

Available from: http://dx.doi.org/10.1007/978-3-540-39813-4_19
A syntax-based approach to reasoning about action and belief update

Quoc Bao Vo\(^1\), Abhaya Nayak\(^2\) and Norman Foo\(^3\)

\(^1\)FR Informatik
Universität des Saarlandes, 66041 Saarbrücken, Germany
Phone: +49-681-302-4163 Fax: +49-681-302-5076
bao@ags.uni-sb.de

\(^2\)Department of Computing,
Division of Information and Communication Sciences
Macquarie University, Sydney, NSW 2109, Australia
Phone: +61 2 9850 9565 Fax: +61 2 9850 9551
abhaya@ics.mq.edu.au

\(^3\)School of Computer Science and Engineering
University of New South Wales, Sydney, NSW 2052, Australia
Phone: +61 2 9385 6921/6920 Fax: +61 2 9385 4936/5995
norman@cse.unsw.edu.au

Abstract

In this paper, we introduce an alternative approach to reasoning about action. The approach provides a solution to the frame and the ramification problem in a uniform manner. The approach involves keeping a (syntax-based) model of the world that is updated when actions are performed. Our approach is similar to the STRIPS system in which formulas are deleted and added as effects of an action. The presented framework however does not suffer from STRIPS’ limitations in expressivity.

**Keywords:** reasoning about action, belief update, knowledge representation, STRIPS-based representation, syntactical approach

1 Introduction

Reasoning about action has been one of the classic areas of artificial intelligence (AI) research (see [16]). One of the main goals for research in the area is a good formalism to capture rational behaviour and common sense for automated agents. Other implications reach as far as database applications,
e.g. database update operators, or system and control theories. As the foundation towards achieving rational behaviours, such a formalism should play a major role in planning. As such, a major problem identified by researchers in the area has been the motivation for the invention of one of the most successful and influential planners, viz. the STRIPS system [3].

Researchers in the community have since then searched for a good solution to their infamous problems, e.g. the frame, the qualification and the ramification problem. Many kinds of logic have been invented including several versions of the Situation Calculus (see [19] and the references therein), the Event Calculus (see [22] and the references therein), Dynamic Logic (see e.g. [25]), Action languages (see [5] and the references therein), etc. Most of them rest on the same base: the frame problem is inherent for any kind of logical axiomatisation. Thus, for any logical approach to reasoning about action to be successful, a good axiomatisation must be discovered. Such an axiomatisation should overcome or at least get around the frame problem. To most people, this could be accomplished through adding a set of axioms, or an axiom, to some base formalisation. Such axioms are called the frame axioms. Depending on the underlying logics, these axioms can be realised as propositional, first-order, or even second-order axioms (e.g. in Situation Calculus, or Event Calculus), or as inference rules (e.g. in default logic, logic programming). The introduction of frame axioms could result in a quite complex representation of the domain under consideration and subsequently incur a great deal of computation. For instance, there has been a long history of formalising the frame axioms in the Situation Calculus and the Event Calculus with early attempts by John McCarthy [14, 15] being proved to be incorrect by Hanks and McDermott [7] and further efforts being invested by Baker [1], Haas [6], and Pednault [17] and Schubert [21] amongst many others, resulting in (more or less) acceptable solutions reported by Shanahan [22] and Reiter [18].

Alternatively, a number of researchers have employed the operational semantics of actions and causation to compute the effects of actions. These are well known as the state based formalisms. In particular, a transition system or an automaton will be introduced to capture the dynamics of the system under consideration. By maintaining the set of states that are compatible with its knowledge, the agent can reason about the resulting states after performing an action. The problems with this approach include: the size of the state space can be huge, it may involve checking a large number of states if the agent’s knowledge is very incomplete, and the problem of expanding the language vocabulary to incorporate new concepts. As a result, many current planners still employ the STRIPS formulations of planning problems, or some variant of of these formulations such as ADL [17]. Note that STRIPS also constitutes the core of PDDL (Planning Domain Definition Language), the official language used in the AIPS planning competitions.

In this paper we introduce a STRIPS-like representation and computation machinery. The agent represents what it believes as a set of sentences. The underlying logic which can be monotonic or non-monotonic tells the agent what can be derived from a given theory. The task of reasoning about actions is carried out through a STRIPS-like mechanism by removing certain formulas from the current theory before adding the updated information to form the updated theory. The process is provably correct relative to a state-based representation. With this simplicity, this hopefully provides people in the planning community with a representation and reasoning machinery to serve as the basis for an efficient planner. On the other hand, we don’t have to sacrifice the expressiveness in representation which has been suffered in STRIPS.

The rest of the paper is organised as follows. Some logical preliminaries are presented in the next section. Then in Section 3 we will discuss our major contribution in this paper, viz. a simple framework for reasoning about dynamic domains, through which a simple solution to the frame problem will also be outlined. As our approach is based on a formalisation of (partial) state update, it shares many commonalities with formalisms which are based on transition systems. We thus
show the connection from our approach to one such formalism, viz. the action language $A$ and also discuss how the ramification problem is straightforwardly dealt with in our framework in Section 4. In Section 5 we discuss how our framework can be interpreted as a belief update operator and consider the related work in the literature of belief update research. Section 6 concludes the paper with a summary about the framework described in this paper. All the proofs of the results presented in the paper are deferred to the Appendix.

2 Logical Preliminaries

We shall be dealing with the finite propositional language $L_0$ consisting of the well formed formulas (wffs) over an alphabet $P = \{p, q, r, \ldots\}$ of propositional letters. We assume the availability of the logical constants $\land, \lor, \lnot, \text{false, true}$ while the other connectives, e.g. $\supset, \equiv$, will be considered as shorthands of more elaborate expressions. $L, L_1, \ldots$ denote literals. Clauses are disjunctions of literals. We will also interchangeably express clauses as sets of literals. Thus, $L_2 \lor L_5 \lor L_6 = \{L_2, L_5, L_6\} = L_2 \cup \{L_5, L_6\}$. $\varphi, \psi, \chi, \ldots$ denote clauses and formulas. Given a set of formulas $S$, we denote by $Cn(S)$ the logical closure of $S$, i.e. the set of all formulas that are logically entailed by $S$.

The following convention is also assumed throughout: If $L$ is a literal, then by $|L|$ we denote its affirmative component, i.e. $|p| = \lnot p = p$ where $p \in P$. We are now introducing several notions to be used in the rest of the paper.

(atm) We denote by $atm(\varphi)$ the set of atoms occurring in the formula $\varphi$. For example, $atm(\lnot p \lor (p \land q)) = \{p, q\}$, $atm(\text{false}) = atm(\text{true}) = \emptyset$.

(theory) Given the language $L_0$, we define a theory to be a finite set of clauses. We can also treat a theory as a formula by taking the conjunction of its clauses.

(partial interpretation) Let $S \subseteq P$ be given. A partial interpretation $I$ over $S$ is a function from $S$ to $\{\text{false, true}\}$. We sometimes represent a partial interpretation $I$ as a set of literals whose atoms are members of $S$ such that for each $p \in S$, (i) $p \in I$ iff $I(p) = \text{true}$ and (i) $\lnot p \in I$ iff $I(p) = \text{false}$.

(interpretation) A complete interpretation, or interpretation, is a partial interpretation over $P$. We denote by $Mod(\varphi)$ the set of all interpretations that satisfy the formula $\varphi$.

3 A simple approach to reasoning about dynamic domains

3.1 Intuition

Given a set of formulas modelling the agent’s beliefs about the dynamic world, the agent wants to reason about the outcome of an event $e$.\(^1\) If the agent is to represent the domain as a set of logical formulas, the problem of axiomatising the effects of an event becomes an interesting issue. A theorem prover can only show that an action of picking up a box does not change the colour of that box, does not change the position of the house, and neither does it create a nuclear war, etc. if these (non) effects are explicitly axiomatised in the action description. This is well known as the frame problem. The most sought after approach to this problem has been to augment the core axiomatisation with the so-called frame axioms to allow the theorem prover to arrive at the conclusions about unchanged fluents after an action. This has led to solutions, e.g. by Reiter [18], Lin and Shoham [12], etc.

\(^1\)We consider actions as a special case of events. Accordingly, the execution of an action corresponds to an occurrence of the corresponding event.
Another approach is to accomplish the task of reasoning about the interesting actions using a special
procedure coupled with a suitable data structure for representing the action descriptions. This has
been the core of the STRIPS planner [3]. In this approach, each state of the world is expressed as
a set of formulas. An event transforms the world from one state to another. The agent is normally
given the initial state. To discover which state the world would become after an occurrence of an
event $E$ in a state $\sigma$, the agent uses the event description of $E$. An event description consists of a
precondition which is a formula, and an add list and a delete list which are sets of formulas. If the
precondition is satisfied in $\sigma$, the next state will be the result of removing all formulas in the delete
list from $\sigma$ and then adding all formulas in the add list to the resulting set of formulas. Lifschitz [10]
has shown that this approach only works under severe restrictions, i.e. when the formulas allowed
to represent the states, and the add and the delete lists are members of a set of specially designed
formulas. For instance, only atomic formulas will be allowed.

