Considerations on Updates of Logic Programs

Thomas Eiter, Michael Fink, Giuliana Sabbatini, and Hans Tompits

Institut und Ludwig Wittgenstein Labor für Informationssysteme, TU Wien
Favoritenstraße 9–11, A-1040 Wien, Austria
`{eiter,michael,giuliana,tompits}@kr.tuwien.ac.at`

Abstract. Among others, Alferes et al. (1998) presented an approach for updating logic programs with sets of rules based on dynamic logic programs. We syntactically redefine dynamic logic programs and investigate their semantical properties, looking at them from perspectives such as a belief revision and abstract consequence relation view. Since the approach does not respect minimality of change, we refine its stable model semantics and present minimal stable models and strict stable models. We also compare the update approach to related work, and find that is equivalent to a class of inheritance programs independently defined by Buccafurri et al. (1999).

1 Introduction

In recent years, agent-based computing has gained increasing interest. The need for software agents that behave "intelligently" in their environment led to question for possibilities of equipping them with advanced reasoning capabilities.

The research on logic-based AI, and in particular the work on logic programming, has produced a number of approaches and methods from which we can take advantage for accomplishing this goal (see e.g. [11]). It has been realized, however, that further work is needed for extending them to fully support that agents must adapt over time and adjust their decision making.

In a simple (but as for currently deployed agent systems, realistic) setting, an agent's knowledge base KB may be modeled as a logic program. The agent may now be prompted to adjust its KB after receiving new information in terms of an *update* U, which is a clause or a set of clauses that need to be incorporated into KB. Simply adding the rules of U to KB does not give a satisfactory solution in practice, and will result in inconsistency even in simple cases. For example, if KB contains the rule $a \leftarrow$ and U consists of the rule $not\, a \leftarrow$ stating that a is not provable, then the union $KB \cup U$ is not consistent under stable semantics (naturally generalized to programs with default negation in rule heads [21]), which is the predominating two-valued semantics for declarative logic programs.

Most recently, several approaches for updating logic programs with (sets of) rules have been presented [2, 5, 17, 13]. In particular, the concept of dynamic logic programs by Alferes et al., introduced in [2] and further developed in [3, 5, 4, 20], has attracted a lot of interest. Their approach has its roots, and generalizes, the idea of revision programming [22], and provides the basis for LUPS, a logic-programming based update specification language [5]. The basic idea behind the approach is that in case of conflicting rules, a rule r in U (which is assumed to be correct as of the time of the update request) is more reliable than any rule r' in KB. Thus, application of r rejects application of r'. In the previous example, the rule $not\ a \leftarrow$ from U rejects the rule $a \leftarrow$ from KB, thus resolving the conflict by adopting that a is not provable. The idea is naturally extended to sequences of updates $U_1, \ldots, U_n$ by considering the rules in more recent updates as more reliable.

While uses and extensions of dynamic logic programming have been discussed, cf. [5, 4, 20], its properties and relationships to other approaches and related formalisms have been less explored (but see [4]). The aim of this paper is to shed light on these issues, and help us to get a better understanding of dynamic logic programming and related approaches in logic programming.

The main contributions of our work can be summarized as follows.

- We syntactically redefine dynamic logic programs to equivalent *update programs*, for which stable models are defined. Update programs are slightly less involved and, as we believe, better reflect the working of the approach than the original definition of dynamic logic programs. For this, information about rule rejection is explicitly represented at the object level through rejection atoms. The syntactic redefinition, which reduces the type of rules in update programs, is helpful for establishing formal results about properties.
- We investigate properties of update programs. We consider them from the perspective of belief revision, and review different sets of postulates that have been proposed in this area. We view update programs as nonmonotonic consequence operators, and consider further properties of general interest. As it turns out, update programs (and thus dynamic logic programs) do not satisfy many of the properties defined in the literature. This is partly explained by the nonmonotonicity of logic programs and the causal rejection principle embodied in the semantics, which strongly depends on the syntax of rules.
- Dynamic logic programs make no attempt to respect minimality of change. We thus refine the semantics of update programs and introduce minimal stable models and strict stable models. Informally, minimal stable models minimize the set of rules that need to be rejected, and strict stable models further refine on this by assigning rules from a later update higher priority.
- We compare update programs to alternative approaches for updating logic programs [13, 17] and related work on inheritance programs [9]. We find that update programs are equivalent to a class of inheritance programs. Thus, update programs (and dynamic logic programs) may be semantically regarded as fragment of the framework in [9], which has been developed independently of [2, 5]. Our results on the semantical properties of update programs apply to this fragment as well.

Due to space reasons, the presentation is necessarily succinct and proofs are omitted. More details will be given in the full version of this paper.

2 Preliminaries

Generalized logic programs [21] consist of rules built over a set $\mathcal{A}$ of propositional atoms where default negation not is available. A *literal*, L, is either an atom A (a *positive literal*) or the negation $not\,A$ of an atom A (a *negative literal*, also called *default literal*). For a literal L, the *complementary literal*, $not\,L$, is $not\,A$ if $L = A$, and A if $L = not\,A$, for some atom A. For a set S of literals, $not\,S$ is given by $not\,S = \{not\,L \mid L \in S\}$. We also denote by $Lit_{\mathcal{A}}$ the set $\mathcal{A} \cup not\,\mathcal{A}$ of all literals over $\mathcal{A}$.

A *rule*, r, is a clause of the form $L_0 \leftarrow L_1, \ldots, L_n$, where $n \geq 0$ and L_0 may be missing, and each L_i $(0 \leq i \leq n)$ is a default literal, i.e., either an atom A or a negated atom $not\,A$. We call L_0 the *head* of r and the set $\{L_1, \ldots, L_n\}$ the *body* of r. The head of r will also be denoted by $H(r)$, and the body of r will be denoted by $B(r)$. If the rule r has an empty head, then r is a *constraint*; if the body of r is empty and the head is non-empty, then r is a *fact*. We say that r has a *negative head* if $H(r) = not\,A$, for some atom A. The set $B^+(r)$ comprises the positive literals of $B(r)$, whilst $B^-(r)$ contains all default literals of $B(r)$.

By $\mathcal{L}_{\mathcal{A}}$ we denote the set of all rules over the set $\mathcal{A}$ of atoms. We will usually write $\mathcal{L}$ instead of $\mathcal{L}_{\mathcal{A}}$ if the underlying set $\mathcal{A}$ is fixed. A *generalized logic program* (GLP) P *over* $\mathcal{A}$ is a finite subset of $\mathcal{L}_{\mathcal{A}}$. If no rule in P contains a negative head, then P is a *normal logic program* (NLP); if no default negation whatsoever occurs in P, then P is a *positive program*.

By an (*Herbrand*) *interpretation* we understand any subset $I \subseteq \mathcal{A}$. The relation $I \models L$ for a literal L is defined as follows:

- if $L = A$ is an atom, then $I \models A$ iff $A \in I$;
- if $L = not\,A$ is a default literal, then $I \models not\,A$ iff $I \not\models A$.

If $I \models L$, then I is a *model* of L, and L is said to be *true* in I (if $I \not\models L$, then L is *false* in I). For a set S of literals, $I \models S$ iff $I \models L$ for all $L \in S$. Accordingly, we say that I is a model of S. Furthermore, for a rule r, we define $I \models r$ iff $I \models H(r)$ whenever $I \models B(r)$. In particular, if r is a contraint, then $I \models r$ iff $I \not\models B(r)$. In both cases, if $I \models r$, then I is a model of r. Finally, $I \models P$ for a program P iff $I \models r$ for all $r \in P$.

If a positive logic program P has some model, it has always a smallest Herbrand model, which we will denote by $lm(P)$. If P has no model, for technical reasons it is convenient to set $lm(P) = Lit_{\mathcal{A}}$.

