\mathsf{H}_{\sigma} \in \overline{\mathsf{H}_{\pi}}$ there is a fact $\rho \in U$ such that either $\mathsf{H}_{\rho} \in \overline{\mathsf{H}_{\pi}}$ or $\mathsf{H}_{\rho} \in \overline{\mathsf{H}_{\sigma}}$.

Considering a rule update semantics S, the new principle simply requires that when U solves all conflicts in P, S will assign *some model* to $\langle P, U \rangle$. Formally:

Early recovery principle: If P is a set of facts and U is a consistent set of facts that solves all conflicts in P, then $[\![\, \langle P, U \rangle \,]\!]_{\mathsf{S}} \neq \emptyset$.

We conjecture that rule update semantics should generally satisfy the above principle. In contrast with the usual behaviour of belief update operators, the nature of existing rule update semantics ensures that recovery from conflict is always possible, and this principle simply formalises and sharpens the sufficient conditions for such recovery.

Our next goal is to define a semantics for rule updates that not only satisfies the outlined principle, but also enjoys other established properties of rule updates identified over the years. As for the original semantics for rule updates, we provide two equivalent definitions, one based on a fixed point equation and the other one on level mappings.

To directly accommodate strong negation in the RD-semantics, we first need to look more closely at the set of rejected rules $\mathsf{rej}_{\geq}(\boldsymbol{P}, J)$, particularly at the fact that it allows conflicting rules within the same component of $\boldsymbol{P}$ to reject one another. This behaviour, along with the constrained set of defaults $\mathsf{def}(\boldsymbol{P}, J)$, is used to prevent tautological and other irrelevant cyclic updates from affecting the semantics. However, in the presence of strong negation, rejecting conflicting rules within the same program has undesired side effects. For example, the early recovery principle requires that some model be assigned to the DLP $\langle \{ p., \neg p. \}, \{ \sim p \} \rangle$ from Example 5, but if the rules in the initial program reject each other, then the only possible stable model to assign is $\emptyset$. However, such a stable model would violate the causal rejection principle since it does not satisfy the initial rule $(\neg p.)$ and there is no rule in the updating program that overrides it.

To overcome the limitations of this approach to the prevention of tautological updates, we disentangle rule rejection per se from ensuring that rejection is done without cyclic justifications. We introduce the set of rejected rules $\mathsf{rej}^{\neg}_{>}(\boldsymbol{P}, S)$ which directly supports strong negation and does not allow for rejection within the same program. Prevention of cyclic rejections is done separately by using a customised immediate consequence operator $T_{\boldsymbol{P},J}$. Given a stable model candidate J, instead of verifying that J^* is the least fixed point of the usual consequence operator, as done in the RD-semantics using $\mathsf{least}(\cdot)$, we verify that J^* is the least fixed point of $T_{\boldsymbol{P},J}$.

Definition 6 (Extended RD-semantics). *Let $\boldsymbol{P} = \langle P_i \rangle_{i<n}$ be a DLP. For an interpretation J and set of literals S, the multiset of rejected rules* $\mathsf{rej}^{\neg}_{>}(\boldsymbol{P}, S)$*, the remainder* $\mathsf{rem}(\boldsymbol{P}, S)$ *and the consequence operator $T_{\boldsymbol{P},J}$ are defined as follows:*