While we enjoy the simplicity of the STRIPS approach, we are annoyed by the restrictions on its
expressivity which would severely limit the actions the agent wants to represent. However, a simple
modification to this approach can lead to a good solution to the frame problem without sacrificing
the simplicity behind the STRIPS approach. Informally, we represent an event with its precondition
and postcondition which are arbitrary formulas. There may even be multiple event descriptions for
a single event depending on their preconditions. A belief state, or state, will be represented as a
finite set of arbitrary formulas.\footnote{Note that a belief state is not required to be consistent. A belief state represents the agent’s beliefs about the world and may contain an inconsistent set of formulas. While abbreviating belief states by calling them states may be misleading, this practice help the reader relate our framework to our source of inspiration, viz. STRIPS.}

Observe that theories are just a special case of belief states in which
all formulas are normalised in CNF. Thus we may interchangeably use a set of clauses to refer to a
formula or a finite set of formulas, and refer to it either as a theory or as a state.

Now, let an event $E$ be performed in a state $\sigma$. If the precondition of $E$ is satisfied in $\sigma$, we
would like to delete several clauses in $\sigma$ before adding the postcondition of $E$ into the resulting set
of clauses. Which clauses should we delete from $\sigma$? Under some conditions, the answer is as simple
as: If $\Delta$ is the set of atoms of the postcondition then just delete all clauses whose sets of atoms have
non-empty intersection with $\Delta$. Intuitively, every atom which occurs in the postcondition of the
event $E$ potentially holds the value dictated in the postcondition of $E$, at least in one of the possible
next states in case there are disjunctions in the postcondition of $E$. Thus the clauses in $\sigma$ that contain
those atoms should not be considered worth believing in the next state. In fact the only case when
such a clause, called $c$, still maintains its status in the next state is when there exists some clause $c'$
such that $c' \subseteq c$ and $c'$ is derivable from the postcondition. In that case, even though $c$ is deleted
from $\sigma$ in the first place, it will be restored in the next state as it is contained in the postcondition of
event $E$. We remark that the reason for $c$ to be in the next state is not because it is worth retaining
from the previous state but rather because it is in the postcondition of the event $E$.

To briefly summarise, our formalism centers around a syntactical approach to systematic detec-
tion of relevant formulas (with respect to a particular action or event) in the current belief state.
Such relevant formulas must be removed from the theory representing the current state before the
postconditions of the action/event are to be added to obtain the next belief state. The set of atoms
$\Delta$ discussed above provides the connection between the postconditions of the action or event with
the “relevant” formulas from the theory representing the current state. We ignore the domain con-
straints, i.e. formulas that hold in all states, for now. We’ll show how our framework is extended to
deal with indirect effects and domain constraints later.

For example, let $\sigma = \{s, t, r, p \lor q\}$ represent the agent’s current beliefs about the state of the
world and consider an event $E$ whose precondition is true and postcondition is $\{p \lor r\}$ then our
framework allows the agent to form her beliefs about the resulting state after performing event $E$ as follows:

1. Since the precondition of $E$ holds in $\sigma$, the postcondition of event $E$, i.e. $p \lor r$, is expected to be take effect. As a consequence, the current beliefs $r$ and $p \lor q$ must be deleted from the agent’s belief set $\sigma$ leaving her with only two beliefs $s$ and $t$;

2. Now, the effect of event $E$, namely $p \lor r$, is added to her belief set allowing her to come up with the following belief set for the resulting state: $\sigma' = (\sigma \setminus \{r, p \lor q\}) \cup \{p \lor r\} = \{s, t, p \lor r\}$.

Note that, because the effect of event $E$ is $p \lor r$, there are possible next states in which $r$ does not hold and the other in which $p \lor q$ does not hold.

Is there a condition for this approach to work correctly? Yes, there is one such condition. The following example shows just that. Assume that $\sigma = \{p, \neg p \lor q\}$ and the precondition of the event $E$ is $p$ and the postcondition of event $E$ is also $p$ then following the above approach, the next state $\sigma' = \{p\}$. However, intuitively we should have $\{p, q\}$ as the next state. The lesson learnt from this example is that the clause $\{q\}$ is a logical consequence of $\sigma$ which happens to be independent of the postcondition of event $E$. Thus, a sound formalism is required to find all such independent clauses and a small modification on the above would do the trick. Now we proceed to formalising this informal idea.

3.2 Formalisation

In the following, we will make extensive use of the notation $\Delta$ as a set of propositional letters to which we have informally referred to in the preceding sub-section. Intuitively, $\Delta$ consists of the atoms that occur in the postconditions of the action or event under consideration and provides the connection to the “relevant” formulas in the theory representing the current belief state.

**Definition 1** Let $\Delta \subseteq \mathbf{P}$ be given. We say that $\varphi \in \mathcal{L}_0$ is $\Delta$-independent iff $\text{atm}(\varphi) \cap \Delta = \emptyset$.

Given a theory $S \subseteq \mathcal{L}_0$, $\text{ind}_\Delta(S)$ denotes the set of all $\Delta$-independent members of $S$.

**Definition 2** Let $\Delta \subseteq \mathbf{P}$ be given and $\Phi$ a set of formulas, a formula $\varphi$ is a $\Delta$-independent consequence of $\Phi$ iff

(i) $\Phi \models \varphi$, and

(ii) $\varphi$ is $\Delta$-independent.

The following definition specifies how independent consequences of a set of clauses are computed.

Let $\Delta \subseteq \mathbf{P}$ be given and $S$ a set of clauses. Define:

- $S_0 = S$; and

- for $i \geq 0$: $S_{i+1} = S_i \cup \{\varphi_1 \lor \varphi_2 : \text{there exists } p \in \Delta \text{ such that } p \lor \varphi_1, \neg p \lor \varphi_2 \in S_i\}$.

Then,

$$\text{res}_{\Delta}(S) \overset{\text{def}}{=} \bigcup_{i=0}^{\infty} S_i.$$  

(1)

Intuitively, $\text{res}_{\Delta}(S)$ represents the closure under resolution of the theory $S$ with respect to the propositional letters from $\Delta$. 

5
Observation 3 Let a theory $S$ be given. For all $\Delta \subseteq P$, $S \models res_\Delta(S)$.

Theorem 4 Given a set of atoms $\Delta$ and a theory $S$, a clause $\varphi$ is an independent consequence of $S$ with respect to $\Delta$ iff $\text{ind}_\Delta(res_\Delta(S)) \models \varphi$.

Example 5 Let $S = \{p \lor q \lor r, \neg p \lor t, \neg r \lor s, \neg s \lor \neg t\}$ and $\Delta = \{p, r\}$.

Then $S_0 = S = \{p \lor q \lor r, \neg p \lor t, \neg r \lor s, \neg s \lor \neg t\}$,

$S_1 = S_0 \cup \{q \lor r \lor t, p \lor q \lor s\}$, and

$S_2 = S_1 \cup \{q \lor s \lor t\} = \{p \lor q \lor r, \neg p \lor t, \neg r \lor s, \neg s \lor \neg t, q \lor r \lor t, p \lor q \lor s, q \lor s \lor t\} = res_\Delta(S)$.

Now we can proceed to eliminating all clauses that contains atoms from $\Delta$ without losing the independent consequences (wrt $\Delta$) in $S$. However, the above computational procedure for $res_\Delta$ may not be very computationally appealing as we potentially may have to enumerate through a number of unwanted clauses. The following more efficient procedure would guarantee a better computation as its computation is linear to the size of $\Delta$.

Let $\Delta = \{p_1, \ldots, p_n\}$ be given and $S$ a theory. Define:

- $D_0 = \emptyset$; and
- for $i \geq 0$, let $\Theta_i = \bigcup_{j=1}^{i} \{c \in S : p_j \in \text{atm}(c)\}$:

$$D_{i+1} = \left\{ \varphi_1 \lor \varphi_2 : p_{i+1} \lor \varphi_1, \neg p_{i+1} \lor \varphi_2 \in (S \setminus \Theta_i) \cup \left( \bigcup_{j=1}^{i} D_j \right) \right\}.$$  

Then, $res'_\Delta(S) \overset{\text{def}}{=} S \cup \bigcup_{i=0}^{n} D_i$.

It is easy to see that the atoms in $\Delta$ will then be dealt with one at a time and the above procedure never re-visits an atom it has previously dealt with. The remaining problem is to show that $res'_\Delta$ will produce exactly what will be produced by $res_\Delta$. The following proposition guarantees that this is the case.

