On Some Differences Between Semantics of Logic Program Updates

João Alexandre Leite

CENTRIA, New University of Lisbon, Portugal

Abstract. Since the introduction of logic program updates based on causal rejection of rules, several different semantics were set forth. All these semantics were introduced with the same underlying motivation i.e., to overcome the drawbacks of interpretation based updates by applying the principle of inertia to program rules, but they were all defined for different classes of logic programs thus making their comparisons difficult. In this paper we redefine such existing semantics, and set forth a new one, all in the more general setting of Generalized Logic Programs, in a way that facilitates their comparisons. Subsequently, we take a closer look at the subtle differences between these otherwise similar approaches.

1 Introduction

Concerning modifications to a knowledge base represented by a propositional theory, two abstract frameworks have been distinguished in [16] and [7]. One, theory revision, deals with incorporating new knowledge about a static world whose previous representation was incomplete or incorrect. The other deals with changing worlds, and is known as theory update. This paper addresses the issue of updates of logic programs, a subject that was recently put into context in [5].

Until recently, the work devoted to this issue followed the so called interpretation update approach based on the idea of reducing the problem of finding the update of a knowledge base by another knowledge base to the problem of finding the updates of its individual models. In [13] the authors introduce the framework of *Revision Programming*[1] where they allow the specification of updates of interpretations.

In [9], it is pointed out that such approach suffers from important drawbacks when the initial knowledge base is not purely extensional (it contains rules), and propose a new approach where the principle of inertia is applied to the rules of the initial knowledge base, rather than to its model literals. This led to the paradigm of *Dynamic Logic Programming*. A *Dynamic Logic Program* (DLP) is a sequence of Logic Programs where each represents a time period (state) and contains some knowledge that is supposed to be true at the state. The mutual relationships existing between different states (specified as the ordering relation) are then used to determine its declarative semantics.

Since the introduction of this form of updates of logic programs, several different semantics were set forth [9, 10, 2, 4, 1, 8], with the motivation of overcoming the drawbacks of interpretation based updates by applying the principle of inertia to program

---

[1] Despite the name, the authors are actually dealing with updates and not revisions.

rules. But they were all defined for different classes of logic programs, thus making their comparisons difficult. For example, [9, 10] define the semantics of *Justified Updates* for sequences of Revision Programs (a variant of Logic Programs), [2, 8] define the semantics of *Stable Models* for sequences of *Generalized (Extended) Logic Programs* (*GLP*) (Logic Programs allowing both default and strong negation in the heads of rules), and [4] define the semantics of Update Answer Sets for sequences of Extended Logic Programs. In [4], bridges between these semantics are established for restricted classes of logic programs, showing that under some restrictions the semantics coincide.

In this paper we take a closer look at the differences between these similar approaches. For this, we start by establishing the definitions of six different semantics, all set forth in a similar manner, thus making their comparisons easier. Four of these six semantics, all defined for sequences of *GLP* [2], either coincide or generalize the semantics mentioned before. The remaining two are new proposals. Subsequently, we compare all six semantics, either by means of examples or by means of properties, mostly with the intention of bringing out their differences rather than their similarities.

The paper is structured as follows: in Sect. 2 we recall some definitions; in Sect 3 we define the six mentioned semantics and relate them to the ones in the literature; in Sect 4 we establish some comparisons; in Sect 5 we wrap up.

2   Preliminaries

Let $\mathcal{A}$ be a set of propositional atoms. An **objective literal** is either an atom A or a strongly negated atom $\neg A$. A **default literal** is an objective literal preceded by not. A **literal** is either an objective literal or a default literal. A **rule** r is an ordered pair $H(r) \leftarrow B(r)$ where $H(r)$ (dubbed the head of the rule) is a literal and $B(r)$ (dubbed the body of the rule) is a finite set of literals. A rule with $H(r) = L_0$ and $B(r) = \{L_1, \ldots, L_n\}$ will simply be written as $L_0 \leftarrow L_1, \ldots, L_n$. A **tautology** is a rule of the form $L \leftarrow Body$ with $L \in Body$. A **generalized logic program** (*GLP*) P, in $\mathcal{A}$, is a finite or infinite set of rules. A program is called an **extended logic program** (*ELP*) if no default literals appear in the heads of its rules. If $H(r) = A$ (resp. $H(r) = not\,A$) then $not\,H(r) = not\,A$ (resp. $not\,H(r) = A$). If $H(r) = \neg A$, then $\neg H(r) = A$. By the **expanded generalized logic program** corresponding to the GLP P, denoted by $\mathbf{P}$, we mean the GLP obtained by augmenting P with a rule of the form $not\,\neg H(r) \leftarrow B(r)$ for every rule, in P, of the form $H(r) \leftarrow B(r)$, where $H(r)$ is an objective literal. An **interpretation** M of $\mathcal{A}$ is a set of objective literals that is consistent i.e., M does not contain both A and $\neg A$. An objective literal L is true in M, denoted by $M \vDash L$, iff $L \in M$, and false otherwise. A default literal $not\,L$ is true in M, denoted by $M \vDash not\,L$, iff $L \notin M$, and false otherwise. A set of literals B is true in M, denoted by $M \vDash B$, iff each literal in B is true in M. An interpretation M of $\mathcal{A}$ is an **answer set** of a GLP P iff $M' = least\,(\mathbf{P} \cup \{not\,A \mid A \notin M\})$, where $M' = M \cup \{not\_A \mid A \notin M\}$, A is an objective literal, and $least(.)$ denotes the least model of the definite program