We define the *reduct*, P^I, of a generalized program P w.r.t. to an Herbrand interpretation I as follows. P^I results from P by

1. deleting any rule r in P such that either $I \models B^-(r)$, or $I \models H(r)$ if $H(r) = not\,A$ for some atom A; and
2. replacing any remaining rule r by r^I, where $r^I = H(r) \leftarrow B^+(r)$ if $H(r)$ is positive, and $r^I = \leftarrow B^+(r)$ otherwise (r^I is called the *reduct* of r).

Observe that P^I is a positive program, hence $lm(P^I)$ is well-defined. We say that I is a *stable model* of P iff $lm(P^I) = I$. By $\mathcal{S}(P)$ we denote the set of all stable models of P. A program is *satisfiable* if $\mathcal{S}(P) \neq \emptyset$.

We regard a logic program P as the *epistemic state* of an agent. The given semantics is used for assigning a *belief state* to any epistemic state P in the following way.

Let $I \subseteq \mathcal{A}$ be an Herbrand interpretation. Define

$$Bel_{\mathcal{A}}(I) = \{r \in \mathcal{L}_{\mathcal{A}} \mid I \models r\}.$$

Furthermore, for a class $\mathcal{I}$ of interpretations, define $Bel_{\mathcal{A}}(\mathcal{I}) = \bigcap_{i \in \mathcal{I}} Bel_{\mathcal{A}}(I)$.

Definition 2.1. *For a logic program P, the belief state, $Bel_{\mathcal{A}}(P)$, of P is given by $Bel_{\mathcal{A}}(P) = Bel_{\mathcal{A}}(\mathcal{S}(P))$, where $\mathcal{S}(P)$ is the collection of all stable models of P.*

We write $P \models_{\mathcal{A}} r$ if $r \in Bel_{\mathcal{A}}(P)$. As well, for any program Q, we write $P \models_{\mathcal{A}} Q$ if $P \models_{\mathcal{A}} q$ for all $q \in Q$. Two programs, P_1 and P_2, are *equivalent* (modulo the set $\mathcal{A}$), symbolically $P_1 \equiv_{\mathcal{A}} P_2$, iff $Bel_{\mathcal{A}}(P_1) = Bel_{\mathcal{A}}(P_2)$. Usually we will drop the subscript "$\mathcal{A}$" in $Bel_{\mathcal{A}}(\cdot)$, $\models_{\mathcal{A}}$, and $\equiv_{\mathcal{A}}$ if no ambiguity can arise.

An alternative for defining the belief state would consist in considering brave rather than cautious inference, which we omit here.

Belief states enjoy the following natural properties:

Theorem 2.1. *For every logic program P, we have that:*

1. $P \subseteq Bel(P)$;
2. $Bel(Bel(P)) = Bel(P)$;
3. $\{r \mid I \models r$, *for every interpretation* $I\} \subseteq Bel(P)$.

Clearly, the belief operator $Bel(\cdot)$ is nonmonotonic, i.e., in general $P_1 \subseteq P_2$ does not imply $Bel(P_1) \subseteq Bel(P_2)$.

3 Update Programs

We introduce a framework for update programs which simplifies the approach introduced in [2]. By an *update sequence*, $\boldsymbol{P}$, we understand a series $P_1, \ldots, P_n$ of general logic programs where each P_i is assumed to update the information expressed by the initial section $P_1, \ldots, P_{i-1}$. This update sequence is translated into a single program P' representing the update information given by $\boldsymbol{P}$. The "intended" stable models of $\boldsymbol{P}$ are identified with the stable models of P' (modulo the original language).

Let $\boldsymbol{P} = P_1, \ldots, P_n$ be an update sequence over a set of atoms $\mathcal{A}$. We assume a set of atoms $\mathcal{A}^*$ extending $\mathcal{A}$ by new, pairwise distinct atoms $rej(\cdot)$, A_i, and A_i^-, where $A \in \mathcal{A}$ and $1 \leq i \leq n$. Furthermore, we assume an injective *naming function* $N(\cdot, \cdot)$, which assigns to each rule r in a program P_i a distinguished name, $N(r, P_i)$, obeying the condition $N(r, P_i) \neq N(r', P_j)$ whenever $i \neq j$. With a slight abuse of notation we shall identify r with $N(r, P_i)$ as usual.

Definition 3.1. *Given an update sequence $\boldsymbol{P} = P_1, \ldots, P_n$ over a set of atoms $\mathcal{A}$ we define the update program $\boldsymbol{P}_{\triangleleft} = P_1 \triangleleft \ldots \triangleleft P_n$ over $\mathcal{A}^*$ consisting of the following items:*

1. *all constraints in P_i, $1 \leq i \leq n$;*

2. *for each* $r \in P_i$, $1 \leq i \leq n$*:*

$$\begin{array}{ll} A_i \leftarrow B(r), \mathit{not}\ \mathit{rej}(r) & \text{if } H(r) = A; \\ A_i^- \leftarrow B(r), \mathit{not}\ \mathit{rej}(r) & \text{if } H(r) = \mathit{not}\ A; \end{array}$$

3. *for each* $r \in P_i$, $1 \leq i < n$*:*

$$\begin{array}{ll} \mathit{rej}(r) \leftarrow B(r), A_{i+1}^- & \text{if } H(r) = A; \\ \mathit{rej}(r) \leftarrow B(r), A_{i+1} & \text{if } H(r) = \mathit{not}\ A; \end{array}$$

4. *for each atom* A *occurring in* P $(1 \leq i < n)$*:*

$$A_i^- \leftarrow A_{i+1}^-; \qquad A_i \leftarrow A_{i+1}; \qquad A \leftarrow A_1; \qquad \leftarrow A_1, A_1^-.$$

Informally, this program expresses layered derivability of an atom A or a literal $\mathit{not}\ A$, beginning at the top layer P_n downwards to the bottom layer P_1. The rule r at layer P_i is only applicable if it is not refuted by a literal L that is incompatible with $H(r)$ derived at a higher level. Inertia rules propagate a locally derived value for A downwards to the first level, where the local value is made global; the constraint $\leftarrow A_1, A_1^-$ is used here in place of the rule $\mathit{not}\ A \leftarrow A_1^-$.

Similar to the transformation given in [2], $P_\triangleleft$ is modular in the sense that the transformation for $P' = P_1, \ldots, P_n, P_{n+1}$ augments $P_\triangleleft = P_1 \triangleleft \ldots \triangleleft P_n$ only with rules depending on $n + 1$.

We remark that $P_\triangleleft$ can obviously be slightly simplified, which is relevant for implementing our approach. All literals $\mathit{not}\ \mathit{rej}(r)$ in rules with heads A_n or A_n^- can be removed: since $\mathit{rej}(r)$ cannot be derived, they evaluate to true in each stable model of $P_\triangleleft$. Thus, no rule from P_n is rejected in a stable model of $P_\triangleleft$, i.e., all most recent rules are obeyed.

The intended models of an update sequence $P = P_1, \ldots, P_n$ are defined in terms of the stable models of $P_\triangleleft$.

Definition 3.2. *Let* $P = P_1, \ldots, P_n$ *be an update sequence over a set of atoms* $\mathcal{A}$. *Then,* $S \subseteq \mathcal{A}$ *is an* (*update*) *stable model of* P *iff* $S = S' \cap \mathcal{A}$ *for some stable model* S' *of* $P_\triangleleft$. *The collection of all update stable models of* P *is denoted by* $\mathcal{U}(P)$.

Following the case of single programs, an update sequence $P = P_1, \ldots, P_n$ is regarded as the epistemic state of an agent, and the belief state $\mathit{Bel}(P)$ is given by $\mathit{Bel}(\mathcal{U}(P))$. As well, the update sequence P is satisfiable iff $\mathcal{U}(P) \neq \emptyset$.

For illustration of Definition 3.2, consider the following example, taken from [2].