Proposition 6 Given a set of atoms $\Delta$ and a theory $S$,

$$Cn(\text{ind}_\Delta(res_\Delta(S))) = Cn(\text{ind}_\Delta(res'_\Delta(S))).$$

Remark 7 The above results (i.e. Theorem 4 and Proposition 6) are important for the framework presented in this paper. Observe that the attractiveness of STRIPS lies mainly in its computational simplicity: To compute the resulting state after executing an action in an initial state, all the reasoner needs to do is to check for set membership, i.e. removing all atoms in the initial state that are member of the delete-list and then adding the atoms from the add-list in. It is this simplicity that we want to retain in the framework proposed in this paper, see the Remark following Definition 17 for a detailed explanation.

Definition 8 An event description is a pair $\langle \text{pre}, \text{post} \rangle$ of formulas expressed in CNF (or, alternatively, as sets of clauses.)

Definition 9 A dynamic domain $D = \langle \text{Evt}, \text{ED} \rangle$ consists of a set of events $\text{Evt}$ and a function $\text{ED}$ from $\text{Evt}$ to the power set of the set of event descriptions.
For instance, \( ED(\text{toggle\_switch}) = \{\langle \neg on, on \rangle, \langle on, \neg on \rangle \} \).

Before we formally define the function to produce the next state in much the same way as we presented in the previous sub-section, an example may help to discover another problematic case.

**Example 10** Consider a dynamic domain \( D = \langle Evt, ED \rangle \) where \( Evt = \{E\} \) and \( ED(E) = \{\langle q, \neg p \rangle\} \). We start with an initial theory \( \sigma_0 = \{p\} \) and apply the event \( E \) in this state. Consider two possible initial worlds: \( w_1 = \{pq\} \) and \( w_2 = \{\neg p\} \). From \( w_1 \) the resulting state would be \( \{pq\} \) and from \( w_2, \{\neg p\} \). However, starting with the theory \( \sigma_0 \), the resulting theory containing \( p \) will be derived from our framework. This is, of course, unintuitive. This is because we treat the case that \( \sigma \not\models pre(E) \) in the same way as \( \sigma \models \neg pre(E) \).

To deal with the above problem, we introduce the notion of extension of a theory. The key to our solution is to distinguish theories that share the same base theory but are different on some special formulas. For instance, in the above example, we want to consider theories that share the same base theory \( \sigma_0 = \{p\} \), but are different on the formula \( q \). This is very similar to the consideration of the possible models of a theory. However, we don’t have to take into account all possible models as the irrelevant features will be ignored. This is the computational advantage of our approach.

**Definition 11** Let \( S \) be a theory and \( \Phi \) a set of formulas. \( S \) is complete wrt \( \Phi \) iff for each \( \varphi \in \Phi \), either \( S \models \varphi \) or \( S \models \neg \varphi \).

**Definition 12** Let \( S \) be a theory. A set of theories \( \Sigma = \{\sigma_1, \ldots, \sigma_n\} \) is an extension of \( S \) (or \( S \)-extension) iff \( S \) is logically equivalent to \( \bigcup_{i=1}^{n} \sigma_i \).

Now, the intention is to come up with the \( S \)-extension whose members are complete wrt a given set of formulas \( \Phi \). This will be done with the following definition:

**Definition 13** (i) given a formula \( \varphi \), we denote by \( cnf(\varphi) \) the CNF of \( \varphi \) which is a set of clauses whose conjunction is equivalent to \( \varphi \).

(ii) given a theory \( S \) and a formula \( \varphi \),

\[
\text{ext}(S, \varphi) = \begin{cases} 
\{S\} & \text{if } S \models \varphi \text{ or } S \models \neg \varphi, \\
S \cup cnf(\varphi), S \cup cnf(\neg \varphi) & \text{otherwise}
\end{cases}
\]

(iii) given a set of theories \( \Sigma \) and a formula \( \varphi \),

\[
\text{ext}(\Sigma, \varphi) = \bigcup_{\sigma \in \Sigma} \text{ext}(\sigma, \varphi)
\]

(iv) given a set of theories \( \Sigma \),

\[
\text{ext}(\Sigma, \emptyset) = \Sigma
\]

(iv.a) and let \( \Phi = \{\varphi_1, \ldots, \varphi_m\} \) be a set of formulas,

\[
\text{ext}(\Sigma, \Phi) = \text{ext}(\text{ext}(\Sigma, \{\varphi_1, \ldots, \varphi_{m-1}\}), \varphi_m)
\]

\[\text{We informally make use of the notation } pre(E) \text{ to represent the precondition(s) of the event } E. \text{ This notation and its counterpart } post(E) \text{ will be formally introduced in Definition 15 below.}\]

\[\text{Recall that } S \text{ is a set of clauses.}\]
Observe that the first argument of function \textit{ext}(...) can be either a theory or a set of theories, but function \textit{ext}(...) always returns a set of theories. Essentially, item (ii) in the above definition dictates that, given the agent’s belief state \( S \), if \( \varphi \) is one of the preconditions of an event \( E \) then:

- if the agent knows whether \( \varphi \) or \( \neg \varphi \) holds in \( S \), i.e. \( S \models \varphi \) or \( S \models \neg \varphi \), then there is no need to extend the theory \( S : \text{ext}(S, \varphi) \) returns the singleton set \( \{ S \} \);
- otherwise, the agent has to extend the original theory \( S \) to obtain the extension \( \{ S \cup \text{cnf}(\varphi), S \cup \text{cnf}(\neg \varphi) \} \) and subsequently considers two distinct cases: when \( \varphi \) holds and when \( \neg \varphi \) holds. Note that function \textit{ext}(...) returns a set of theories which is the reason for the normalisation of \( \varphi \) (respectively, \( \neg \varphi \)) into CNF.

Item (iii) in the above definition essentially indicates that when the agent considers that all theories from the set \( \Sigma \) are equally possible, then in order to reason about the outcome of an event \( E \) whose precondition is \( \varphi \), she has to extend each of these possibilities and add them together.

Strictly speaking, item (iv) in the above definition is a consequence of (i)-(iii) since the finite set of formulas \( \Phi = \{ \varphi_1, \ldots, \varphi_m \} \) can be equivalently represented as the conjunction \( \varphi_1 \land \ldots \land \varphi_m \). However, these two representations certainly come with different computational consequences. For instance, consider the theory \( S = \{ q, q \lor a \} \) and the set \( \Phi = \{ a, b \} \). Then, \( \text{ext}(S, \Phi) = \{ \{ q, q \lor a, b \}, \{ q, q \lor a, a, b \} \} \) whereas, \( \text{ext}(S, a \land b) = \{ \{ q, q \lor a, a, b \}, \{ q, q \lor a, a \lor b \} \} \). These two different representations also incur different time complexity at step (ii) in the above definition for the number of (CNF) normalisations and satisfiability checkings as well as the complexity of the involved formulas are different. Whereas step (iv) produces a clear advantage with respect to space complexity, it is not so clear whether it enjoys an advantage with respect to the time complexity of the computation carried out at step (ii).

**Proposition 14** Let \( S \) be a theory and \( \Phi = \{ \varphi_1, \ldots, \varphi_m \} \) a finite set of formulas,
1. \( \text{ext}(S, \Phi) \) is a \( S \)-extension,
2. for each theory \( \sigma \in \text{ext}(S, \Phi) \), \( \sigma \) is complete w.r.t. \( \Phi \), and
3. different orders of applying \( \varphi_1, \ldots, \varphi_m \) in general result in syntactically different sets of theories \( \text{ext}(S, \Phi) \). However, they are logically equivalent.

Now, the intention is clear. Start from a theory that may not have a clear stand on every precondition of an event \( E \in \text{Evt} \), we create a set of theories that is equivalent to the original theory. We introduce some syntactical sugar for convenience. Let \( E \in \text{Evt} \) be an event with the description \( ED(E) = \{ \text{pre}_1, \text{post}_1 \}, \ldots, \{ \text{pre}_n, \text{post}_n \} \), we denote by \( \text{pre}(E) \) the set \( \{ \text{pre}_1, \ldots, \text{pre}_n \} \) and by \( \text{post}(E) \) the set \( \{ \text{post}_1, \ldots, \text{post}_n \} \). In the following definition, we will abuse the notation by overloading the functions \textit{pre} and \textit{post}:

**Definition 15** Let \( E \in \text{Evt} \) be an event and \( S \) a theory, we define:

1. \( \text{pre}(E, S) \overset{\text{def}}{=} \{ c : \text{there exists } \varphi \in \text{pre}(E) \text{ such that } S \models \varphi \text{ and } c \in \varphi \} \)
2. \( \text{post}(E, S) \overset{\text{def}}{=} \{ c : \text{there exists } \langle \varphi, \psi \rangle \in ED(E) \text{ such that } S \models \varphi \text{ and } c \in \psi \} \).

Recall from Definition 8 that \( \varphi \) and \( \psi \) in the above definition are CNF-formulas which are expressed as sets of clauses. Essentially, \( \text{post}(E, S) \) returns the set of all possible effects (represented as a set of clauses) the event \( E \) could cause to the world relative to the agent’s belief state \( S \). On the other hand, \( \text{pre}(E, S) \) renders the set of all conditions accountable for those changes.
When $pre(E)$ or $post(E)$ consists of a single element, we will omit the set notation to simplify the presentation. For instance, the action of picking a box (in the blocks world domain) can be described as follows: $E = \text{pick}$, $pre(\text{pick}) = \text{free(hand)} \land \text{clear(block)}$ and $post(\text{pick}) = \text{inhand(block)}$.