---

[2] For motivation on the need for GLPs in updates, the reader is referred to [2, 8]. Note that within the context of updates, GLPs are not equivalent to Logic Programs with Integrity Constraints.

obtained from the argument program by replacing every default literal $not\ A$ by a new atom not_A. Let $AS\,(P)$ denote the set of answer-sets of P.

A **dynamic logic program** (*DLP*) is a sequence of generalized logic programs. Let $\mathcal{P} = (P_1, ..., P_s)$, $\mathcal{P}' = (P'_1, ..., P'_n)$ and $\mathcal{P}'' = (P''_1, ..., P''_s)$ be DLPs. We use $\rho\,(\mathcal{P})$ to denote the multiset all rules appearing in the programs $\mathbf{P}_1, ..., \mathbf{P}_s$, and $(\mathcal{P}, \mathcal{P}')$ to denote $(P_1, ..., P_s, P'_1, ..., P'_n)$ and $\mathcal{P} \cup \mathcal{P}''$ to denote $(P_1 \cup P''_1, ..., P_s \cup P''_s)$.

3   Semantics of Updates

In this Section we set forth the definitions of several semantics for dynamic logic programs in a uniform manner. Subsequently we establish that some of these semantics either coincide or generalize the semantics proposed in the literature. Without loss of generality, we only consider the semantics at the last state of the DLP. The common aspect that relates the semantics for updates addressed here, known as those *based on causal rejection of rules*, is their relying on the notion that a newer rule that is in conflict with an older one may reject it to avoid contradiction. We start by defining the notion of conflicting rules as follows: two rules r and r' are **conflicting**, denoted by $r \bowtie r'$, iff $H(r) = not\ H(r')$. Note that we do not establish a pair of rules whose heads are the strong negation of one another as being conflicting. Intuitively, these rules should be regarded as conflicting. They are not explicitly stated as such since, by using the expanded versions of GLPs, we have that for every pair of rules r_1 and r_2 in a DLP such that $H\,(r_1) = \neg H\,(r_2)$ there are two rules r'_1 and r'_2, introduced by the expansion operation, that are conflicting with the original ones i.e. $r_1 \bowtie r'_2$ and $r'_1 \bowtie r_2$. As will become clear below, this is enough to accomplish the desired rejection when a newer rule, whose head is the strongly negated atom of the rule being rejected, exists.

Next we define three notions of rejected rules in a DLP. Intuitively, when we consider an interpretation, a rule that belongs to a program of a DLP should be rejected if there is a newer conflicting rule whose body is true in the interpretation. This amounts to one notion of rejection. The second builds upon the first and allows for rules of the same state to reject each other. Intuitively this may seem odd but, as will be seen below, it produces desirable results. The third notion builds upon the first to further impose that the rejector rule is itself not rejected. Intuitively this condition seems reasonable but, as will be seen below, it produces results that may not be desirable.

Definition 1 (Rejected Rules). *Let* $\mathcal{P} = (P_1, ..., P_s)$ *be a* DLP *and* M *an interpretation. We define:*

$$Rej\,(\mathcal{P}, M) = \{r \mid r \in \mathbf{P}_i, \exists r' \in \mathbf{P}_j, i < j, r \bowtie r', M \vDash B(r')\}$$
$$Rej^+\,(\mathcal{P}, M) = \{r \mid r \in \mathbf{P}_i, \exists r' \in \mathbf{P}_j, i \leq j, r \bowtie r', M \vDash B(r')\}$$
$$Rej^*\,(\mathcal{P}, M) = \{r \mid r \in \mathbf{P}_i, \exists r' \in \mathbf{P}_j \setminus Rej^*\,(\mathcal{P}, M), i < j, r \bowtie r', M \vDash B(r')\}$$