Example 3.1. Consider the update of P_1 by P_2, where

$$P_1 = \{\, r_1 : \mathit{sleep} \leftarrow \mathit{not}\ \mathit{tv\_on}, \quad r_2 : \mathit{tv\_on} \leftarrow, \quad r_3 : \mathit{watch\_tv} \leftarrow \mathit{tv\_on} \,\};$$
$$P_2 = \{\, r_4 : \mathit{not}\ \mathit{tv\_on} \leftarrow \mathit{power\_failure}, \quad r_5 : \mathit{power\_failure} \leftarrow \,\}.$$

The single stable model of $P = P_1, P_2$ is, as desired, $S = \{\mathit{power\_failure}, \mathit{sleep}\}$, since S' is the only stable model of $P_\triangleleft$:

$$S' = \{\, \mathit{power\_failure}_2, \mathit{power\_failure}_1, \mathit{power\_failure}, \\ \mathit{tv\_on}_2^-, \mathit{tv\_on}_1^-, \mathit{rej}(r_2), \mathit{sleep}_1, \mathit{sleep} \,\}.$$

If new information arrives in form of the program P_3:

$$P_3 = \{\ r_6 :\ \mathit{not\,power\_failure} \leftarrow\ \},$$

then the update sequence P_1, P_2, P_3 has the stable model $T = \{\mathit{tv\_on}, \mathit{watch\_tv}\}$, generated by the model T' of $P_1 \lhd P_2 \lhd P_3$:

$$T' = \{\ \mathit{power\_failure}_3^-, \mathit{power\_failure}_2^-, \mathit{power\_failure}_1^-, \\ \mathit{rej}(r_5), \mathit{tv\_on}_1, \mathit{tv\_on}, \mathit{watch\_tv}_1, \mathit{watch\_tv}\ \}.$$

Next, we discuss some properties of our approach. The first result guarantees that stable models of P are uniquely determined by the stable models of $P_\lhd$.

Theorem 3.1. *Let $P = P_1, \ldots, P_n$ be an update sequence over a set of atoms $\mathcal{A}$, and let S, T be stable models of $P_\lhd$. Then, $S \cap \mathcal{A} = T \cap \mathcal{A}$ only if $S = T$.*

If an update sequence P consists of a single program, the notion of update stable models of P and regular stable models of P coincide.

Theorem 3.2. *Let P be an update sequence consisting of a single program P_1, i.e., $P = P_1$. Then, $\mathcal{U}(P) = \mathcal{S}(P_1)$.*

Stable models of update sequences can also be characterized in a purely *declarative* way. To this end, we introduce the following concept.

For an update sequence $P = P_1, \ldots, P_n$ over a set of atoms $\mathcal{A}$ and $S \subseteq \mathcal{A}$, we define the *rejection set* of S by $\mathit{Rej}(S,P) = \bigcup_{i=1}^{n} \mathit{Rej}_i(S,P)$, where $\mathit{Rej}_n(S,P) = \emptyset$, and, for $n > i \geq 1$,

$$\mathit{Rej}_i(S,P) = \{r \in P_i \mid \exists r' \in P_j \setminus \mathit{Rej}_j(S,P),\ \text{for some } j \in \{i+1, \ldots, n\}, \\ \text{such that } H(r') = \mathit{not}\, H(r) \text{ and } S \models B(r) \cup B(r')\}.$$

That is, $\mathit{Rej}(S,P)$ contains those rules from P which are rejected on the basis of rules which are not rejected themselves.

We obtain the following characterization of stable models, mirroring a similar result given in [2].

Theorem 3.3. *Let $P = P_1, \ldots, P_n$ be an update sequence over a set of atoms $\mathcal{A}$, and let $S \subseteq \mathcal{A}$. Then, S is a stable model of P iff $S = \mathit{lm}((P \setminus \mathit{Rej}(S,P))^S)$.*

4 Principles of Update Sequences

In this section, we discuss several kinds of postulates which have been advocated in the literature on belief change and examine to what extent update sequences satisfy these principles. This issue has not been addressed extensively in previous work [2, 3]. We first consider update programs from the perspective of *belief revision*, and assess the relevant postulates from this area. Afterwards, we briefly analyze further properties, like viewing update programs as *nonmonotonic consequence operators* and other general principles.

4.1 Belief revision

Following [14], two different approaches to belief revision can be distinguished: (i) *immediate revision*, where the new information is simply added to the current stock of beliefs and the belief change is accomplished through the semantics of the underlying (often, nonmonotonic) logic; and (ii) *logic-constrained revision*, where the new stock of beliefs is determined by a nontrivial operation which adds and retracts beliefs, respecting logical inference and some constraints.

In the latter approach, it is assumed that beliefs are sentences from some given logical language $\mathcal{L}_{\mathcal{B}}$ which is closed under the standard boolean connectives. A *belief set*, K, is a subset of $\mathcal{L}_{\mathcal{B}}$ which is closed under a consequence operator $Cn(\cdot)$ of the underlying logic. A *belief base* for K is a subset $B \subseteq K$ such that $K = Cn(B)$. A belief base is a special case of an *epistemic state* [10], which is a set of sentences E representing an associated belief set K in terms of a mapping $Bel(\cdot)$ such that $K = Bel(E)$, where E need not necessarily have the same language as K.

In what follows, we first introduce different classes of postulates, and then we examine them with respect to update sequences.

AGM Postulates One of the main aims of logic-constrained revision is to characterize suitable revision operators through postulates. Alchourrón, Gärdenfors, and Makinson (AGM) [1] considered three basic operations on a belief set K:

– *expansion* $K + \phi$, which is simply adding the new information $\phi \in \mathcal{L}_B$ to K;
– *revision* $K \star \phi$, which is sensibly revising K in the light of ϕ (in particular, when K contradicts ϕ); and
– *contraction* $K - \phi$, which is removing ϕ from K.

AGM presented a set of postulates, K⋆1–K⋆8, that any revision operator $\star$ mapping a belief set $K \subseteq \mathcal{L}_B$ and a sentence $\phi \in \mathcal{L}_B$ into the revised belief set $K \star \phi$ should satisfy. If, following [10, 8], we assume that K is represented by an epistemic state E, then the postulates K⋆1–K⋆8 can be reformulated as follows:

(K1) $E \star \phi$ represents a belief set.
(K2) $\phi \in Bel(E \star \phi)$.
(K3) $Bel(E \star \phi) \subseteq Bel(E + \phi)$.
(K4) $\neg\phi \notin Bel(E)$ implies $Bel(E + \phi) \subseteq Bel(E \star \phi)$.
(K5) $\bot \in Bel(E \star \phi)$ iff ϕ is unsatisfiable.
(K6) $\phi_1 \equiv \phi_2$ implies $Bel(E \star \phi_1) = Bel(E \star \phi_2)$.
(K7) $Bel(E \star (\phi \wedge \psi)) \subseteq Bel((E \star \phi) + \psi)$.
(K8) $\neg\psi \notin Bel(E \star \phi)$ implies $Bel((E \star \phi) + \psi) \subseteq Bel(E \star (\phi \wedge \psi))$.

Here, $E \star \phi$ and $E + \phi$ is the revision and expansion operation, respectively, applied to E. Informally, these postulates express that the new information should be reflected after the revision, and that the belief set should change as little as possible. As has been pointed, this set of postulates is appropriate for new information about an *unchanged world*, but not for incorporation of a change to the actual world. Such a mechanism is addressed by the next set of postulates, expressing *update* operations.

Update Postulates For update operators $B \diamond \phi$ realizing a change ϕ to a belief base B, Katsuno and Mendelzon [18] proposed a set of postulates, U$\star$1–U$\star$8, where both ϕ and B are propositional sentences over a finitary language. For epistemic states E, these postulates can be reformulated as follows.