**Remark 16** As discussed by Baral [2], there are two types of conditions for an action/event, namely the “executability condition of an action” and the “preconditions of effects (of an action).” For instance, consider the action $\text{shoot}$: It is impossible to perform this action if the agent does not have a gun. Thus, having a gun is an executability condition for the action $\text{shoot}$. On the other hand, whether or not the effect of the victim getting killed will take place when the action $\text{shoot}$ is performed depends on the preconditions for this effect such as the gun being $\text{loaded}$. While it’s easy to see that the above formulation of actions and events renders the second type of conditions for actions, i.e. the “preconditions of effects (of an action),” it’s less clear whether it could also address the “executability condition of actions.” It’s worth noting that our framework is change-oriented in the sense that the agent’s main focus of attention is in the changes (or the effects) an action/event would make to the world. In this sense, the “executability conditions” of actions can be encoded in the preconditions of the actions. For instance, the following event description for the action $\text{toggle\_switch}$ incorporates the executability condition $\text{free\_hand}$:

$$ED(\text{toggle\_switch}) = \{(\text{free\_hand} \land \neg \text{on}, \text{on}), (\text{free\_hand} \land \text{on}, \neg \text{on})\}.$$  

On the other hand, one of the main inspirations behind Baral’s [2] introduction of the “executability conditions” of actions is to deal with the qualification problem. This is in particular the case when the agent has to reason about the occurrences of actions and events, e.g. when solving the explanation problem. For instance, when solving the mystery of Fred’s murder, detective Mike easily rules out the possibility of Fred getting killed because Jim shot him for Jim never had a gun. Thus, the event of Jim shooting Fred never occurred let alone the question of whether any effect of this action was materialised. However, the framework introduced in this paper can not be used to reason about occurrences of actions and events for lack of a representation of time. Thus, in this sense, our framework does not address the executability conditions of actions.\(^5\)

**Definition 17** Given an event $E \in \text{Evt}$ and a theory $S$ which is complete wrt $pre(E)$, the application of $E$ to $S$ is defined as follows:

$$result(S, E) = \{c \in res_\Delta(S) : \text{atm}(c) \land \Delta = \emptyset\} \cup post(E, S)$$ \hspace{1cm} (2)

where $\Delta = \text{atm}(post(E, S))$.

**Remark 18** The above definition shows the basic mechanism used by our framework to compute the resulting state $result(S, E)$ after the occurrence of an action/event $E$ in an initial state $S$. It also reflects the simplicity of the STRIPS update operator: Given a theory $S$ and an event $E$, once $res_\Delta(S)$ has been computed, the agent only needs to remove all clauses from $res_\Delta(S)$ that contain an atom from $\Delta = \text{atm}(post(E, S))$ before adding the set of clauses $post(E, S)$ in. Now, the problem lies in the computation of $res_\Delta(S)$: from equation 1, we may have to repeat the computation of the sets $S_i$’s an unknown number of times before a fixed point is reached to obtain $res_\Delta(S)$. Here, Theorem 4 and Proposition 6 show their significance: computing the set $\{c \in res_\Delta(S) : \text{atm}(c) \land \Delta = \emptyset\}$

\(^5\)We would like to thank an anonymous referee for pointing out this subtle point.
in equation 2 can be reduced to computing the set \( \text{ind}_\Delta(\text{res}_\Delta'(S)) \). The computational complexity of computing \( \text{ind}_\Delta(\text{res}_\Delta'(S)) \) is linear to the size of \( \Delta \), or linear to the size of \( \text{atm}(\text{post}(E, S)) \). Thus, the more fluents the event \( E \) changes in the world, the more computation is involved which is essentially inevitable in reasoning about dynamic domains. Note that the function \( \text{result}(., .) \) is defined as in Definition 17 because it allows Theorem 22, one of the major results of this paper, to be obtained straightforwardly.

Definition 17 is extended to an arbitrary theory \( S \) in a straightforward way:

Given an event \( E \in \text{Evt} \) and a theory \( S \), the application of \( E \) to \( S \) is defined as follows:

\[
\text{result}(S, E) = \bigcup_{\sigma \in \text{ext}(S, \text{pre}(E))} \text{result}(\sigma, E)
\]

One distinguished feature of our approach is its ability to handle actions or events with disjunctive effects, i.e. non-deterministic actions. This is a clear advantage compared to approaches in belief update research (e.g. Winslett’s [23] possible model approach, or PMA) and reasoning about action research (e.g. McCain and Turner’s [13] framework) that are based on the principle of minimal change. We consider an example to illustrate this advantage.

Example 19 6 The only action in this domain is throwing a chip onto a chess board with black and white squares (i.e., throw) whose sole precondition is that the chip is in hand (of the thrower). The event description of throw is as follows: \( \text{pre}(\text{throw}) = \text{chip\_in\_hand} \) and \( \text{post}(\text{throw}) = (\text{chip\_in\_white}\lor\text{chip\_in\_black})\land\neg\text{chip\_in\_hand} \). The fluents \text{chip\_in\_white} (resp. \text{chip\_in\_black}) indicate that, when the chip is on the chess board, it is in a white (resp. black) square. Consider also the following initial state \( \sigma_0 = \{\text{chip\_in\_hand}, \neg\text{chip\_in\_white}, \neg\text{chip\_in\_black}\} \). Using Winslett’s PMA or McCain and Turner’s minimal change approach we have the result:

\[
(\neg\text{chip\_in\_hand}\land\text{chip\_in\_white}\land\neg\text{chip\_in\_black})\lor(\neg\text{chip\_in\_hand}\land\neg\text{chip\_in\_white}\land\text{chip\_in\_black}),
\]

which says that the chip must be in exactly one of the squares, either black or white, but not both. This is clearly unintuitive.

On the other hand, \( \text{result}(\sigma_0, \text{throw}) = \{\} \cup \text{post}(\text{throw}, \sigma_0) \), i.e. firstly, the fluents \text{chip\_in\_hand}, \text{chip\_in\_white} and \text{chip\_in\_black} are removed from \( \sigma_0 \) since they all occur in the postcondition of \( \text{throw} \) and then the CNF of \( \text{post}(\text{throw}, \sigma_0) \) is added to form the next state. In other words, \( \text{result}(\sigma_0, \text{throw}) = \{\text{chip\_in\_white} \lor \text{chip\_in\_black}, \neg\text{chip\_in\_hand}\} \). In this approach, the chip is no longer in hand and can now be in a white square or in a black square or occupy both black and white squares.

One issue with our approach is due to the syntax-sensitivity with respect to the postcondition of the actions. For instance, \( \neg p \lor p \) and \( \text{true} \) are logically equivalent but the resulting states as computed by the function \( \text{result} \) defined above may be different on these two postconditions. We note that \( \text{atm}(\neg p \lor p) = \{p\} \) while \( \text{atm}(\text{true}) = \emptyset \). The common approach to avoid syntax-sensitivity is to eliminate the redundant atoms from a formula by reducing it to an equivalent one without that redundant atom. This process is known to be coNP-complete. However, such a redundancy may be intended as part of the action description. In other words, a process of eliminating the redundant atoms may not be desirable and it could be better that the framework would be sensitive to different syntaxes.

---

6This example was suggested by Ray Reiter and discussed in [8].
Example 20 Given the following event description of flipping a coin: \( \text{pre(}\text{flip}\text{)} = \text{true} \) and \( \text{post(}\text{flip}\text{)} = \text{head} \lor \neg \text{head} \). This is of course very different from having \( \text{post(}\text{flip}\text{)} = \text{true} \), especially if the initial state is \( \sigma_0 = \{\text{head}\} \). This is because the set \( \Delta \) of atoms used to determine the connection between the postcondition of the event \( \text{flip} \) and the theory representing the next state contains the atom \( \text{head} \) in the former and is empty in the latter. As a consequence, \( \text{result}(\sigma_0, \text{flip}) = \{\text{head} \lor \neg \text{head}\} \) when \( \text{post(}\text{flip}\text{)} = \text{head} \lor \neg \text{head} \) and \( \text{result}(\sigma_0, \text{flip}) = \{\text{head}\} \) when \( \text{post(}\text{flip}\text{)} = \text{true} \). Certainly, the former would be preferred over the latter.

Therefore, the syntax-sensitivity of our approach should be considered as a feature rather than a problem.

3.3 Technical results

One might worry that there are clauses that should be removed from \( \sigma \) but would not be removed by the above definition (i.e. removing too few) or there are ones that should not be removed but would be removed by our definition (i.e. removing too many). The following theorem will assure that that will not be the case.