$$\begin{aligned}
&\mathsf{rej}^{\neg}_{>}(\boldsymbol{P}, S) = \left\{ \pi \in P_i \;\middle|\; i < n \wedge \exists j > i \, \exists \sigma \in P_j : \mathsf{H}_\sigma \in \overline{\mathsf{H}_\pi} \wedge \mathsf{B}_\sigma \subseteq S \right\}, \\
&\mathsf{rem}(\boldsymbol{P}, S) = \mathsf{all}(\boldsymbol{P}) \setminus \mathsf{rej}^{\neg}_{>}(\boldsymbol{P}, S), \\
&T_{\boldsymbol{P},J}(S) = \big\{ \mathsf{H}_\pi \mid \pi \in (\mathsf{rem}(\boldsymbol{P}, J^*) \cup \mathsf{def}(J)) \\
&\qquad\qquad \wedge \mathsf{B}_\pi \subseteq S \wedge \neg \left(\exists \sigma \in \mathsf{rem}(\boldsymbol{P}, S) : \mathsf{H}_\sigma \in \overline{\mathsf{H}_\pi} \wedge \mathsf{B}_\sigma \subseteq J^* \right) \big\}.
\end{aligned}$$

Furthermore, $T^{0}_{\boldsymbol{P},J}(S) = S$ and for every $k \geq 0$, $T^{k+1}_{\boldsymbol{P},J}(S) = T_{\boldsymbol{P},J}(T^{k}_{\boldsymbol{P},J}(S))$. The set $[\![\boldsymbol{P}]\!]^{\neg}_{\mathsf{RD}}$ of extended RD-models of $\boldsymbol{P}$ *consists of all interpretations J such that $J^* = \bigcup_{k \geq 0} T^{k}_{\boldsymbol{P},J}(\emptyset)$.*

Adding support for strong negation to the WS-semantics is done by modifying the set of rejected rules $\mathsf{rej}_{\ell}(\boldsymbol{P}, J)$ to account for the new type of conflict. Additionally, to ensure that rejection of a literal L cannot be based on the assumption that some conflicting literal $L' \in \overline{L}$ is true, a rejecting rule σ must satisfy the stronger condition $\ell^{\downarrow}(\overline{L}) > \ell^{\uparrow}(\mathsf{B}_\sigma)$. Finally, to prevent defeated rules from affecting the resulting models, we require that all supporting rules belong to $\mathsf{rem}(\boldsymbol{P}, J^*)$.

Definition 7 (Extended WS-semantics). *Let $\boldsymbol{P} = \langle P_i \rangle_{i<n}$ be a DLP. For interpretation J and a level mapping ℓ, the multiset of rejected rules* $\mathsf{rej}^{\neg}_{\ell}(\boldsymbol{P}, J)$ *is defined by:*
$\mathsf{rej}^{\neg}_{\ell}(\boldsymbol{P}, J) = \{ \pi \in P_i \mid i < n \wedge \exists j > i \, \exists \sigma \in P_j : \mathsf{H}_\sigma \in \overline{\mathsf{H}_\pi} \wedge J \models \mathsf{B}_\sigma \wedge \ell^{\downarrow}\left(\overline{\mathsf{H}_\pi}\right) > \ell^{\uparrow}(\mathsf{B}_\sigma) \}$.