(U1) $\phi \in Bel(E \diamond \phi)$.
(U2) $\phi \in Bel(E)$ implies $Bel(E \diamond \phi) = Bel(E)$.
(U3) If $Bel(E)$ is consistent and ϕ is satisfiable, then $Bel(E \diamond \phi)$ is consistent.
(U4) If $Bel(E) = Bel(E')$ and $\phi \equiv \psi$, then $Bel(E \diamond \phi) = Bel(E \diamond \psi)$.
(U5) $Bel(E \diamond (\phi \wedge \psi)) \subseteq Bel((E \diamond \phi) + \psi)$.
(U6) If $\phi \in Bel(E \diamond \psi)$ and $\psi \in Bel(E \diamond \phi)$, then $Bel(E \diamond \phi) = Bel(E \diamond \psi)$.
(U7) If $Bel(E)$ is complete, then $Bel(E \diamond (\psi \vee \psi')) \subseteq Bel(E \diamond \psi) \wedge Bel(E \diamond \psi'))$.[1]
(U8) $Bel((E \vee E') \diamond \psi) = Bel((E \diamond \psi) \vee (E' \diamond \psi)$.

Here, conjunction and disjunction of epistemic states are presumed to be definable in the given language (like, e.g., in terms of intersection and union of associated sets of models, respectively).

The most important differences between (K1)–(K8) and (U1)–(U8) are that revision, if ϕ is compatible with E, should yield the same result as expansion $E + \phi$, which is not desirable for update in general, cf. [24]. On the other hand, (U8) says that if E can be decomposed into a disjunction of states (e.g., models), then each case can be updated separately and the overall results are formed by taking the disjunction of the emerging states.

Iterated Revision Darwiche and Pearl [10] have proposed postulates for iterated revision, which can be rephrased in our setting as follows (we omit parentheses in sequences $(E \star \phi_1) \star \phi_2$ of revisions):

(C1) If $\psi_2 \in Bel(\psi_1)$, then $Bel(E \star \psi_2 \star \psi_1) = Bel(E \star \psi_1)$.
(C2) If $\neg\psi_2 \in Bel(\psi_1)$, then $Bel(E \star \psi_1 \star \psi_2) = Bel(E \star \psi_2)$.
(C3) If $\psi_2 \in Bel(E \star \psi_1)$, then $\psi_2 \in Bel(E \star \psi_2 \star \psi_1)$.
(C4) If $\neg\psi_2 \notin Bel(E \star \psi_1)$, then $\neg\psi_2 \notin Bel(E \star \psi_2 \star \psi_1)$.
(C5) If $\neg\psi_2 \in Bel(E \star \psi_1)$ and $\psi_1 \notin Bel(E \star \psi_2)$, then $\psi_1 \notin Bel(E \star \psi_1 \star \psi_2)$.
(C6) If $\neg\psi_2 \in Bel(E \star \psi_1)$ and $\neg\psi_1 \in Bel(E \star \psi_2)$, then $\neg\psi_1 \in Bel(E \star \psi_1 \star \psi_2)$.

Another set of postulates for iterated revision, corresponding to a sequence E of observations, has been formulated by Lehmann [19]. Here each observation is a sentence which is assumed to be consistent (i.e., falsity is not observed), and the epistemic state E has an associated belief set $Bel(E)$. Lehmann's postulates read as follows, where E, E' denote sequences of observations and "," stands for concatenation:

(I1) $Bel(E)$ is a consistent belief set.
(I2) $\phi \in Bel(E, \phi)$.
(I3) If $\psi \in Bel(E, \phi)$, then $\phi \Rightarrow \psi \in Bel(E)$.
(I4) If $\phi \in Bel(E)$, then $Bel(E, \phi, E') = Bel(E, E)$.
(I5) If $\psi \vdash \phi$ then $Bel(E, \phi, \psi, E') = Bel(E, \psi, E')$.
(I6) If $\neg\psi \notin Bel(E, \phi)$, then $Bel(E, \phi, \psi, E') = Bel(E, \phi, \psi, E')$.
(I7) $Bel(E, \neg\phi, \phi) \subseteq Cn(E + \phi)$.

[1] A belief set K is *complete* iff, for each atom A, either $A \in K$ or $\neg A \in K$.

Postulate	Interpretation	Postulate holds
(K1)	(P_1, P_2) represents a belief set	yes
(K2), (U1)	$P_2 \subseteq Bel(P_1, P_2)$	yes
(U2)	$Bel(P_2) \subseteq Bel(P_1)$ implies $Bel(P_1, P_2) = Bel(P_1)$	no
(K3)	$Bel(P_1, P_2) \subseteq Bel(Bel(P_1) \cup P_2)$	yes
(U3)	If P_1 and P_2 are satisfiable, then (P_1, P_2) is satisfiable	no
(K4)	If $Bel(P_1) \cup P_2$ has a stable model, then $Bel(Bel(P_1) \cup P_2) \subseteq Bel(P_1, P_2)$	no
(K5)	(P_1, P_2) is unsatisfiable iff P_2 is unsatisfiable	no
(K6), (U4)	$P_1 \equiv P_1'$ and $P_2 \equiv P_2'$ implies $(P_1, P_2) \equiv (P_1', P_2')$	no
(K7), (U5)	$Bel(P_1, P_2 \cup P_3) \subseteq Bel(Bel(P_1, P_2) \cup P_3)$	yes
(U6)	If $Bel(P_3) \subseteq Bel(P_1, P_2)$ and $Bel(P_2) \subseteq Bel(P_1, P_3)$, then $Bel(P_1, P_2) = Bel(P_1, P_3)$	no
(K8)	If $Bel(P_1, P_2) \cup P_3$ is satisfiable then $Bel(Bel(P_1, P_2) \cup P_3) \subseteq Bel(P_1, P_2 \cup P_3)$	no

Table 1. Interpretation of Postulates (K1)–(K8) and (U1)–(U6).

Analysis of the Postulates In order to evaluate the different postulates, we need to adapt them for the setting of update programs. Naturally, the epistemic state $P = P_1, \ldots, P_n$ of an agent is subject to revision. However, the associated belief set $Bel(P)$ $(\subseteq \mathcal{L}_{\mathcal{A}})$ does not belong to a logical language closed under boolean connectives. Closing $\mathcal{L}_{\mathcal{A}}$ under conjunction does not cause much troubles, as the identification of finite GLPs with finite conjunctions of clauses permits that updates of a GLP P by a program P_1 can be viewed as the update of P with a single sentence from the underlying belief language. Ambiguities arise, however, with the interpretation of expansion, as well as the meaning of negation and disjunction of rules and programs, respectively.

Depending on whether the particular structure of the epistemic state E should be respected, different definitions of expansion are imaginable in our framework. At the "extensional" level of sentences, represented by a program or sequence of programs P, $Bel(P + P')$ is defined as $Bel(Bel(P) \cup P')$. At the "intensional" level of sequences $P = P_1, \ldots, P_n$, $Bel(P + P')$ could be defined as $Bel(P_1, \ldots, P_n \cup P')$. An intermediate approach would be defining $Bel(P + P') = Bel_{\mathcal{A}}(P_{\triangleleft} \cup P')$. We adopt the extensional view here. Note that, in general, adding P' to $Bel(P)$ does not amount to the semantical intersection of P' and $Bel(P)$ (nor of P and P', respectively).

As for negation, we might interpret the condition $\neg\phi \notin Bel(E)$ (or $\neg\psi \notin Bel(E \star \phi)$ in (K4) and (K8)) as satisfiability requirement for $E + \phi$ (or $(E \star \phi) + \psi$).

Disjunction $\vee$ of rules or programs (as epistemic states) appears to be meaningful only at the semantical level. The union $\mathcal{S}(P_1) \cup \mathcal{S}(P_2)$ of the sets of stable models of programs P_1 and P_2 may be represented syntactically through a program P_3, which in general requests an extended set of atoms. We thus do not consider the postulates involving $\vee$.