We first introduce the computation which is based on some complete description of the world. This is well-known through state-based formalisms of reasoning about action, e.g. Lin and Shoham’s [12] epistemological completeness, Gelfond and Lifschitz’s [4, 5] transition systems.

Let us ignore the ramification problem for now by assuming that no domain constraints are present in the domain of interest. The idea of this solution to the frame problem is similar to those in which the number of changed fluents is minimised. However, as explained above, all fluents that occur in the postcondition of an event will be subject to change. This is similar to the so-called non-inertial fluents introduced by Gelfond and Lifschitz [5]. We believe that this provides a better treatment as fluents are not required to be strictly non-inertial throughout but they can be non-inertial relative to some events while remaining inertial relative to the rest.

Before considering the resulting states by performing an event \( E \) in a state \( \sigma \), we consider the issue of updating a state \( \sigma \) with a set of literals, or a partial interpretation, \( I \).

Definition 21 Let \( \omega \) be an interpretation and \( \Delta \) a set of atoms. Given a partial interpretation \( I \) over \( \Delta \), we define

\[
\text{updated}(\omega, I)(p) = \begin{cases} 
I(p) & \text{if } p \in \Delta, \\
\omega(p) & \text{otherwise} 
\end{cases}
\]

In order to deal with the (direct) effects of an event \( E \), we will enumerate all the possible partial interpretations that can be the effects of \( E \) in the interpretation \( \omega \). Then the set of possible resulting interpretations will be the updated interpretations from \( \omega \) relative to those partial interpretations.

Given an interpretation \( \omega \) and an event \( E \in \text{Evt} \), we denote by \( TA(E, \omega) \) the set of all possible partial interpretations over the set of atoms \( \text{atm}(\text{post}(E, \omega)) \) that satisfy \( \text{post}(E, \omega) \), i.e. if \( \tau \in TA(E, \omega) \) then \( \tau \models \text{post}(E, \omega) \).

For instance, given \( \text{post}(E, \omega) = (p \lor q) \land (p \lor r) \land (q \lor r) \), then

\[
TA(E, \omega) = \{ \{p, q, r\}, \{p, q, \neg r\}, \{p, \neg q, r\}, \{\neg p, q, r\} \}.
\]

We can then introduce \( \text{nextState} \) of a given interpretation \( \omega \) for an event \( E \in \text{Evt} \):

\[
\text{nextState}(\omega, E) = \{ \text{updated}(\omega, \tau) : \tau \in TA(E, \omega) \}.
\]

Theorem 22 Let \( D = (\text{Evt}, ED) \) be a dynamic domain. Given a theory \( S \) and an event \( E \in \text{Evt} \),

\[
\text{Mod} (\text{result}(S, E)) = \bigcup_{\omega \in \text{Mod}(S)} \text{nextState} (\omega, E).
\]
Example 23  We illustrate our approach with the classic example Yale Shooting Problem (YSP) [7]:
Consider a dynamic domain $D = (Evt, ED)$ where $Evt = \{\text{load}, \text{wait}, \text{shoot}\}$ and

$$
\begin{align*}
\text{pre}(\text{load}) &= \text{true} & \text{post}(\text{load}) &= \text{loaded} \\
\text{pre}(\text{wait}) &= \text{true} & \text{post}(\text{wait}) &= \text{true} \\
\text{pre}(\text{shoot}) &= \text{loaded} & \text{post}(\text{shoot}) &= \neg \text{alive}
\end{align*}
$$

We start with an initial theory $\sigma_0 = \{\}$ representing the agent’s beliefs about the current state of
the world and assume that our agent wishes to reason about the state of the world after applying the
sequence of actions $\text{load}; \text{wait}; \text{shoot}$ in this state.

Note that $\sigma_0 \models \text{pre}(\text{load})$ and thus, $\text{post}(\text{load}, \sigma_0) = \text{loaded}$. As a consequence, $\text{result}(\sigma_0, \text{load}) = \{\} \cup \{\text{loaded}\}$.
Therefore, after performing the action $\text{load}$, the computed theory for the resulting state will be $\sigma_1 = \text{result}(\sigma_0, \text{load}) = \{\text{loaded}\}$. On the other hand, within the state-based approach, the theory $\sigma_0$ corresponds to the following four interpretations (provided that there are only

- $\omega_1 = \{\text{loaded}, \text{alive}\}$, (and thus, $\text{nextState}(\omega_1, \text{load}) = \{\text{loaded}, \text{alive}\}$);
- $\omega_2 = \{\text{loaded}, \neg \text{alive}\}$, (and thus, $\text{nextState}(\omega_2, \text{load}) = \{\text{loaded}, \neg \text{alive}\}$);
- $\omega_3 = \{\neg \text{loaded}, \text{alive}\}$, (and thus, $\text{nextState}(\omega_3, \text{load}) = \{\text{loaded}, \text{alive}\}$);
- $\omega_4 = \{\neg \text{loaded}, \neg \text{alive}\}$, (and thus, $\text{nextState}(\omega_4, \text{load}) = \{\text{loaded}, \neg \text{alive}\}$).

In other words, the set of interpretations for $\bigcup_{\omega \in \text{Mod}(\sigma_0)} \text{nextState}(\omega, \text{load}) = \{\omega_1, \omega_2\}$ which corresponds to the theory $\sigma_1$.

Now, observe that $\sigma_1 \models \text{pre}(\text{wait})$ and thus, $\text{post}(\text{wait}, \sigma_1) = \text{true}$. Therefore, after performing the action $\text{wait}$, the computed theory for the resulting state will be $\sigma_2 = \text{result}(\sigma_1, \text{wait}) = \{\text{loaded}\}$ as no fluent literal needs to be deleted from or added to $\sigma_1$. The state-based approach would also maintain the two interpretations $\omega_1$ and $\omega_2$ as the possible next states after performing the action $\text{wait}$ in $\sigma_0$.

Since $\sigma_2 \models \text{pre}(\text{shoot})$, we have $\text{post}(\text{shoot}, \sigma_2) = \neg \text{alive}$. Thus, after performing the action $\text{shoot}$, the computed theory for the resulting state will be $\sigma_3 = \text{result}(\sigma_2, \text{shoot}) = \{\text{loaded}, \neg \text{alive}\}$.

The above example shows the simplicity of this approach to the problem of reasoning about
action. This also looks a very natural way to represent and reason about the effects of actions.

Consider a small modification to the postcondition of the action $\text{shoot}$. Assume that $\text{post}(\text{shoot}) = \{\neg \text{alive}, \text{loaded} \lor \neg \text{loaded}\}$. The idea of this description is of course that the gun can possibly get
unloaded after the action $\text{shoot}$ in case, say, there is only one bullet left in the cartridge. In other
words, the agent may not know whether the gun is still loaded after performing the action $\text{shoot}$.
It’s easy to see that the computed theory for the agent’s belief state after performing the action $\text{shoot}$
from the state represented by $\sigma_2$ gives us:

- As $\sigma_2 \models \text{pre}(\text{shoot})$, $\text{post}(\text{shoot}, \sigma_2) = \{\neg \text{alive}, \text{loaded} \lor \neg \text{loaded}\}$;

- Thus, $\text{atm}(\text{post}(\text{shoot}, \sigma_2)) = \{\text{alive}, \text{loaded}\}$. Consequently, after performing the action
$\text{shoot}$, the computed theory for the resulting state will be $\sigma_3 = \{\text{loaded} \lor \neg \text{loaded}, \neg \text{alive}\}$, which is a desirable outcome.

We continue this example with another modification on the action description of the action $\text{shoot}$
to illustrate how our framework deals with actions/events with conditional effects: Assume that, in

\[\text{Note that the formula } \text{loaded} \text{ is equivalent to the clause } \{\text{loaded}\}.\]

\[\text{As the atom } \text{loaded} \in \text{ of the set } \text{atm}(\text{post}(\text{shoot}, \sigma_2)), \text{ the clause } \text{loaded} \text{ is removed from } \sigma_2 \text{ before the}
\text{clauses } \text{loaded} \lor \neg \text{loaded} \text{ and } \neg \text{alive} \text{ being added to form } \sigma_3.\]
addition to killing the victim when the gun is loaded, the action *shoot* also causes the gun to be unloaded regardless of whether or not the gun is loaded before the action *shoot* is performed. The event description for *shoot* is as follows:

\[ ED(\text{shoot}) = \{\text{loaded, } \neg\text{-alive}, \text{true, } \neg\text{-loaded}\} \]

Consider the case of the action *shoot* being performed in state \( \sigma_2 \): Observe that \( \sigma_2 \models \text{loaded} \) and \( \sigma_2 \models \text{true} \). Thus, \( \text{post}(\text{shoot}, \sigma_2) = \{\neg\text{-alive}, \neg\text{-loaded}\} \). As a consequence, after performing the action *shoot*, the computed theory for the resulting state will be \( \sigma'_2 = \{\neg\text{-alive}, \neg\text{-loaded}\} \).