Table 1. Desirable properties of rule update semantics

Generalisation of stable models	$\llbracket \langle P \rangle \rrbracket_{\mathsf{S}} = \llbracket P \rrbracket_{\mathsf{SM}}$.
Primacy of new information	If $J \in \llbracket \langle P_i \rangle_{i<n} \rrbracket_{\mathsf{S}}$, then $J \models P_{n-1}$.
Fact update	A sequence of consistent sets of facts $\langle P_i \rangle_{i<n}$ has the single model $\{\, l \in \mathcal{L} \mid \exists i < n : (l.) \in P_i \wedge (\forall j > i : \{\, \neg l., \sim l. \,\} \cap P_j = \emptyset) \,\}$.
Support	If $J \in \llbracket \boldsymbol{P} \rrbracket_{\mathsf{S}}$ and $l \in J$, then there is some rule $\pi \in \mathsf{all}(\boldsymbol{P})$ such that $\mathsf{H}_\pi = l$ and $J \models \mathsf{B}_\pi$.
Idempotence	$\llbracket \langle P, P \rangle \rrbracket_{\mathsf{S}} = \llbracket \langle P \rangle \rrbracket_{\mathsf{S}}$.
Absorption	$\llbracket \langle P, U, U \rangle \rrbracket_{\mathsf{S}} = \llbracket \langle P, U \rangle \rrbracket_{\mathsf{S}}$.
Augmentation	If $U \subseteq V$, then $\llbracket \langle P, U, V \rangle \rrbracket_{\mathsf{S}} = \llbracket \langle P, V \rangle \rrbracket_{\mathsf{S}}$.
Non-interference	If U and V are over disjoint alphabets, then $\llbracket \langle P, U, V \rangle \rrbracket_{\mathsf{S}} = \llbracket \langle P, V, U \rangle \rrbracket_{\mathsf{S}}$.
Immunity to empty updates	If $P_j = \emptyset$, then $\llbracket \langle P_i \rangle_{i<n} \rrbracket_{\mathsf{S}} = \left\llbracket \langle P_i \rangle_{i<n \wedge i \neq j} \right\rrbracket_{\mathsf{S}}$.
Immunity to tautologies	If $\langle Q_i \rangle_{i<n}$ is a sequence of sets of tautologies, then $\llbracket \langle P_i \cup Q_i \rangle_{i<n} \rrbracket_{\mathsf{S}} = \llbracket \langle P_i \rangle_{i<n} \rrbracket_{\mathsf{S}}$.
Causal rejection principle	For every $i < n$, $\pi \in P_i$ and $J \in \llbracket \langle P_i \rangle_{i<n} \rrbracket_{\mathsf{S}}$, if $J \not\models \pi$, then there exists some $\sigma \in P_j$ with $j > i$ such that $\mathsf{H}_\sigma \in \overline{\mathsf{H}_\pi}$ and $J \models \mathsf{B}_\sigma$.

The set $\llbracket \boldsymbol{P} \rrbracket_{\mathsf{WS}}^{\neg}$ *of* extended WS-models of $\boldsymbol{P}$ *consists of all interpretations J such that for some level mapping* ℓ*, the following conditions are satisfied: 1. J is a model of* $\mathsf{all}(\boldsymbol{P}) \setminus \mathsf{rej}_{\ell}^{\neg}(\boldsymbol{P}, J)$ *and 2. For every* $l \in J$ *there exists some rule* $\pi \in \mathsf{rem}(\boldsymbol{P}, J^*)$ *such that* $\mathsf{H}_\pi = l \wedge J \models \mathsf{B}_\pi \wedge \ell(\mathsf{H}_\pi) > \ell^{\uparrow}(\mathsf{B}_\pi)$.

The following theorems establish that the two defined semantics are equivalent, that they coincide with the original on DLPs without strong negation, and, unlike the transformational semantics for strong negation, the new semantics satisfy the early recovery principle.

Theorem 1. *Let* $\boldsymbol{P}_1$ *be a DLP and* $\boldsymbol{P}_2$ *be a DLP without strong negation. Then,* $\llbracket \boldsymbol{P}_1 \rrbracket_{\mathsf{WS}}^{\neg} = \llbracket \boldsymbol{P}_1 \rrbracket_{\mathsf{RD}}^{\neg}$ *and* $\llbracket \boldsymbol{P}_2 \rrbracket_{\mathsf{WS}}^{\neg} = \llbracket \boldsymbol{P}_2 \rrbracket_{\mathsf{RD}}^{\neg} = \llbracket \boldsymbol{P}_2 \rrbracket_{\mathsf{WS}} = \llbracket \boldsymbol{P}_2 \rrbracket_{\mathsf{RD}}$.

Theorem 2. *The extended RD-semantics and extended WS-semantics satisfy the early recovery principle.*

4 Properties

The various approaches to rule updates [5,7–18] share a number of basic characteristics, significantly differing in their technical realisation and classes of supported inputs, and desirable properties such as immunity to tautologies are violated by many of them.