Given these considerations, Table 1 summarizes our interpretation of postulates (K1)–(K8) and (U1)–(U6), together with indicating whether the respective property holds or fails. We assume that P_1 is a nonempty sequence of GLPs.

Postulate	Interpretation	Postulate holds
(C1)	If $P_3 \subseteq Bel(P_2)$, then $Bel(P_1, P_3, P_2) = Bel(P_1, P_2)$	no
(C2)	If $S \not\models P_3$, for all $S \in \mathcal{S}(P_2)$, then $Bel(P_1, P_3, P_2) = Bel(P_1, P_2)$	no
(C3)	If $P_3 \subseteq Bel(P_1, P_2)$, then $P_3 \subseteq Bel(P_1, P_3, P_2)$	no
(C4)	If $S \models P_3$ for some $S \in \mathcal{S}(P_1, P_2)$, then $S \models P_3$ for some $S \in \mathcal{S}(P_1, P_3, P_2)$	yes
(C5)	If $S \not\models P_3$ for all $S \in \mathcal{S}(P_1, P_2)$ and $P_2 \not\subseteq Bel(P_1, P_3)$, then $P_2 \not\subseteq Bel(P_1, P_2, P_3)$	no
(C6)	If $S \not\models P_3$ for all $S \in \mathcal{S}(P_1, P_2)$ and $S \not\models P_2$ for all $S \in \mathcal{S}(P_1, P_3)$, then $S \not\models P_2$ for all $S \in \mathcal{S}(P_1, P_2, P_3)$	no
(I1)	$Bel(P_1)$ is a consistent belief set	no
(I2)	$P_2 \subseteq Bel(P_1, P_2)$	yes
(I3)	If $L_0 \leftarrow \in Bel(P_1, \{L_1, \ldots, L_k\})$, then $L_0 \leftarrow L_1, \ldots, L_k \in Bel(P_1)$	yes
(I4)	If $P_2 \subseteq Bel(P_1)$, then $Bel(P_1, P_2, P_3, \ldots, P_n) = Bel(P_1, P_3, \ldots, P_n)$	no
(I5)	If $Bel(P_3) \subseteq Bel(P_2)$, then $Bel(P_1, P_2, P_3, P_4, \ldots, P_n)$=$Bel(P_1, P_3, P_4, \ldots, P_n)$	no
(I6)	If $S \models P_3$ for some $S \in \mathcal{S}(P_1, P_2)$, then $Bel(P_1, P_2, P_3, P_4, \ldots, P_n) = Bel(P_1, P_2, P_2 \cup P_3, P_4, \ldots, P_n)$	no

Table 2. Interpretation of Postulates (C1)–(C6) and (I1)–(I6).

Thus, apart from very simple postulates, the majority of the adapted AGM and update postulates are violated by update programs. This holds even for the case where P_1 is a single program. In particular, $Bel(P_1, P_2)$ violates discriminating postulates such as (U2) for update and (K4) for revision. In the light of this, update programs neither have update nor revision flavor.

We remark that the picture does not change if we abandon extensional expansion and consider the postulates under intensional expansion. Thus, also under this view, update programs do not satisfy minimality of change.

The postulates (C1)–(C6) and (I1)–(I7) for iterated revision are treated in Table 2. Concerning Lehmann's [19] postulates, (I3) is considered as the pendant to AGM postulate K$\star$3. In a literal interpretation of (I3), we may, since the belief language associated with GLPs does not have implication, consider the case where ψ is a default literal L_0 and $\phi = L_1 \wedge \cdots \wedge L_k$ is a conjunction of literals L_i, such that $\phi \Rightarrow \psi$ corresponds to the rule $L_0 \leftarrow L_1, \ldots, L_k$. Since the negation of GLPs is not defined, we do not interpret (I7).

Note that, although postulate (C3) fails in general, it holds if P_3 contains a single rule. Thus, all of the above postulates except C4 fail, already if P_1 is a single logic program, and, with the exception of C3, each change is given by a single rule.

A question at this point is whether, after all, the various belief change postulates from above are meaningful for update programs.

We can view the epistemic state $P = P_1, \ldots, P_n$ of an agent as a prioritized belief base in the spirit of [7, 23, 6]. Revision with a new piece of information Q is accomplished by simply changing the epistemic state to $P = P_1, \ldots, P_n, Q$. The change of the belief base is then automatically accomplished by the nonmonotonic semantics of a sequence of logic programs. Under this view, updating logic programs amounts to an instance of the immediate revision approach.

On the other hand, referring to the update program, we may view the belief set of the agent represented through a pair $\langle P, \mathcal{A} \rangle$ of a logic program P and a (fixed) set of atoms $\mathcal{A}$, such that its belief set is given by $Bel_{\mathcal{A}}(P)$. Under this view, a new piece of information Q is incorporated into the belief set by producing a representation, $\langle P', \mathcal{A} \rangle$, of the new belief set, where $P' = P \lhd Q$. Here, (a set of) sentences from an extended belief language is used to characterize the new belief state, which is constructed by a nontrivial operation employing the semantics of logic programs. Thus, update programs enjoy to some extent also a logic-constrained revision flavor. Nonetheless, as also the failure of postulates shows, they are more an instance of *immediate* than *logic-constrained* revision. What we naturally expect, though, is that the two views described above amount to the same *at a technical level*. However, as we shall demonstrate below, this is not true in general.

4.2 Further Properties

Belief revision has been related in [14] to nonmonotonic logics by interpreting it as an abstract consequence relation on sentences, where the epistemic state is fixed. In the same way, we can interpret update programs as abstract consequence relation $\mid\!\sim$ on programs as follows. For a fixed epistemic state P and GLPs P_1 and P_2, we define

$$P_1 \mid\!\sim_P P_2 \text{ if and only if } P_2 \subseteq Bel(P, P_1),$$

i.e., if the rules P_2 are in the belief state of the agent after update of the epistemic state with P_1.

Various properties for nonmonotonic inference operations have been identified in the literature (see, e.g., [14]). Among them are *Cautious Monotonicity*, *Cut*, (*Left*) *Conjunction*, *Rational Cautious Monotonicity*, and *Equivalence*. Except for Cut, none of these properties hold. We recall that Cut denotes the following schema:

$$\frac{A \land B_1 \land \ldots \land B_m \mid\!\sim_P C \quad A \mid\!\sim_P B_1 \land \ldots \land B_m}{A \mid\!\sim_P C}$$

Additionally, we can also identify some very elemental properties which, as we believe, updates and sequences of updates should satisfy. The following list of properties is not developed in a systematic manner, though, and is by no means exhaustive. Update programs do enjoy, unless stated otherwise, these properties.

Addition of Tautologies: If the program P_2 contains only tautological clauses, then $(P_1, P_2) \equiv P_1$.

Initialization: $(\emptyset, P) \equiv P$.

Idempotence: $(P, P) \equiv P$.

Idempotence for Sequences: $(P_1, P_2, P_2) \equiv (P_1, P_2)$.
Update of Disjoint Programs: If $P = P_1 \cup P_2$ is a union of programs P_1, P_2 on disjoint alphabets, then $(P, P_3) \equiv (P_1, P_3) \cup (P_2, P_3)$.
Parallel updates: If P_2 and P_3 are programs defined over disjoint alphabets, then $(P_1, P_2) \cup (P_1, P_3) \equiv (P_1, P_2 \cup P_3)$. (*Fails.*)
Noninterference: If P_2 and P_3 are programs defined over disjoint alphabets, then $(P_1, P_2, P_3) \equiv (P_1, P_3, P_2)$.
Augmented update: If $P_2 \subseteq P_3$ then $(P_1, P_2, P_3) \equiv (P_1, P_3)$.

As mentioned before, a sequence of updates $P = P_1, \ldots, P_n$ can be viewed from the point of view of "immediate" revision or of "logic-constrained" revision. The following property, which deserves particular attention, expresses equivalence of these views (the property is formulated for the case $n = 3$):

Iterativity: For any epistemic state P_1 and GLPs P_2 and P_3, it holds that $P_1 \lhd P_2 \lhd P_3 \equiv_{\mathcal{A}} (P_1 \lhd P_2) \lhd P_3$.