## 4 A framework for reasoning about action

### 4.1 The relationship with action language \( \mathcal{A} \)

Why is the above notion of *nextState* of any interest to us at all? We will show that, in the special case of non-redundant postconditions of deterministic events, there is an equivalent translation from a dynamic domain to an action description in the action language \( \mathcal{A} \) ([5]), and vice versa.

Recall that in the action language \( \mathcal{A} \) ([5]),

1. An *action signature* consists of three nonempty sets: a set \( \mathcal{V} \) of *value names*, a set \( \mathcal{F} \) of *fluent names*, and a set \( \mathcal{A} \) of *action names*. Intuitively, any “fluent” (represented by a symbol from \( \mathcal{F} \)) has a specific “value” (represented by a symbol from \( \mathcal{V} \)) in any state of the world.

2. an *\( \mathcal{A} \)-proposition* is an expression of the form

\[ A \text{ causes } L \text{ if } F \]

where \( A \) is an action name, \( L \) a literal and \( F \) a conjunction of literals (possibly empty),

3. an *\( \mathcal{A} \)-action description* is a set of propositions.

The semantics of the action language \( \mathcal{A} \) is given as follows:

**Definition 24 ([5])** Let \( D \) be an action description in \( \mathcal{A} \). The transition system \( \langle S, V, R \rangle \) described by \( D \) is defined as follows:

1. \( S \) is the set of all interpretations;
2. for \( \omega \in S \) and a proposition \( P \), \( V(P, \omega) = \omega(P) \);
3. \( R \) is the set of all triples \( \langle \omega, A, \omega' \rangle \) such that:

\[ \text{Res}(A, \omega) \subseteq \omega' \subseteq \text{Res}(A, \omega) \cup \omega \]

where \( \text{Res}(A, \omega) \) stands for the set of the heads \( L \) of all propositions \( A \text{ causes } L \text{ if } F \) in \( D \) such that \( \omega \) satisfies \( F \).
4.1.1 Translation from dynamic domains to \( A \)-action descriptions

For each event \( E \in Evt \): for each event description \( \langle \text{pre}, \text{post} \rangle \in ED(E) \), as \( E \) is deterministic \( \text{post} \) can be written as a conjunction of literals: \( \text{post} = l_1 \land \ldots \land l_k \) and we just simply add the following \( A \)-propositions to the \( A \)-action description: \( E \text{ causes } l_i \) if \( \text{pre} \) (for \( i = 1, \ldots, k \)).

The following proposition is immediate from Theorem 22 (and is proved in the Appendix):

**Proposition 25** Let a deterministic dynamic domain \( D = (Evt, ED) \) be given. If \( D \) is an \( A \)-action description obtained from \( D \) by the above translation and \( \langle S, V, R \rangle \) is the transition system described by \( D \) then, for any theory \( T \) and any event \( E \in Evt, \langle \omega, E, \omega' \rangle \in R \) if and only if \( \omega' \in Mod(\text{result}(T, E)) \) provided \( \omega \) is a model of \( T \).

4.1.2 Translation from \( A \)-action descriptions to dynamic domains

For each action \( A \):

(i) we add \( A \) to the set of events \( Evt \) and assign the empty set to \( ED(A) \),

(ii) for each \( A \)-propositions \( A \text{ causes } L \text{ if } F \), we add the following event description to \( ED(A) \):

\[ \langle \text{cnf}(F), L \rangle \], where \( \text{cnf}(F) \) is the conjunctive normal form of the formula \( F \).

**Proposition 26** Let \( D \) be a (propositional) \( A \)-action description over the action signature \( \{\text{false, true}\}, F, A \) and the transition system \( \langle S, V, R \rangle \) be described by \( D \) (see [5] for the definitions of these concepts).

If \( D = (Evt, ED) \) is the dynamic domain obtained by the above translation then for any \( \omega \in S \) and \( A \in A \),

\[ \text{Mod}(\text{result}(\omega, A)) = \{ \omega' \in S \mid \langle \omega, A, \omega' \rangle \in R \} \].

4.2 The Ramification problem

So far we have not taken into consideration the indirect effects as well as the issues related to domain constraints. Lin ([11]) and McCain and Turner ([13]) have pointed out that domain constraints represented as formulas may not be sufficient to express the relationships between fluents of the domain. The missing information as largely recognised by researchers is the direction in which the fluents influence each others. For instance, killing the turkey causes it to stop walking but it is not the case that making a turkey walk causes it to be alive. We represent the set of such causal relationships as a special kind of event. One can actually call them natural events after all. In this way, we can uniformly treat ramifications as events that occur right after the occurrence of the main event but possibly in some unknown order.

We will assume that the set of events \( Evt \) is extended so that it includes a countable set of ramification rules \( Ram \). As every indirect effect can be associated with a distinct ramification rule, for each \( r \in Ram, ED(r) \) is a singleton. Therefore, we will abuse the notation sometimes by writing \( \text{pre}(r) \) and \( \text{post}(r) \) instead.

**Definition 27** A theory \( \sigma \) is stable iff there does not exists any ramification rule \( r \in Ram \) such that \( \sigma \models \text{pre}(r) \) and \( \sigma \not\models \text{post}(r) \).

**Definition 28** A ramification rule \( r \in Ram \) is applicable in a theory \( \sigma \) iff \( \sigma \models \text{pre}(r) \).

A sequence \( sr = \langle r_1, \ldots, r_n \rangle \) of ramification rules is applicable to a theory \( \sigma \) iff there exists a sequence of theories \( \langle \sigma_0, \ldots, \sigma_n \rangle \) such that the following conditions hold:

1. \( \sigma = \sigma_0 \),
2. \( r_i \) (for \( i = 1, \ldots, n \)) is applicable in \( \sigma_{i-1} \).
3. \( \sigma_i \in \text{result}(\sigma_{i-1}, r_i) \).

A sequence of ramification rules \( sr = \langle r_1, \ldots, r_n \rangle \) which is applicable to a theory \( \sigma \) produces a possible next state iff all possible \( n \)-th states are stable.

Alternatively, instead of listing all the possible sequence of ramification rules, the agent may care only about what fluents would hold after the occurrence of some event \( e \). In that case, many sequences of ramification rules may bring about the same eventual effects. The agent could effectively take into account the possible resulting theories. Such theories will be called reachable theories.

**Definition 29** Given a theory \( \tau \), a theory \( \sigma \) is reachable from \( \tau \) iff there exists a sequence of ramification rules which is applicable in \( \tau \) such that \( \sigma \) is logically derivable from the resulting theory. Formally,

\[
\begin{align*}
\sigma_0 &= \tau \\
\sigma_{i+1} &= \{ \varsigma : \text{there exist } r \in \text{Ram} \text{ and } \omega \in \sigma_i \text{ such that } \varsigma \in \text{result}(\omega, r) \}. 
\end{align*}
\]

Observe that the translations between dynamic domains and \( \mathcal{A} \)-action descriptions presented in section 4.1 can be extended in a natural way for dynamic domains that include a set of ramification rules \( \text{Ram} \) and \( \mathcal{B} \)-action descriptions (see [5] for the definitions of the action descriptions in the action language \( \mathcal{B} \), or \( \mathcal{B} \)-action descriptions): For each \( r \in \text{Ram} \), if \( \text{post}(r) = l_1 \land \ldots \land l_k \) then the ramification rule \( r \) can be translated to the action language \( \mathcal{B} \) as a set of static laws: \( \{ l_i \text{ if } \text{pre}(r) \} \}_{i=1}^{k} \).

Conversely, each static law \( L \text{ if } F \) in a \( \mathcal{B} \)-action description is translated to a dynamic domain by adding a ramification rule \( \rho \) to \( \text{Evt} \) such that \( ED(\rho) = (F, L) \). Propositions 25 and 26 can then be straightforwardly established for these translations.

On the other hand, there is no direct correspondence between dynamic domains described in the present paper and the action language \( \mathcal{C} \) (see [5] for details about the action language \( \mathcal{C} \)). In the action language \( \mathcal{C} \), combinations of elementary action names (from \( \mathcal{A} \)) using propositional connectives are allowed.

## 5 A framework for belief update

The problem of belief update can essentially be considered as a simplified version of the problem of reasoning about action in which actions and time are generally not explicitly represented. The effects of the (implicit) actions are encoded in the updating formulas while the agent’s knowledge about the current state of the world, i.e. the updated theory, is represented as a finite set of formulas and usually referred to in the literature as a belief base. Provided our formalisation for reasoning about action, it’s easy to treat our framework as a belief update mechanism: the updating formula is the postcondition of a (uniquely-determined) action whose precondition is true. Formally, for each updating formula \( \mu \), we introduce a unique action \( E_\mu \) such that \( ED(E_\mu) = \{ \text{true}, \mu \} \). The resulting update operator is denoted by \( \circ_{sb} \).