Before we define the semantics, we turn our attention to the notion of default assumptions. In Section 2, when we defined the semantics of answer-sets for GLPs, we purposely did it in a somehow non standard way. Instead of using the standard GL transformation [6] (or, to be more precise, its modified version for GLPs [12]), we explicitly add the default assumptions (default literals for all those objective literals that do not belong to the considered interpretation) to the program and then determine its

least model. Similarly, when determining the semantics for DLPs, one also considers the least model of a program, consisting of all rules belonging to the programs of the DLP, without the rejected rules, together with a set of default assumptions. But unlike for the answer-set semantics above, not all the semantics for DLPs will allow for the 7explicit addition of *every* default literal whose corresponding objective literal does not belong to the interpretation being considered. Instead, for such DLP semantics, the default assumptions ($not\ A$) are restricted to those corresponding to unsupported objective literals i.e., those objective literals A for which there is no rule in the DLP whose body is true in the interpretation being considered. The intuition behind this is that if there exists a rule in the DLP (rejected or not) that would support the truth of some objective literal A, then A should not be able to be assumed *false by default*. Instead, its falsity, to exist, should only be obtained by some newer rule r that forces it to be false i.e., one with $H(r) = not\ A$ and $M \vDash B(r)$ (where M is the interpretation being considered). Otherwise, we can have situations where the assumption that some objective literal A is false indirectly leads to the rejection of a fact A, as will be shown below. Since not all semantics impose this restriction, we define two notions of default assumptions:

Definition 2 (Default Assumptions). *Let* $\mathcal{P} = (P_1, ..., P_s)$ *be a* DLP *and* M *an interpretation. Define (where A is an objective literal):* $Def^*(\mathcal{P}, M) = \{not\ A \mid A \notin M\}$ *and* $Def(\mathcal{P}, M) = \{not\ A \mid \nexists r \in \rho(\mathcal{P}), H(r) = A, M \vDash B(r)\}$.

Using each combination of rejected rules and default assumptions, we are now ready to define six distinct semantics for DLPs, all of which based on the intuition that some interpretation is a model according to a semantics iff it obeys an equation based on the least model of the multiset of all the rules in the (expanded) DLP, without those rejected rules, together with a set of default assumptions. The six semantics are dubbed *dynamic stable model semantics* (*DSM*), *dynamic justified update semantics* (*DJU*), *dynamic u-model semantics* (*DUM*), *dynamic answer-set semantics* (*DAS*), *refined dynamic stable model semantics* (*RDSM*) and *refined dynamic justified update semantics* (*RDJU*).

Definition 3 (Semantics of Updates). *Let* $\mathcal{P} = (P_1, ..., P_s)$ *be a* DLP *and* M *an interpretation. Then:*

DSM*: M is a* dynamic stable model *of* $\mathcal{P}$ *iff*
$$M' = least\left(\rho(\mathcal{P}) - Rej(\mathcal{P}, M) \cup Def(\mathcal{P}, M)\right)$$
DJU*: M is a* dynamic justified update *of* $\mathcal{P}$ *iff*
$$M' = least\left(\rho(\mathcal{P}) - Rej(\mathcal{P}, M) \cup Def^*(\mathcal{P}, M)\right)$$
DUM*: M is a* dynamic u-model *of* $\mathcal{P}$ *iff*
$$M' = least\left(\rho(\mathcal{P}) - Rej^*(\mathcal{P}, M) \cup Def(\mathcal{P}, M)\right)$$
DAS*: M is a* dynamic answer-set *of* $\mathcal{P}$ *iff*
$$M' = least\left(\rho(\mathcal{P}) - Rej^*(\mathcal{P}, M) \cup Def^*(\mathcal{P}, M)\right)$$
RDSM*: M is a* refined dynamic stable model *of* $\mathcal{P}$ *iff*
$$M' = least\left(\rho(\mathcal{P}) - Rej^+(\mathcal{P}, M) \cup Def(\mathcal{P}, M)\right)$$
RDJU*: M is a* refined dynamic justified update *of* $\mathcal{P}$ *iff*
$$M' = least\left(\rho(\mathcal{P}) - Rej^+(\mathcal{P}, M) \cup Def^*(\mathcal{P}, M)\right)$$

where $M', \rho(.)$ *and* $least(.)$ *are as before. Let* $DSM(\mathcal{P})$ *denote the set of all* dynamic stable models *of* $\mathcal{P}$, $DJU(\mathcal{P})$ *the set of all* dynamic justified updates *of* $\mathcal{P}$, $DUM(\mathcal{P})$

the set of all dynamic u-models *of* $\mathcal{P}$, DAS $(\mathcal{P})$ *the set of all* dynamic answer-sets *of* $\mathcal{P}$, $RDSM$ $(\mathcal{P})$ *the set of all* refined dynamic stable models *of* $\mathcal{P}$, *and* $RDJU$ $(\mathcal{P})$ *the set of all* refined dynamic justified updates *of* $\mathcal{P}$.