However, this property fails. Informally, soundness of this property would mean that a sequence of three updates is a shorthand for iterated update of a single program, i.e., the result of $P_1 \lhd P_2$ is viewed as a singleton sequence. Stated another way, this property would mean that the definition for $P_1 \lhd P_2 \lhd P_3$ can be viewed as a shorthand for the nested case. Vice versa, this property reads as possibility to forget an update once and for all, by incorporating it immediately into the current belief set.

For a concrete counterexample, consider $P_1 = \emptyset$, $P_2 = \{a \leftarrow\ , \mathit{not}\, a \leftarrow\ \}$, $P_3 = \{a \leftarrow\ \}$. The program $P_{\lhd} = P_1 \lhd P_2 \lhd P_3$ has a unique stable model, in which a is true. On the other hand, $(P_1 \lhd P_2) \lhd P_3$ has no stable model. Informally, while the "local" inconsistency of P_2 is removed in $P_1 \lhd P_2 \lhd P_3$ by rejection of the rule $\mathit{not}\, a \leftarrow$ via P_3, a similar rejection in $(P_1 \lhd P_2) \lhd P_3$ is blocked because of a renaming of the predicates in $P_1 \lhd P_2$. The local inconsistency of P_2 is thus not eliminated.

However, under certain conditions, which exclude such possibilities for local inconsistencies, the iterativity property holds, given by the following result:

Theorem 4.1. *Let $P = P_1, \ldots, P_n$, $n \geq 2$, be an update sequence on a set of atoms $\mathcal{A}$. Suppose that, for any rules $r_1, r_2 \in P_i$, $i \leq n$, such that $H(r_1) = \mathit{not}\, H(r_2)$, the union $B(r_1) \cup B(r_2)$ of their bodies is unsatisfiable. Then:*

$$(\cdots(P_1 \lhd P_2) \lhd P_3) \cdots \lhd P_{n-1}) \lhd P_n \equiv_{\mathcal{A}} P_1 \lhd P_2 \lhd P_3 \lhd \cdots \lhd P_n.$$

5 Refined Semantics and Extensions

Minimal and strict stable models Even if we abandon the AGM view, update programs do intuitively not respect minimality of change, as a new set of rules P_2 should be incorporated into an existing program P_1 with as little change as possible.

It appears natural to measure change in terms of the set of rules in P_1 which are abandoned. This leads us to prefer a stable model S_1 of $P = P_1, P_2$ over another stable model S_2 if S_1 satisfies a larger set of rules from P_1 than S_2.

Definition 5.1. *Let $P = P_1, \ldots, P_n$ be a sequence of GLPs. A stable model $S \in \mathcal{U}(P)$ is minimal iff there is no $T \in \mathcal{U}(P)$ such that $Rej(T, P) \subset Rej(S, P)$.*

Example 5.1. Consider $P_1 = \{r_1 : \; not\, a \leftarrow \; \}$, $P_2 = \{r_2 : \; a \leftarrow not\, c\}$, and $P_3 = \{r_3 : \; c \leftarrow not\, d, \, r_4 : \; d \leftarrow not\, c\;\}$. Then (P_1, P_2) has the single stable model $\{a\}$, which rejects the rule in P_1. The sequence (P_1, P_2, P_3) has two stable models: $S_1 = \{c\}$ and $S_2 = \{a, d\}$. S_1 rejects no rule, while S_2 rejects the rule r_1. Thus, S_1 is preferred to S_2 and S_1 is minimal.

Minimal stable models put no further emphasis on the temporal order of updates. Rules in more recent updates may be violated in order to satisfy rules from previous updates. Eliminating this leads us to the following notion.

Definition 5.2. *Let $S, S' \in \mathcal{U}(P)$ for an update sequence $P = P_1, \ldots, P_n$. Then, S is preferred to S' iff some $i \in \{1, \ldots, n\}$ exists such that (1) $Rej_i(S, P) \subset Rej_i(S', P)$, and (2) $Rej_j(S', P) = Rej_j(S, P)$, for all $j = i+1, \ldots, n$. A stable model S of P is* strict, *if no $S' \in \mathcal{U}(P)$ exists which is preferred to S.*

Example 5.2. Consider $P = P_1, P_2, P_3, P_4$, where $P_1 = \{r_1 : \; not\, a \leftarrow \; \}$, $P_2 = \{r_2 : \; a \leftarrow not\, c\}$, $P_3 = \{r_3 : not\, c \leftarrow \; \}$, and $P_4 = \{r_4 : \; c \leftarrow not\, d, \, r_5 : \; d \leftarrow not\, c\;\}$. Then, P has two stable models, namely $S_1 = \{c\}$ and $S_2 = \{a, d\}$. We have $Rej(S_1, P) = \{r_3\}$ and $Rej(S_2, P) = \{r_1\}$. Thus, $Rej(S_1, P)$ and $Rej(S_2, P)$ are incomparable, and hence both S_1 and S_2 are minimal stable models. However, compared to S_2 in S_1 the more recent rule of P_3 is violated. Thus, S_2 is the unique strict stable model.

Clearly every strict stable model is minimal, but not vice versa. Unsurprisingly, minimal and strict stable models do not satisfy AGM minimality of change.

The trade-off for epistemic appeal is higher computational complexity than for arbitrary stable models. Let $Bel_{min}(P)$ (resp., $Bel_{str}(P)$) be the set of beliefs induced by the collection of minimal (resp., strict) stable models of $P = P_1, \ldots, P_n$.

Theorem 5.1. *Given a sequence of programs $P = P_1, P_2, \ldots, P_n$ over a set of atoms $\mathcal{A}$, deciding whether*

1. *P has a stable model is* NP*-complete;*
2. *$L \in Bel(P)$ for a given literal L is* coNP*-complete;*
3. *$L \in Bel_{min}(P)$ (resp. $L \in Bel_{str}(P)$) for a given literal L is Π_2^P-complete.*

Similar results have been derived by Inoue and Sakama [17]. The complexity results imply that minimal and strict stable models can be polynomially translated into disjunctive logic programming, which is currently under investigation.

Strong negation Update programs can be easily extended to the setting of generalized extended logic programs (GELPs), which have besides not also strong negation $\neg$ as in [21]. Viewing, for $A \in \mathcal{A}$, the formula $\neg A$ as a fresh atom, the rules $not\, A \leftarrow \neg A$ and $not\, \neg A \leftarrow A$ emulate the interpretation of $\neg$ in answer set semantics (cf., e.g., [2]). More precisely, the consistent answer sets of a GELP P correspond one-to-one to the stable models of $P^{\neg}$, which is P augmented with the emulation rules for $\neg A$. Answer sets of a sequence of GELPs $P = P_1, \ldots, P_n$ can then be defined through this correspondence in terms of the stable models of $P^{\neg} = P_1^{\neg}, \ldots, P_n^{\neg}$, such that $Bel(P) = Bel(P^{\neg})$.

Like for dynamic logic programs [3], $P^{\neg}$ can be simplified by removing some of the emulation rules. Let $CR(P)$ be the set of all emulation rules for atoms A such that $\neg A$ occurs in some rule head of P.

Theorem 5.2. *For any sequence of GELPs $P = P_1, \ldots, P_n$ over $\mathcal{A}$, $S \subseteq \mathcal{A} \cup \{\neg A \mid A \in \mathcal{A}\}$ is an answer set of P iff $S \in \mathcal{U}(P_1, \ldots, P_{n-1}, P_n \cup CR(P))$.*

First-order programs The semantics of a sequence $P = P_1, \ldots, P_n$ of first-order GLPs, i.e., where $\mathcal{A}$ consists of nonground atoms in a first order-language, is reduced to the ground case by defining it in terms of the sequence of instantiated programs $P^* = P_1^*, \ldots, P_n^*$ over the Herbrand universe of P as usual. That is, $\mathcal{U}(P) = \mathcal{U}(P^*)$. The definition of update program $P_{\triangleleft}$ can be easily generalized to non-ground programs, such that $P_{\triangleleft} \equiv P^*{}_{\triangleleft}$, i.e., $P_{\triangleleft}$ faithfully represents the update program for P^*.