Subsequently, we would like to investigate the following fundamental question: How suitable is the proposed framework for the purpose of belief update? The standard way employed by most researchers working in the field is to compare an update operator with the Katsuno-Mendelzon (KM) postulates (see [9]): Let \( \psi \circ \mu \) denote the result of updating the knowledge base \( \psi \) with the sentence \( \mu \). KM postulates for update are:

---

9 Note that, similar to the action language \( \mathcal{A} \), action descriptions in the action language \( \mathcal{B} \) only deal with deterministic actions.
(U1) \( \psi \circ \mu \) implies \( \mu \).

(U2) If \( \psi \) implies \( \mu \) then \( \psi \circ \mu \) is equivalent to \( \psi \).

(U3) If both \( \psi \) and \( \mu \) are satisfiable then \( \psi \circ \mu \) is also satisfiable.

(U4) If \( \psi_1 \iff \psi_2 \) and \( \mu_1 \iff \mu_2 \) then \( \psi_1 \circ \mu_1 \iff \psi_2 \circ \mu_2 \).

(U5) \( (\psi \circ \mu) \land \phi \) implies \( \psi \circ (\mu \land \phi) \).

(U6) If \( \psi \circ \mu_1 \) implies \( \mu_2 \) and \( \psi \circ \mu_2 \) implies \( \mu_1 \) then \( \psi \circ \mu_1 \iff \psi \circ \mu_2 \).

(U7) If \( \psi \) is complete then \( (\psi \circ \mu_1) \land (\psi \circ \mu_2) \) implies \( \psi \circ (\mu_1 \lor \mu_2) \).

(U8) \( (\psi_1 \lor \psi_2) \circ \mu \iff (\psi_1 \circ \mu) \lor (\psi_2 \circ \mu) \).

As expected, the fact that the update operator \( \circ_{sb} \) is syntax sensitive destroys several postulates introduced by Katsuno and Mendelzon:

**Theorem 30** The proposed syntax-based update operator \( \circ_{sb} \) satisfies Katsuno-Mendelzon’s update postulates (U1), (U3), (U5), and (U7)–(U8).

That \( \circ_{sb} \) doesn’t satisfy (U2) was illustrated in Example 20, i.e. when the updated theory \( \psi \equiv head \), and the updating formula is \( \mu \equiv \neg head \lor head \). Note that \( \psi \) implies \( \mu \), but \( \psi \circ_{sb} \mu \equiv \neg head \lor head \) is not equivalent to \( \psi \). This is in stark contrast to the outcomes produced by other belief update operators which are introduced to deal with updates with disjunctive information. For instance, both the MCE and the MCD update operators proposed by Zhang and Foo [26, 27] satisfy (U2). That is, \( \psi \circ_{MCE} \mu \equiv \psi \circ_{MCD} \mu \equiv head \). This outcome of the action of tossing a coin is clearly unintuitive.

The sceptical reader might question the well-definedness of the updating formula \( \neg head \lor head \) and suggest that this is the reason behind MCE and MCD update operators to fail to produce an intuitive outcome in the above example. However, a modified version of example 19 should persuasively illustrate this subtle point. Let \( \psi \equiv chip\_in\_black \) and \( \mu \equiv chip\_in\_black \lor chip\_in\_white \). Observe that \( \psi \) implies \( \mu \), but the update operator \( \circ \) that satisfies (U2) must produce: \( \psi \circ \mu \equiv \psi \equiv chip\_in\_black \) which is a counter-intuitive outcome for this example. In other words, this example illustrates that the KM postulate (U2) is not a suitable axiom for updates with disjunctive information.

Postulate (U4) states that an update operator should not be sensitive to the syntax of the involved formulas. This postulate is violated by our update operator \( \circ_{sb} \) for it is sensitive to the syntax of the updating formula \( \mu \). As discussed in the example 20, we consider this syntax sensitivity as a desirable feature of our update mechanism rather than a shortcoming of the framework. However, \( \circ_{sb} \) satisfies the following weaker version of (U4):

**(U4′)** If \( \psi_1 \iff \psi_2 \) then \( \psi_1 \circ \mu \iff \psi_2 \circ \mu \).

The following example shows that \( \circ_{sb} \) doesn’t satisfy (U6):

**Example 31** Let \( \psi \equiv p \land q \) and \( \mu_1 \equiv \neg p \lor p \) and \( \mu_2 \equiv \neg q \lor q \), then \( \psi \circ_{sb} \mu_1 \equiv q \) and \( \psi \circ_{sb} \mu_2 \equiv p \). The problem here is again due to the syntax sensitivity of the update operator \( \circ_{sb} \) regarding the syntax of the updating formula \( \mu \), rather than some peculiar behaviour of \( \circ_{sb} \) when performing the update.

\( \mu \) can be considered as the effects caused by the action of flicking a chip on the board and making it land on an arbitrary location on the board (which can be either in a black square, or in a white square, or touching both a black square and a white square).
5.1 Related work

In the context of belief update and revision, Marianne Winslett ([23, 24]) introduces a framework for theory update: the PMA (Possible Models Approach) – a classical minimal change approach for updates. Winslett’s framework is also sensitive to the syntax of the formulas to be updated to the current theory. These are the consequences of the so-called observations by Winslett with the pre-requisites that the updating formulas be incorporated to the observations. The formulas that are brought forward from the old theory to the updated theory are those that don’t contain any ground atom (or are not unifiable to a predicate) that is present in the observation.

In the PMA, the knowledge base update is achieved by updating every possible model of the world satisfying the updated theory \( \psi \) with the observations incorporating the updating formula \( \mu \). Furthermore, such state updates are constructed based on minimal changes on the models. However, as pointed out in our paper, these are the sources of the two severe problems: intractable space complexity and counter-intuitive results for updates with disjunctive information. On the other hand, mechanisms which are designed to deal with updates with disjunctive information such as the MCE and the MCD update operators ([26, 27]) also produce counter-intuitive update outcomes in a number of situations as discussed in the previous section.

Another issue with Winslett’s formalism is the generation of new atoms. As the update operation is carried out, the so-called history atoms are generated to guarantee that the formulas to be got rid off will not be thrown away altogether but will still be kept in the form of the same formula but in regard to the historic atoms whose truth values are potentially changed during the course of the update operation. This way, Winslett guarantees that the formulas that provide information about the irrelevant atoms (relative to the on-going update) will not be missed out. This is what our framework achieves by carrying out the resolution step in the update operation presented above.

In the worst case, Winslett’s framework increases the size of the language used to represent the knowledge base (or, data base) in exponential space as each atom may require a new history atom to be added to the language for each update. Moreover, as all formulas are kept during the update, either in the original form or as a historic formula, the size of the theory is strictly monotonically increased with each update. Thus, in terms of space complexity, our formalism is much more economical than Winslett’s framework. This is because we do not change the vocabulary of the language which is used to represent the knowledge base. Furthermore, comparing to related model-based approaches which do not increase the language of representation such as McCain and Turner’s [13] or Zhang and Foo’s [26, 27], our approach still enjoys the feature of being able to avoid generating all possible world models. Instead, we only account for a proposition when it is necessary to determine whether our current theory about the world entails that proposition, i.e. in order to determine the applicability of the event/action. Certainly, in the worst case when the preconditions of a (very unusual) action require that every proposition be checked then the notion of theory extension presented in the present paper collapses to the set of all possible world models and thus suffers the same complexity as existing model-based approaches to reasoning about action and belief update.

6 Concluding Remarks

In this paper we have described an approach to several problems of reasoning about action, namely the frame and the ramification problems. The approach is attractive in the way new states are computed. By treating ramifications as natural events, a uniform treatment for general actions and indirect effects can be achieved. The approach provides a machinery for general problem solvers that are required to function in dynamic domains.
The approach developed here is in some sense a formalised extension of the STRIPS system. It’s similar to STRIPS in the way new information is updated to a current theory. Thus, it retains the conceptual simplicity of STRIPS. On the other hand, as arbitrary propositional theories are accepted and represented, it provides a significant improvement in the expressivity of STRIPS. In particular, our proposed framework for reasoning about dynamic domain is able to handle non-deterministic actions/events and produce intuitive inferences in such cases while many other approaches to reasoning about action and belief update can not account for these actions/events easily.

We also discussed on how our proposed framework would be used as an update operator. We show that the resulting update mechanism enjoys several advantages in comparison to the various belief update operators proposed in the literature.

References


Appendix

We introduce some notations which will be used in the proofs below:

Let \( S \) be a set of clauses and \( c \) a clause, a \textit{resolution-based derivation} \( D \) of \( c \) from \( S \) is a sequence \( \langle c_1, \ldots, c_n \rangle \) such that:

1. \( c = c_n \), and
2. for each \( k \geq 1 \), either \( c_k \in S \) or there exist \( i, j < k \) such that \( c_k \) is the result of resolving the two clauses \( c_i \) and \( c_j \) on some atom \( p \), denoted by \( c_k = \text{resolve}(c_i, c_j, p) \). \( c_k \) is called the \textit{resolvent} of \( c_i \) and \( c_j \), and \( c_i \) and \( c_j \) are called the \textit{resolvers} of \( c_k \).