Table 1 lists several generic properties proposed for rule updates that have been identified and formalised throughout the years [5,8,9,11]. The rule update semantics we defined in the previous section enjoys all of them, while retaining the same computational complexity as the stable models.

Theorem 3. *The extended RD-semantics and extended WS-semantics satisfy all properties listed in Table 1.*

Theorem 4. *Let $\boldsymbol{P}$ be a DLP. The problem of deciding whether some $J \in [\![\boldsymbol{P}]\!]^{\neg}_{\mathsf{WS}}$ exists is NP-complete. Given a literal L, the problem of deciding whether for all $J \in [\![\boldsymbol{P}]\!]^{\neg}_{\mathsf{WS}}$ it holds that $J \models L$ is coNP-complete.*

5 Concluding Remarks

In this paper we have identified shortcomings in the existing semantics for rule updates that fully support both strong and default negation, and proposed a generic *early recovery principle* that captures them formally. Subsequently, we provided two equivalent definitions of a new semantics for rule updates. We have shown that the newly introduced rule update semantics constitutes a strict improvement upon the state of the art in rule updates as it enjoys the following combination of characteristics, unmatched by any previously existing semantics:

- It allows for both strong and default negation in heads of rules, making it possible to move between any pair of epistemic states by means of updates;
- It satisfies the *early recovery principle* which guarantees the existence of a model whenever all conflicts in the original program are satisfied;
- It enjoys all rule update principles and desirable properties reported in Table 1;
- It does not increase the computational complexity of the stable model semantics upon which it is based.

However, the early recovery principle, as it is formulated in Sect. 3, only covers a single update of a set of facts by another set of facts. Can it be generalised further without rendering it too strong? Certain caution is appropriate here, since in general the absence of a stable model can be caused by odd cycles or simply by the fundamental differences between different approaches to rule update, and the purpose of this principle is not to choose which approach to take.

Nevertheless, one generalisation that should cause no harm is the generalisation to iterated updates, i.e. to sequences of sets of facts. Another generalisation that appears very reasonable is the generalisation to *acyclic DLPs*, i.e. DLPs such that $\mathsf{all}(\boldsymbol{P})$ is an acyclic program. An acyclic program has at most one stable model, and if we guarantee that all potential conflicts within it certainly get resolved, we can safely conclude that the rule update semantics should assign some model to it. We formalise these ideas in what follows.

A program P is *acyclic* [24] if for some level mapping ℓ, such that for every $l \in \mathcal{L}$, $\ell(l) = \ell(\neg l)$, and every rule $\pi \in P$ it holds that $\ell(\mathsf{H}_\pi) > \ell^{\uparrow}(\mathsf{B}_\pi)$. Given a DLP $\boldsymbol{P} = \langle P_i \rangle_{i<n}$, we say that *all conflicts in $\boldsymbol{P}$ are solved* if for every $i < n$ and each pair of rules $\pi, \sigma \in P_i$ such that $\mathsf{H}_\sigma \in \overline{\mathsf{H}_\pi}$ there is some $j > i$ and a fact $\rho \in P_j$ such that either $\mathsf{H}_\rho \in \overline{\mathsf{H}_\pi}$ or $\mathsf{H}_\rho \in \overline{\mathsf{H}_\sigma}$.

Generalised early recovery principle: If $\mathsf{all}(\boldsymbol{P})$ is acyclic and all conflicts in $\boldsymbol{P}$ are solved, then $[\![\boldsymbol{P}]\!]_{\mathsf{S}} \neq \emptyset$.