6 Related Work

Dynamic logic programming Recall that our update programs syntactically redefine dynamic logic programs for update in [2, 5], which generalize the idea of updating interpretations through revision programs [22]. As we feel, they more transparently reflect the working behind this approach.

The major difference between our update programs and dynamic logic programs is that the latter determine the values of atoms from the bottom level P_1 *upwards* towards P_n, using interia rules, while update programs determine the values in a downward fashion.

Denote by $\oplus P = P_1 \oplus \cdots \oplus P_n$ the dynamic logic program of [2] for updating P_1 with $P_2, \ldots, P_n$ over atoms $\mathcal{A}$, which is a GLP over atoms $\mathcal{A}_{dyn} \supseteq \mathcal{A}$. For any model M of P_n in $\mathcal{A}$, let

$$\begin{aligned} \mathit{Rejected}(M,P) &= \textstyle\bigcup_{i=1}^{n}\{r \in P_i \mid \exists r' \in P_j,\ \text{for some } j \in \{i+1,\ldots,n\},\ \text{such} \\ &\qquad\qquad \text{that } H(r') = \mathit{not}\, H(r)\ \wedge\ S \models B(r) \cup B(r')\}, \\ \mathit{Defaults}(M,P) &= \{\mathit{not}\, A \mid \forall r \in P : H(r) = A \Rightarrow M \not\models B(r)\}. \end{aligned}$$

Stable models of $\oplus P$, projected to $\mathcal{A}$, are semantically characterized as follows.

Definition 6.1. *For a sequence $P = P_1, \ldots, P_n$ of GLPs over atoms $\mathcal{A}$, an interpretation $N \subseteq \mathcal{A}_{dyn}$ is a stable model of $\oplus P$ iff $M = N \cap \mathcal{A}$ is a model of U such that*

$$M = lm(P \setminus \mathit{Rejected}(M,P) \cup \mathit{Defaults}(M,P)).$$

Here, literals $\mathit{not}\, A$ are considered as new atoms, where implicitly the constraint $\leftarrow A, \mathit{not}\, A$ is added. Let us call any such M a *dynamic stable model* of P.

As one can see, we may replace $Rej(S,P)$ in Theorem 3.3 by $\mathit{Rejected}(S,P)$ and add all rules in $\mathit{Defaults}(S,P)$, as they vanish in the reduction by S. However, this implies that update and dynamic stable models coincide.

Theorem 6.1. *For any sequence $P = P_1, \ldots, P_n$ of GLPs over atoms $\mathcal{A}$, $S \subseteq \mathcal{A}$ is a dynamic stable model of P iff $S \in \mathcal{U}(P)$.*

Inheritance programs A framework for logic programs with inheritance is introduced in [9]. In a hierarchy of objects $o_1, \ldots, o_n$, represented by a disjunctive extended logic program $P_1, \ldots, P_n$ [15], possible conflicts in determining the properties of o_i are resolved by favoring rules which are more specific according to the hierarchy, which is given by a (strict) partial order $<$ over the objects.

If we identify o_i with the indexed program P_i, an *inheritance program* consists of a set $P = \{P_1, \ldots, P_n\}$ of programs over atoms $\mathcal{A}$ and a partial order $<$ on P. The program $\mathcal{P}(P_i)$ for P_i (as an object) is given by $\mathcal{P}(P_i) = \{P_i\} \cup \{P_j \mid P_i < P_j\}$, i.e., the collection of programs at and above P_i.

The semantics of $\mathcal{P}(P_i)$ is defined in terms of answer sets. In the rest of this section, we assume that any program $P_i \in \mathcal{P}$ is disjunction-free and we simplify definitions in [9] accordingly. Let, for each literal L of form A or $\neg A$, denote $\neg L$ its opposite, and let $Lit_{\mathcal{A}} = \mathcal{A} \cup \{\neg A \mid A \in \mathcal{A}\}$.

Definition 6.2. *Let $I \subseteq Lit_{\mathcal{A}}$ be an interpretation and $r \in P_j$. Then, r is overridden in I, if (1) $I \models B(r)$, (2) $\neg H(r) \in I$, and (3) there exists a rule $r_1 \in P_i$ for some $P_i < P_j$ such that $H(r_1) = \neg H(r)$.*

An interpretation $I \subseteq Lit_{\mathcal{A}}$ is a model of $\mathcal{P}$, if I satisfies all non-overridden rules in $\mathcal{P}$ and the constraint $\leftarrow A, not\ A$ for each atom $A \in \mathcal{A}$; moreover, I is minimal if it is the least model of all these rules. Answer sets are now as follows.

Definition 6.3. *A model M of $\mathcal{P} = \mathcal{P}(P_i)$, is a $DLP^{<}$-answer set of $\mathcal{P}$ iff M is a minimal model of $\mathcal{P}^M$, where $\mathcal{P}^M = \{r \in \mathcal{P} \mid r \text{ is not overridden in } M\}^M$ is the reduct of $\mathcal{P}$ by M.*

It is natural to view an update sequence $P = P_1, \ldots, P_n$ as an inheritance program where later updates are considered more specific. That is, we might view P as an inheritance program $P_n < P_{n-1} < \ldots < P_1$. It appears that the latter is in fact equivalent to the update program $P_1 \lhd \ldots \lhd P_n$.

For a sequence of GLPs $P = P_1, \ldots, P_n$ over $\mathcal{A}$, define the inheritance program $\mathcal{Q} = Q_n < Q_{n-1} < \cdots < Q_1$ as follows. Let P_i^- be the program resulting from P_i by replacing in rule heads the default negation not through $\neg$. Define $Q_1 = P_1^- \cup \{\neg A \leftarrow not\ A \mid A \in \mathcal{A}\}$ and $Q_j = P_j^-$, for $j = 2, \ldots, n$. Then we have the following.

Theorem 6.2. *Let $P = P_1, \ldots, P_n$ be a sequence of GLPs over atoms $\mathcal{A}$. Then, $S \in \mathcal{U}(P)$ iff $S \cup \{\neg A \mid A \in \mathcal{A} \setminus S\}$ is a DLP$^{<}$-answer set of $\mathcal{Q}(P_1, \ldots, P_n)$.*

Conversely, linear inheritance programs yield the same result as update programs in the extension with classical negation.

Theorem 6.3. *Let $P = P_1 < \cdots < P_n$ be an inheritance program over atoms $\mathcal{A}$. Then, S is a DLP$^{<}$-answer set of P iff S is an answer set of the sequence of GELPs $P_n, P_{n-1}, \ldots, P_1$.*

Thus, dynamic logic programs and inheritance programs are equivalent.

Program updates through abduction On the basis of their notion of *extended abduction*, Inoue and Sakama [17] define a framework for various update problems. The most general is *theory update*, which is update of an extended logic program (ELP) P_1 by another such program P_2. Informally, an abductive update of P_1 by P_2 is a largest consistent program P' such that $P_1 \subseteq P' \subseteq P_1 \cup P_2$ holds. This is formally captured in [17] by reducing the update problem to computing a minimal set of abducible rules $Q \subseteq P_1 \setminus P_2$ such that $(P_1 \cup P_2) \setminus Q$ is consistent. In terms of [16], $P_1 \cup P_2$ is considered for abduction where the rules in $P_1 \setminus P_2$ are abducible, and the intended update is realized via a minimal *anti-explanation* for falsity, which removes abducible rules to restore consistency.