We now relate these semantics with those defined in the literature.

Proposition 1. *Let* $\mathcal{P}$ *be a DLP.* M *is a* dynamic stable model *of* $\mathcal{P}$ *iff* M *is a* stable model *of* $\mathcal{P}$ *(as of [2, 8]).*

Remark 1. Strictly speaking, if we consider the definitions in [2], the result only holds for normal generalized logic programs (i.e. without strong negation). This is so because of an incorrection in the original definition which has been corrected in [8].

Proposition 2. *Let* $\mathcal{P}$ *be a DLP.* M *is a* dynamic justified update *of* $\mathcal{P}$ *iff* M *is a* $\mathcal{P}-$justified update *(as of [10, 9]).*

Remark 2. The semantics of $\mathcal{P}-$*justified updates* was originally established in a setting where *Revision Programs* [13] were used instead of GLP's. We consider the trivial correspondence between GLPs and Revision Programs where each in (A) (resp. out (A)) of the latter corresponds to a A (resp not A) of the former.

The *dynamic answer-set semantics* generalizes the *update answer-set semantics* [4], originally defined for DLPs consisting of extended logic programs only i.e., without default negation in the heads of rules, to the case of DLPs consisting of GLPs.

Proposition 3. *Let* $\mathcal{P}$ $=$ $(P_1, ..., P_s)$ *be a DLP where each* P_i *is an extended logic program.* M *is a* dynamic answer-set *of* $\mathcal{P}$ *iff* M *is an* update answer-set *(as of [4]).*

In [1] the use of strong negation is not allowed. The *refined dynamic stable model semantics* defined here extends the semantics of [1] to allow for strong negation.

Proposition 4. *Let* $\mathcal{P} = (P_1, ..., P_s)$ *be a DLP where each* P_i *is a normal generalized logic programs (i.e. without strong negation).* M *is a* refined dynamic stable model *of* $\mathcal{P}$ *iff* M *is a* refined dynamic stable model *(as of [1]).*

The *dynamic u-model semantics* and the *refined dynamic justified update semantics* do not correspond to any previously defined semantics. Their definitions are only justified for the sake of completeness since they both suffer from some drawbacks.

4 Properties and Comparisons

In this Section we compare all six semantics by means of some properties and examples that illustrate their similarities and differences. We start by relating all semantics with the answer-set semantics:

Proposition 5 (Generalization of Answer-Set Semantics). *Let* $\mathcal{P}$ $=$ (P) *be a DLP consisting of a single GLP. Then* $DSM(\mathcal{P}) = DJU(\mathcal{P}) = DUM(\mathcal{P}) = DAS(\mathcal{P}) = RDSM(\mathcal{P}) = AS(P)$

A similar proposition does not hold for the *refined dynamic justified update semantics* $(P = \{a \leftarrow; not\ a \leftarrow\}$ serves as a counter-example). Since this semantics fails to obey

this simple and desirable property, we will not consider it further in this paper. Next we relate the sets of models that characterize each semantics:

Theorem 1. *Let* $\mathcal{P} = (P_1, ..., P_s)$ *be a* DLP. *Then* $RDSM(\mathcal{P}) \subseteq DSM(\mathcal{P}) \subseteq DJU(\mathcal{P}) \subseteq DAS(\mathcal{P})$ *and* $RDSM(\mathcal{P}) \subseteq DSM(\mathcal{P}) \subseteq DUM(\mathcal{P}) \subseteq DAS(\mathcal{P})$.

Example 1. Let $\mathcal{P} = (P_1, P_2, P_3)$ be the *DLP* where $P_1 = \{a \leftarrow\}$, $P_2 = \{not\ a \leftarrow\}$, and $P_3 = \{a \leftarrow a\}$. We obtain $RDSM(\mathcal{P}) = DSM(\mathcal{P}) = DJU(\mathcal{P}) = \{\{\}\}$ and $DUM(\mathcal{P}) = DAS(\mathcal{P}) = \{\{\}, \{a\}\}$.