Note that this generalisation of the early recovery principle applies to a much broader class of DLPs than the original one. We illustrate this in the following example:

Example 6 (Recovery in a stratified program). Consider the following programs programs P, U and V: $P = \{\, p \leftarrow q, \sim r., \sim p \leftarrow s., q., s \leftarrow q. \,\}$, $U = \{\neg p., r \leftarrow q., \neg r \leftarrow q, s.\}$, and $V = \{\, \sim r. \,\}$. Looking more closely at program P, we see that atoms q and s are derived by the latter two rules inside it while atom r is false by default since there is no rule that could be used to derive its truth. Consequently, the bodies of the first two rules are both satisfied and as their heads are conflicting, P has no stable model. The single conflict in P is solved after it is updated by U, but then another conflict is introduced due to the latter two rules in the updating program. This second conflict can be solved after another update by V. Consequently, we expect that some stable model be assigned to the DLP $\langle P, U, V \rangle$.

The original early recovery principle does not impose this because the DLP in question has more than two components and the rules within it are not only facts. However, the DLP is acyclic, as shown by any level mapping ℓ with $\ell(p) = 3$, $\ell(q) = 0$, $\ell(r) = 2$ and $\ell(s) = 1$, so the generalised early recovery principle does apply. Furthermore, we also find the single extended RD-model of $\langle P, U, V \rangle$ is $\{\, \neg p, q, \neg r, s \,\}$, i.e. the semantics respects the stronger principle in this case.

It is no coincidence that the extended RD-semantics respects the stronger principle in the above example – the principle is generally satisfied by the semantics introduced in this paper.

Theorem 5. *The extended RD-semantics and extended WS-semantics satisfy the generalised early recovery principle.*

Both the original and the generalised early recovery principle can guide the future addition of full support for both kinds of negations in other approaches to rule updates, such as those proposed in [10, 13, 15, 18], making it possible to reach any belief state by updating the current program. Furthermore, adding support for strong negation is also interesting in the context of recent results on program revision and updates that are performed on the *semantic level*, ensuring syntax-independence of the respective methods [25–28], in the context of finding suitable condensing operators [29], and unifying with updates in classical logic [30].

References

1. Slota, M., Baláž, M., Leite, J.: On strong and default negation in logic program updates. In: Procs. of NMR 2014. INFSYS Research Report Series, vol. 1843–14-01, pp. 73–81. TU Wien (2014)