While this looks similar to our minimal updates, there is a salient difference: abductive update does not respect *causal rejection*. A rule r from $P_1 \setminus P_2$ may be rejected even if no rule r' P_2 fires whose head contradicts applying r. For example, consider $P_1 = \{q \leftarrow\ , \neg q \leftarrow a\}$ and $P_2 = \{a \leftarrow\ \}$. Both P_1 and P_2 have consistent answer sets, while (P_1, P_2) has no stable model. In Inoue and Sakama's approach, one of the two rules in P_1 will be removed. Note that contradiction removal in a program P occurs as a special case ($P_1 = P$, $P_2 = \emptyset$).

Abductive updates are, due to inherent minimality of change, harder than update programs; some abductive reasoning problems are Σ_2^P-complete [17].

Updates through priorities Zhang and Foo [13] define update of an ELP P_1 by an ELP P_2 based on their work on preferences [12] as a two-step approach: In Step 1, each answer set S of P_1 is updated to a closest answer set S' of P_2, where distance is in terms of the set of atoms on which S,S' disagree and closeness is set inclusion. Then, a maximal set $Q \subseteq P_1$ is chosen such that $P_3 = P_2 \cup Q$ has an answer set containing S'. In Step 2, the answer sets of P_3 are computed using priorities, where rules of P_2 have higher priority than rules of Q.

This approach is different from ours. It is in the spirit of the *possible models approach* [24], which updates models of a propositional theory separately, thus satisfying the update postulate U8. However, like in Inoue and Sakama's approach, rules are not removed on the basis of causal rejection. In particular, the same result is obtained on the example there. Step 2 indicates a strong update flavor of the approach, since rules are unnecessarily abandoned. For example, update of $P_1 = \{p \leftarrow \mathit{not}\, q\}$ with $P_2 = \{q \leftarrow \mathit{not}\, p\}$ results in P_2, even though $P_1 \cup P_2$ is consistent. Since the result of an update leads to a set of programs, in general, naive handling of updates requires exponential space.

7 Conclusion

We have considered the approach to updating logic programs based on dynamic logic programs [2, 3] and investigated various properties of this approach. Comparing it to other approaches and related work, we found that it is equivalent to a fragment of inheritance programs in [9].

Several issues remain for further work. A natural issue is the inverse of addition, i.e. retraction of rules from a logic program. Dynamic logic programming evolved into

LUPS [3], which is a language for specifying update behavior in terms of addition and retraction of sets of rules to a logic program. LUPS is generic, however, as in principle, different approaches to updating logic programs could provide the semantical basis for an update step. Exploring properties of the general framework, as well as of particular such instantiations, would be worthwhile. Furthermore, reasoning about update programs describing the behavior of agents programmed in LUPS is an interesting issue.

Another issue are postulates for update operators on logic programs and, more generally, on nonmonotonic theories. As we have seen, several postulates from the area of logical theory change fail for dynamic logic programs (see [8] for related observations). This may partly be explained by nonmonotonicity of stable semantics and the dominant role of syntax for update embodied by causal rejection. However, similar features are not exceptional in the context of logic programming. It would be interesting to know further postulates and desiderata for update of logic programs besides the ones considered here, and an AGM style characterization of update operators compliant with them.

Acknowledgments. This work was partially supported by the Austrian Science Fund (FWF) under grants P13871-INF and N Z29-INF.

References

1. C. Alchourrón, P. Gärdenfors, and D. Makinson. On the Logic of Theory Change: Partial Meet Functions for Contraction and Revision. *Journal of Symbolic Logic*, 50:510–530, 1985.
2. J. Alferes, J. Leite, L. Pereira, H. Przymusinska, and T. Przymusinski. Dynamic Logic Programming. In A. Cohn and L. Schubert, editors, *Proc. KR'98*, pp. 98–109. Morgan Kaufmann, 1998.
3. J. Alferes, J. Leite, L. Pereira, H. Przymusinska, and T. C. Przymusinski. Dynamic Updates of Non-Monotonic Knowledge Bases. *Journal of Logic Programming*, 2000. To appear.
4. J. Alferes and L. Pereira. Updates plus Preferences. In *Proc. JELIA 2000*, LNCS, this volume. Springer, 2000.
5. J. Alferes, L. Pereira, H. Przymusinska, and T. Przymusinski. Lups: A Language for Updating Logic Programs. In *Proc. LPNMR'99*, LNAI 1730, pp. 162–176. Springer, 1999.
6. S. Benferhat, C. Cayrol, D. Dubois, J. Lang, and H. Prade. Inconsistency Management and Prioritized Syntax-Based Entailment. In *Proc. IJCAI-93*, pp. 640–645. Morgan Kaufman, 1993.
7. G. Brewka. Belief Revision as a Framework for Default Reasoning. In *The Logic of Theory Change*, LNAI 465, pp. 206–222. Springer, 1991.
8. G. Brewka. Declarative Representation of Revision Strategies. In *Proc. NMR '2000*, 2000.
9. F. Buccafurri, W. Faber, and N. Leone. Disjunctive Logic Programs with Inheritance. In *Proc. ICLP '99*, pp. 79–93, 1999. The MIT Press. Full version available as Technical Report DBAI-TR-99-30, Institut für Informationssysteme, Technische Universität Wien, Austria, May 1999.
10. A. Darwiche and J. Pearl. On the Logic of Iterated Belief Revision. *Artificial Intelligence*, 89(1-2):1-29, 1997.
11. F. Sadri and F. Toni. Computational Logic and Multiagent Systems: A Roadmap. Available from the ESPRIT Network of Excellence in Computational Logic (Compulog Net), http://www.compulog.org/, 1999.

12. N. Foo and Y. Zhang. Towards Generalized Rule-based Updates. In *Proc. IJCAI'97*, pp. 82–88. Morgan Kaufmann, 1997.
13. N. Foo and Y. Zhang. Updating Logic Programs. In *Proc. ECAI'98*, pp. 403-407. Wiley, 1998.
14. P. Gärdenfors and H. Rott. Belief Revision. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook of logic in Artificial Intelligence and Logic Programming*. Oxford Science Publications, 1995.
15. M. Gelfond and V. Lifschitz. Classical Negation in Logic Programs and Disjunctive Databases. *New Generation Computing*, 9:365–385, 1991.
16. K. Inoue and C. Sakama. Abductive Framework for Nonomonotonic Theory Change. In *Proc. IJCAI'95*, pp. 204–210. Morgan Kaufmann, 1995.
17. K. Inoue and C. Sakama. Updating Extended Logic Programs Through Abduction. In *Proc. LPNMR'99*, LNAI 1730, pp. 147–161. Springer, 1999.
18. H. Katsuno and A. Mendelzon. On the Difference Between Updating a Knowledge Database and Revising it. In *Proc. KR'91*, pp. 387–394, 1991.
19. D. Lehmann. Belief Revision, Revised. In *Proc. IJCAI'95*, pp. 1534–1540. Morgan Kaufmann, 1995.
20. J. Leite, J. Alferes, and L. Pereira. Multi-Dimensional Dynamic Logic Programming. In *Proc. Workshop on Computational Logic in Multi-Agent Systems (CLIMA)*, London, UK, July 2000.
21. V. Lifschitz and T. Y. C. Woo. Answer Sets in General Nonmonotonic Reasoning (Preliminary Report). In *Proc. KR '92*, pp. 603–614. Morgan Kaufmann, 1992.
22. V. W. Marek and M. Truszczyński. Revision Specifications by Means of Programs. In *Proc. JELIA'94*, LNAI 838, pp. 122–136. Springer, 1994.
23. B. Nebel. Belief Revision and Default Reasoning: Syntax-Based Approaches. In *Proc. KR'91*, pp. 417–428, 1991.
24. M. Winslett. Reasoning about Action Using a Possible Models Approach. In *Proc. AAAI'88*, pp. 89–93, 1988.