This example serves to illustrate the difference in the results caused by allowing rejected rules to reject rules or otherwise. For those semantics that use $Rej(\mathcal{P}, M)$ and $Rej^+(\mathcal{P}, M)$ (RDSM, DSM and DJU), we only obtain one model, namely $\{\}$. Since, with $Rej(\mathcal{P}, M)$ and $Rej^+(\mathcal{P}, M)$, rejected rules can reject other rules, the rule $a \leftarrow$ in P_1 is always rejected by the rule $not\ a \leftarrow$ in P_2, independently of the interpretation M being considered. Then, since there are no rules that support a (the rule $a \leftarrow a$ in P_3 is not sufficient, by itself), we cannot have a model such that a belongs to it. The interpretation $\{\}$ is then the only one that verifies the condition to be a model. Note that this is valid for RDSM, DSM and DJU. For those semantics that use $Rej^*(\mathcal{P}, M)$ (DUM and DAS), since rejected rules are not allowed to reject other rules, the rejection of the rule $a \leftarrow$ in P_1 now depends on the rule $not\ a \leftarrow$ in P_2 being rejected or not. If we consider the interpretation $\{\}$, then the rule $a \leftarrow$ is rejected and $\{\}$ verifies the condition to be a model. If we consider the interpretation $\{a\}$, then the rule $a \leftarrow a$ in P_3 rejects the rule $not\ a \leftarrow$ in P_2 and, consequently, $a \leftarrow$ in P_1 is no longer rejected. Since $least(a \leftarrow; a \leftarrow a) = \{a\}$, we have that $\{a\}$ is also a model according to DUM and DAS. By inspecting $\mathcal{P}$, we argue that the intuitive result should be to allow for one model only, namely $\{\}$. Since we are dealing with updates, the rule $not\ a \leftarrow$ in P_2 should be understood as a change in the world into a state where a is unconditionally not true. Therefore, we should not be able to reuse the rule $a \leftarrow$ in P_1 again. According to DUM and DAS, this old rule in P_1 serves as the support for itself not to be rejected i.e. it serves as the support for a, rendering the rule $a \leftarrow a$ in P_3 one that rejects $not\ a \leftarrow$ in P_2, and therefore not able to reject $a \leftarrow$ in P_1. The reader may find this line of argumentation more convincing after replacing the proposition a with the proposition $alive$. Below, we come back to this issue when we define a property that encodes the intuition behind it, which holds for RDSM, DSM and DJU, but not for DUM and DAS.

Example 2. Consider $\mathcal{P} = (P_1, P_2)$ to be the *DLP* where $P_1 = \{a \leftarrow\}$ and $P_2 = \{not\ a \leftarrow not\ a\}$. We obtain $RDSM(\mathcal{P}) = DSM(\mathcal{P}) = DUM(\mathcal{P}) = \{\{a\}\}$ and $DJU(\mathcal{P}) = DAS(\mathcal{P}) = \{\{\}, \{a\}\}$.

With this example we can observe the different results obtained by using either $Def(\mathcal{P}, M)$ or $Def^*(\mathcal{P}, M)$. When $Def(\mathcal{P}, M)$ is used (RDSM, DSM and DUM), then only one model is obtained namely $\{a\}$. If we use $Def^*(\mathcal{P}, M)$ (DJU and DAS), the model $\{\}$ also exists. This model is obtained by assuming a to be false by default i.e. $not\ A \in Def^*(\mathcal{P}, M)$. This default assumption justifies the use of the rule $not\ a \leftarrow not\ a$ in P_2 to reject the fact $a \leftarrow$ in P_1. Arguably, if we look at the rule $not\ a \leftarrow not\ a$ as a tautological one, it is fair to expect it not to be responsible, by itself, for a change in the semantics. We argue that this DLP should only have one model, namely $\{a\}$.

The previous example leads to the next property, which serves as argument in favor of using $Def(\mathcal{P}, M)$ instead of $Def^*(\mathcal{P}, M)$. It relates these semantics with *Revision Programming* [13]. Such framework, which for lack of space we cannot formally recapitulate here, characterizes the interpretations that should be accepted as the result of the update of an initial interpretation by a revision program. Such accepted interpretations are called M-justified updates. If we encode the initial interpretation as a purely extensional GLP and make it the first program of a DLP with two programs, where the second encodes the revision program, then it is desirable that a semantics for the DLP coincide with the one provided by the interpretation updates. It turns out that only the three semantics that use $Def(\mathcal{P}, M)$ coincide with interpretation updates.

Definition 4 (Generalization of Interpretation Updates). *Let M be an interpretation and RP a revision program (according to [13]). Let P_{RP} be the GLP obtained from RP by replacing atoms of the form $in(L)$ (resp. $out(L)$) with L (resp $not\ L$). Let $P_M = \{A \leftarrow | A \in M\}$. We say that an update semantics SEM generalizes Interpretation Updates (IU)(in the sense of [13]) iff for every M and P_{RP} it holds that an interpretation I is a M-justified update iff it is a model of (P_M, P_{RP}) according to SEM i.e., $I \in SEM((P_M, P_{RP}))$.*

Theorem 2 (Generalization of Interpretation Updates). *RDSM, DSM and DUM generalize IU. DJU and DAS do not generalize IU.*

The DLP in Example 2 serves as a counter example to show that DJU and DAS do not generalize IU because according to the latter, $\{a\}$ is the only M-justified update.