2. Berners-Lee, T., Hendler, J., Lassila, O.: The semantic web. Sci. Am. **284**(5), 28–37 (2001)
3. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: Procs. of ICLP 1988, pp. 1070–1080. MIT Press (1988)
4. Gelfond, M., Lifschitz, V.: Classical negation in logic programs and disjunctive databases. New Generation Computing **9**(3–4), 365–385 (1991)
5. Leite, J.A., Pereira, L.M.: Generalizing Updates: From Models to Programs. In: Dix, J., Moniz Pereira, L., Przymusinski, T.C. (eds.) LPKR 1997. LNCS (LNAI), vol. 1471, pp. 224–246. Springer, Heidelberg (1998)
6. Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H., Przymusinski, T.C.: Dynamic logic programming. In: Procs. of KR 1998, pp. 98–111. Morgan Kaufmann (1998)
7. Alferes, J.J., Leite, J.A., Pereira, L.M., Przymusinska, H., Przymusinski, T.C.: Dynamic updates of non-monotonic knowledge bases. The Journal of Logic Programming **45**(1–3), 43–70 (2000)
8. Eiter, T., Fink, M., Sabbatini, G., Tompits, H.: On properties of update sequences based on causal rejection. Theory and Practice of Logic Programming (TPLP) **2**(6), 721–777 (2002)
9. Leite, J.A.: Evolving Knowledge Bases, vol. 81. Frontiers of Artificial Intelligence and Applications. IOS Press (2003)
10. Sakama, C., Inoue, K.: An abductive framework for computing knowledge base updates. Theory and Practice of Logic Programming (TPLP) **3**(6), 671–713 (2003)
11. Alferes, J.J., Banti, F., Brogi, A., Leite, J.A.: The refined extension principle for semantics of dynamic logic programming. Stud. Logica. **79**(1), 7–32 (2005)
12. Banti, F., Alferes, J.J., Brogi, A., Hitzler, P.: The Well Supported Semantics for Multidimensional Dynamic Logic Programs. In: Baral, C., Greco, G., Leone, N., Terracina, G. (eds.) LPNMR 2005. LNCS (LNAI), vol. 3662, pp. 356–368. Springer, Heidelberg (2005)
13. Zhang, Y.: Logic program-based updates. ACM Transactions on Computational Logic **7**(3), 421–472 (2006)
14. Šefránek, J.: Irrelevant Updates and Nonmonotonic Assumptions. In: Fisher, M., van der Hoek, W., Konev, B., Lisitsa, A. (eds.) JELIA 2006. LNCS (LNAI), vol. 4160, pp. 426–438. Springer, Heidelberg (2006)
15. Delgrande, J.P., Schaub, T., Tompits, H.: A Preference-Based Framework for Updating Logic Programs. In: Baral, C., Brewka, G., Schlipf, J. (eds.) LPNMR 2007. LNCS (LNAI), vol. 4483, pp. 71–83. Springer, Heidelberg (2007)
16. Osorio, M., Cuevas, V.: Updates in answer set programming: An approach based on basic structural properties. Theory and Practice of Logic Programming **7**(4), 451–479 (2007)
17. Šefránek, J.: Static and dynamic semantics: Preliminary report. In: Procs. of MICAI 2011, pp. 36–42 (2011)
18. Krümpelmann, P.: Dependency semantics for sequences of extended logic programs. Logic Journal of the IGPL **20**(5), 943–966 (2012)
19. Alferes, J.J., Pereira, L.M.: Update-programs can update programs. In: Dix, J., Pereira, L.M., Przymusinski, T. C. (eds.) Procs. of NMELP 1996. LNCS, vol. 1216, 110–131. Springer, Heidelberg (1996)
20. Fages, F.: A new fixpoint semantics for general logic programs compared with the well-founded and the stable model semantics. New Generation Computing **9**(3/4), 425–444 (1991)
21. Keller, A.M., Winslett, M.: On the use of an extended relational model to handle changing incomplete information. IEEE Trans. Software Eng. **11**(7), 620–633 (1985)
22. Katsuno, H., Mendelzon, A.O.: On the difference between updating a knowledge base and revising it. In: Procs. of KR 1991, pp. 387–394. Morgan Kaufmann Publishers (1991)
23. Herzig, A., Rifi, O.: Propositional belief base update and minimal change. Artif. Intell. **115**(1), 107–138 (1999)
24. Apt, K.R., Bezem, M.: Acyclic programs. New Generation Computing **9**(3/4), 335–364 (1991)

25. Delgrande, J., Schaub, T., Tompits, H., Woltran, S.: A model-theoretic approach to belief change in answer set programming. ACM Transactions on Computational Logic (TOCL) 14(2), 14:1–14:46 (2013)
26. Slota, M., Leite, J.: The rise and fall of semantic rule updates based on se-models. Theory and Practice of Logic Programming FirstView, 1–39 (January 2014)
27. Slota, M., Leite, J.: Robust equivalence models for semantic updates of answer-set programs. In: Procs. of KR 2012, pp, 158–168. AAAI Press (2012)
28. Slota, M., Leite, J.: On semantic update operators for answer-set programs. In: Procs. of ECAI 2010. Frontiers in Artificial Intelligence and Applications, vol. 215, 957–962. IOS Press (2010)
29. Slota, M., Leite, J.: On condensing a sequence of updates in answer-set programming. In: Procs. of IJCAI 2013. IJCAI/AAAI (2013)
30. Slota, M., Leite, J.: A Unifying Perspective on Knowledge Updates. In: del Cerro, L.F., Herzig, A., Mengin, J. (eds.) JELIA 2012. LNCS, vol. 7519, pp. 372–384. Springer, Heidelberg (2012)