From the exposition so far, it becomes apparent that the differences between the semantics are very subtle, and all related to the extent that these semantics are immune to tautologies. Immunity to tautologies is desirable and can be defined as follows:

Definition 5 (Immunity to Tautologies). *An update semantics SEM is immune to tautologies iff for any DLP $\mathcal{P} = (P_1, ..., P_s)$ and any sequence of sets of tautologies $\mathcal{E} = (E_1, ..., E_s)$, it holds that $SEM(\mathcal{P}) = SEM(\mathcal{P} \cup \mathcal{E})$.*

Theorem 3. *DSM, DUM, DJU and DAS are not immune to tautologies.*

Example 3. Consider the *DLP* $\mathcal{P} = (P_1, P_2)$ where $P_1 = \{a \leftarrow; not\ a \leftarrow\}$ and $P_2 = \{a \leftarrow a\}$. All DSM, DUM, DJU and DAS have a single model, namely $\{a\}$. According to these four semantics, if one program is contradictory, a tautological update has the effect of removing such contradiction.

Theorem 4. *[1] RDSM is immune to tautologies.*

Before we proceed, we establish a notion of equivalence between two DLPs under some update semantics. Intuitively two DLPs are equivalent if they have the same semantics and, when both are updated by the same arbitrary sequence of programs (i.e. the updating sequence is appended to both DLPs), their semantics still coincides.

Definition 6 (Update Equivalence). *Two* DLP*s $\mathcal{P}_\alpha$ and $\mathcal{P}_\beta$ are* update equivalent *under semantics SEM, denoted by $\mathcal{P}_\alpha \overset{SEM}{\equiv} \mathcal{P}_\beta$, iff for every DLP $\mathcal{P}$, it holds that $SEM((\mathcal{P}_\alpha,\mathcal{P})) = SEM((\mathcal{P}_\beta,\mathcal{P}))$.*

This notion of equivalence is important inasmuch as it allows us to replace a DLP that describes the history of some world by a simpler one, if we are not concerned with the past history, but we want to guarantee that the present and future are preserved. Ideally, we would like to devise, for each semantics, an operator that would condense any two consecutive programs belonging to a DLP into a single one, written in the same language. By repeatedly applying such an operator we would reduce a DLP to a single GLP. Such operators have the following formal definition:

Definition 7 (General State Condensing Operator). *Let $\mathcal{A}$ be a set of propositional atoms. Let Π denote the set of all generalized logic programs over the set of atoms $\mathcal{A}$. Let Θ be an operator with signature $\Theta : \Pi \times \Pi \to \Pi$. We say that Θ is a* general state condensing operator *for language $\mathcal{A}$ and semantics SEM iff for every DLP $\mathcal{P} = (P_1, ..., P_s)$ over $\mathcal{A}$ it holds that $\mathcal{P} \overset{SEM}{\equiv} (P_1, ..., P_{i-1}, \Theta(P_i, P_{i+1}), P_{i+2}, ..., P_s)$.*

Theorem 5. *Let $\mathcal{A}$ be a non-empty language. General state condensing operators for language $\mathcal{A}$ and semantics RDSM, DSM, DJU, DUM, and DAS, do not* exist.

Example 4. Let $\mathcal{P} = (P_1, P_2)$ be the *DLP*, over $\mathcal{A} = \{a, b\}$, where $P_1 = \{a \leftarrow b; b \leftarrow\}$ and $P_2 = \{not\, b \leftarrow not\, a\}$. We have $RDSM(\mathcal{P}) = DSM(\mathcal{P}) = DJU(\mathcal{P}) = DUM(\mathcal{P}) = DAS(\mathcal{P}) = \{\{\}, \{a, b\}\}$. Since a general state condensing operator Θ would condense P_1 and P_2 into a single program $\Theta(P_1, P_2)$, and all five update semantics coincide with the answer-set semantics, for DLPs with a single program, then, $AS(\Theta(P_1, P_2))$ would have to be equal to $\{\{\}, \{a, b\}\}$. But there is no generalized logic program whose answer sets are $\{\}$ and $\{a, b\}$, because $\{\} \subset \{a, b\}$ and it is known that answer-sets are minimal. Similar examples written in a language containing just one propositional atom exist, from which we prove the theorem.

The definition for general state condensing operators requires the language in which the resulting program is written to be the same as that of the original DLP. If we allow extensions to the original language, then there exists, for each of the semantics introduced, a polynomial transformation that takes a DLP and produces a single GLP, written in an extended language, whose answer-sets (when restricted to the initial language) coincide with the models of the update semantics. The existence of these transformations allows for the use of available answer-set software (e.g. *SMODELS* [14] and *DLV* [11]) to determine the update semantics, by means of a preprocessor that implements the transformation. Some of these implementations are publicly available. We do not present such transformations here for lack of space (some are in the cited literature), but their existence suffices to establish the following complexity results for all the semantics (inherited from those of Logic Programming under the Answer-set semantics):

Theorem 6 (Computational Complexity). *Let $\mathcal{P}$ be a DLP. Deciding if $DSM(\mathcal{P})$ (resp. $DJU(\mathcal{P})$, $DUM(\mathcal{P})$, $DAS(\mathcal{P})$, $RDSM(\mathcal{P})$) is not empty is $NP-complete$. Deciding if an interpretation belongs to $DSM(\mathcal{P})$ (resp. $RDSM(\mathcal{P})$, $DUM(\mathcal{P})$,*

$DAS(\mathcal{P})$, $DJU(\mathcal{P})$) is P. Deciding if an atom is true in at least one interpretation that belongs to $DSM(\mathcal{P})$ (resp. $DJU(\mathcal{P})$, $DUM(\mathcal{P})$, $DAS(\mathcal{P})$, $RDSM(\mathcal{P})$) is $NP-complete$; Deciding if an atom is true in all interpretations that belong to $DSM(\mathcal{P})$ (resp. $DJU(\mathcal{P})$, $DUM(\mathcal{P})$, $DAS(\mathcal{P})$, $RDSM(\mathcal{P})$) is $coNP-complete$.

Since the size of the program obtained by the mentioned transformations depends on the number of rules and the number of states of the DLP, we now address ways of simplifying a DLP both by eliminating certain states and certain rules, while preserving update equivalence.

Definition 8 (State Elimination). *Let* $\mathcal{P} = (P_1, ..., P_s)$ *be a* DLP. *Define:*
SEa: *If* $P_i = \{\}$, *then* $\mathcal{P} \stackrel{SEM}{\equiv} (P_1, ..., P_{i-1}, P_{i+1}, ..., P_s)$;
SEb: *If* $\forall r \in P_i, r' \in P_{i+1} : r \not\bowtie r'$, *then* $\mathcal{P} \stackrel{SEM}{\equiv} (P_1, ..., P_{i-1}, P_i \cup P_{i+1}, P_{i+2}, ..., P_s)$.

The first one, *SEa*, encodes that updates by empty programs should not change the semantics and should be allowed to be removed. *SEb* encodes that two consecutive programs that have no conflicting rules should be able to be merged into a single one. We now proceed to state simplification i.e. (syntactical) removal of superfluous rules:

Definition 9 (State Simplification). *Let* $\mathcal{P} = (P_1, ..., P_s)$ *be a* DLP. *Define:*
SSa: *If* $r \in P_i$ *and* $\exists r' \in P_j, i < j, H(r') = H(r)$ *and* $B(r') \subseteq B(r)$, *then* $\mathcal{P} \stackrel{SEM}{\equiv} (P_1, ..., P_i \backslash \{r\}, ..., P_s)$;
SSb: *If* $r \in P_i$ *and* $\exists r' \in P_j, i < j, H(r') = not\ H(r)$ *and* $B(r') \subseteq B(r)$, *then* $\mathcal{P} \stackrel{SEM}{\equiv} (P_1, ..., P_i \backslash \{r\}, ..., P_s)$.

The first one, *SSa*, encodes that one should be able to remove an older rule for some literal, if a newer rule for that literal exists and is equal or more general (i.e. its body is a subset of the older rule's body). *SSb* encodes that one should be able to remove an older rule for some literal if a newer rule for its default complement exists and its body is equal or a subset of the older rule's body. *SSb* seems intuitive because if the body of the newer rule is always true when the body of the older rule also is, then the conclusion of the newer rule should always prevail over the conclusion of the older one.

Theorem 7. *SEa and SEb hold for* $RDSM$, DSM, DJU, DUM *and* DAS. *SSa holds for* $RDSM$, DSM, DJU, DUM *and* DAS. *SSb holds for* $RDSM$, DSM *and* DJU. *SSb does not hold for* DUM *and* DAS.

The following example, illustrates why *SSb* does not hold for DUM and DAS, which is related to their use of $Rej^*(\mathcal{P}, M)$, instead of $Rej(\mathcal{P}, M)$.

Example 5. Consider the DLP of Ex. 1. By removing rule $a \leftarrow$ from P_1 (due to rule $not\ a \leftarrow$ in P_2) we obtain $\mathcal{P}' = (\{\}, P_2, P_3)$. Note that $\mathcal{P} \stackrel{DSM}{\equiv} \mathcal{P}'$ and $\mathcal{P} \stackrel{DJU}{\equiv} \mathcal{P}'$, but $\mathcal{P} \stackrel{DUM}{\not\equiv} \mathcal{P}'$ and $\mathcal{P} \stackrel{DAS}{\not\equiv} \mathcal{P}'$. To confirm the later negative case, just observe that $DAS(\mathcal{P}) = \{\{\}, \{a\}\}$ and $DAS(\mathcal{P}') = \{\{\}\}$, i.e. $DAS(\mathcal{P}) \neq DAS(\mathcal{P}')$. Likewise for DUM.

5   Discussion and Conclusions

From the definitions and results above, it becomes apparent that the differences between the semantics are very subtle, and all related to the extent that they are immune to tautologies[3]. Of the five semantics presented, the *Refined Dynamic Stable Model* semantics of [1] is the only one that is immune to tautologies. It turns out that such semantics goes a step further and is also immune to more elaborate tautological updates involving dependencies amongst more rules (c.f. [1]).

Related to Logic Program Updates, we can find other semantics in the literature [17, 15], although not following the *causal rejection of rules* approach. They follow a mixture of interpretation updates and rule based updates as they both determine the models of the theory before performing the update, yielding results that differ significantly from the ones described in this paper. In [3] the authors propose Disjunctive Logic Programs with Inheritance, which can be used to encode updates. In [4], the non-disjunctive fragment is proved equivalent to the Update Answer Sets semantics.

References

1. J. J. Alferes, F. Banti, A. Brogi, and J. A. Leite. Semantics for dynamic logic programming: a principle-based approach. In *Procs. of LPNMR-7*, volume 2923 of *LNAI*. Springer, 2004.
2. J. J. Alferes, J. A. Leite, L. M. Pereira, H. Przymusinska, and T. Przymusinski. Dynamic updates of non-monotonic knowledge bases. *Journal of Logic Programming*, 45(1-3), 2000.
3. F. Buccafurri, W. Faber, and N. Leone. Disjunctive logic programs with inheritance. In *Procs. of ICLP'99*. MIT Press, 1999.
4. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. On properties of update sequences based on causal rejection. *Theory and Practice of Logic Programming*, 2(6), 2002.
5. T. Eiter, M. Fink, G. Sabbatini, and H. Tompits. Using methods of declarative logic programming for intelligent information agents. *Theory and Practice of Logic Programming*, 2(6), 2002.
6. M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *Procs. of ICLP'90*. MIT Press, 1990.
7. H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In *Procs. of KR'91*. Morgan Kaufmann, 1991.
8. J. A. Leite. *Evolving Knowledge Bases*, volume 81 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2003.
9. J. A. Leite and L. M. Pereira. Generalizing updates: From models to programs. In *Procs. of LPKR'97*, volume 1471 of *LNAI*. Springer, 1997.
10. J. A. Leite and L. M. Pereira. Iterated logic program updates. In *Procs. of JICSLP'98*. MIT Press, 1998.
11. N. Leone, G. Pfeifer, W. Faber, F. Calimeri, T. Dell'Armi, T. Eiter, G. Gottlob, G. Ianni, G. Ielpa, S. Perri C. Koch, and A. Polleres. The dlv system. In *Procs. of JELIA'02*, volume 2424 of *LNAI*. Springer-Verlag, 2002.
12. V. Lifschitz and T. Woo. Answer sets in general non-monotonic reasoning (preliminary report). In *Procs. of KR'92*. Morgan-Kaufmann, 1992.

---

[3] In this paper, for lack of space, we have not explored the classes of programs for which these semantics coincide. In [4, 8] the reader can find some results on this subject.

13. V. Marek and M. Truszcczyński. Revision programming. *Theoretical Computer Science*, 190(2), 1998.
14. I. Niemelä and P. Simons. Smodels: An implementation of the stable model and well-founded semantics for normal LP. In *Procs. of LPNMR'97*, volume 1265 of *LNAI*. Springer, 1997.
15. C. Sakama and K. Inoue. Updating extended logic programs through abduction. In *Procs. of LPNMR'99*, volume 1730 of *LNAI*. Springer, 1999.
16. M. Winslett. Reasoning about action using a possible models approach. In *Procs. of AAAI'88*, 1988.
17. Y. Zhang and N. Y. Foo. Updating logic programs. In *Procs. of ECAI'98*. John Wiley & Sons, 1998.