A (resolution-based) derivation \( D = \langle c_1, \ldots, c_n \rangle \) is \textit{minimal} if it is non-redundant in the sense that for all \( i < n \), \( c_i \) must be involved in the derivation of \( c_k \) for some \( k > i \).

The \textit{length} of a resolution-based derivation from \( S \) is the number of derivatives that are not given in the initial set \( S \). If a resolution-based derivation is represented as a tree then its length is the number of internal nodes on that tree.

A \textit{resolution proof} of \( c \) from \( S \), denoted by \( S \vdash_{\text{res}} c \), is a resolution-based derivation of \textit{false} from \( S \cup \{\neg c\} \), where \( \{\neg c\} \) denotes the set of clauses obtained from \( \neg c \), i.e. let \( \{\neg c_1, \ldots, \neg c_m\} \) be a set of clauses such that \( \neg c \equiv \wedge_{i=1}^{m} \neg c_i \), then \( \{\neg c\} = \{\neg c_1, \ldots, \neg c_m\} \). Henceforth we will only consider minimal resolution proofs.

The completeness of resolution states that for all \( S \subseteq L \) and \( F \in L \), \( S \models F \) iff \( S \vdash_{\text{res}} F \) (see e.g. Robinson [20]).

Before proceeding to prove Theorem 4, we introduce the following lemma which will assist in its proof.

\textbf{Lemma 32} Let \( S \subseteq L \) be a given theory (i.e. a set of clauses) and \( \Delta \) a set of atoms. If \( p \lor c_1, \neg p \lor c_2 \in S \) for some \( p \in P \) and let \( S' = S \cup \{c_1 \lor c_2\} \), then \( \text{ind}_{\Delta}(\text{res}_{\Delta}(S)) = \text{ind}_{\Delta}(\text{res}_{\Delta}(S')) \).

\textbf{Proof} Let \( c \) denote the clause \( c_1 \lor c_2 \). If \( p \in \Delta \) then it’s easy to verify that the lemma holds for, then, \( c \in \text{res}_{\Delta}(S) \). Thus, \( \text{res}_{\Delta}(S') \subseteq \text{res}_{\Delta}(S) \). And, \( \text{ind}_{\Delta}(\text{res}_{\Delta}(S')) \subseteq \text{ind}_{\Delta}(\text{res}_{\Delta}(S)) \).

If \( p \notin \Delta \):

We consider the non-trivial case: \( \text{ind}_{\Delta}(\text{res}_{\Delta}(S')) \setminus \text{ind}_{\Delta}(\text{res}_{\Delta}(S)) \neq \emptyset \).

Let \( d \in \text{ind}_{\Delta}(\text{res}_{\Delta}(S')) \setminus \text{ind}_{\Delta}(\text{res}_{\Delta}(S)) \), we prove that \( \text{ind}_{\Delta}(\text{res}_{\Delta}(S)) \models d \).

As \( S' \setminus S = \{c\} \), there exists a minimal resolution-based derivation \( D = \langle d_1, \ldots, d_n \rangle \) in \( \text{res}_{\Delta}(S') \) such that \( d_n = d \) and \( c \in \{d_1, \ldots, d_n\} \). We now construct a derivation \( D' \) of \( d \) in \( \text{ind}_{\Delta}(\text{res}_{\Delta}(S)) \) through the construction of the derivation \( D \) as follows:

For each \( k \in \{1, \ldots, n\} \):
- if \( d_k \in S \) then \( \delta_k = \{d_k\} \),
- if \( d_k = c \) then \( \delta_k = \{c_1, c_2\} \),
- if \( d_k = \text{resolve}(d_{k_1}, d_{k_2}, p_k) \) for some \( p_k \in \Delta \) and \( k_1, k_2 < k \) and assuming, without loss of generality, that \( \neg p_k \in d_{k_1} \) and \( p_k \in d_{k_2} \), then \( \delta_k = \{\alpha \lor \beta \mid \neg p_k \lor \alpha \in \delta_{k_1}, \neg p_k \lor \beta \in \delta_{k_2}\} \cup \{g \in \delta_{k_1}, \delta_{k_2} \mid p_k \notin \text{atm}(\gamma)\} \setminus \{\theta \mid \text{there exists } q \in P \text{ such that } \{q, \neg q\} \in \alpha\} \).

We can now show the following:

For each \( k \leq n \):
- (i) \( \delta_k \neq \emptyset \),
- (ii) \( \bigcup_{\gamma \in \delta_k} \gamma \subseteq d_k \cup \{p, \neg p\} \), and

\footnote{The intuition behind the construction of \( \delta_k \) is to (i) obtain all possible resolvers over the atom \( p_k \); (ii) keep all clauses from previous constructions that do not contain the atom \( p_k \) or its negation; and (iii) remove all redundant clauses from the resulting set, i.e., clauses of the form \( \cdots \lor q \lor \cdots \lor \neg q \lor \cdots \).}
(iii) if either $p$ is a member of every $\gamma \in \delta_k$ or $\neg p$ is a member of every $\gamma \in \delta_k$, then $\bigcup_{\gamma \in \delta_k} \gamma \subseteq d_k$.

We prove the above by induction on $k$:

$k = 1$: Case $d_1 \in S$: Then, $\delta_1 = \{d_1\}$. The above are satisfied.

Case $d_1 = c$: Then, $\delta_1 = \{c_1, c_2\}$. The above are trivially satisfied.

$k \Rightarrow k + 1$: Case $d_{k+1} \in S \cup \{c\}$: Similar to the base case. Otherwise, let $i, j \leq k$ be such that $d_{k+1} = \text{resolve}(d_i, d_j, q)$ for some $q \in \Delta$.

We prove that $\delta_{k+1}$ satisfies (i):

Assume by way of contradiction that $\delta_{k+1} = \emptyset$, then

$\{\gamma \in \delta_i \cup \delta_j \mid q \notin \text{atm}(\gamma)\} = \emptyset$. (Note that $\{\gamma \in \delta_i \cup \delta_j \mid q \notin \text{atm}(\gamma)\} \cap \{\alpha \mid \text{there exists } r \in \Delta \text{ such that } r, \neg r \in \alpha \} = \emptyset$.) In other words, for each $\gamma \in \delta_i \cup \delta_j, q \in \text{atm}(\gamma)$.

From the inductive hypothesis, $\delta_i, \delta_j \neq \emptyset$. Hence, for each $\gamma \in \delta_i, \neg q \notin \gamma$ and for each $\gamma \in \delta_j, q \notin \gamma$. Thus, $\Gamma = \{\alpha \mid \neg q \notin \gamma, q \notin \gamma, \alpha \in \delta_i \text{ and } q \in \text{atm}(\gamma)\} \neq \emptyset$.

As $\delta_{k+1} = \emptyset$, for each $\gamma \in \Gamma$, there exists $r \in P$ such that $\{r, \neg r\} \subseteq \gamma$. That is, $r = p$; otherwise, as $\gamma \subseteq d_{k+1}$ we would have $d_{k+1} = \text{true}$, violating the minimality of $D$.

Therefore, it must be the case that there exists $l_p \in \{p, \neg p\}$ such that for each $\alpha \in \delta_i, l_p \in \alpha$ and for each $\beta \in \delta_j, \neg l_p \in \beta$.

From the inductive hypothesis (on condition (iii)), $\bigcup_{\gamma \in \delta_i} \gamma \subseteq d_i$ and $\bigcup_{\gamma \in \delta_j} \gamma \subseteq d_j$. Thus, $l_p \in d_i$ and $\neg l_p \in d_j$. Or, $\{p, \neg p\} \subseteq d_{k+1}$. Contradiction.

Therefore, $\delta_{k+1} \neq \emptyset$.

We prove that $\delta_{k+1}$ satisfies (ii):

For each $\gamma \in \delta_{k+1}$:

Case 1: $\gamma \in \delta_i \cup \delta_j$, and $q \notin \text{atm}(\gamma)$.

From the inductive hypothesis (on condition (ii)), $\gamma \subseteq d_i \cup \{p, \neg p\}$ and $\gamma \subseteq d_j \cup \{p, \neg p\}$.

Also, $d_{k+1} = \text{resolve}(d_i, d_j, q)$. Thus, $d_i, d_j \subseteq d_{k+1} \cup \{q, \neg q\}$. Therefore, $\gamma \setminus \{q, \neg q\} \subseteq d_{k+1} \cup \{p, \neg p\}$. But $q \notin \text{atm}(\gamma)$. Thus, $\gamma \setminus \{q, \neg q\} = \gamma$.

Therefore, $\gamma \subseteq d_{k+1} \cup \{p, \neg p\}$.

Case 2: $\gamma \in \{\alpha \mid \neg q \notin \gamma, q \notin \gamma, \alpha \in \delta_i \text{ and } q \in \text{atm}(\gamma)\}$.

From the inductive hypothesis, $\neg q \notin \gamma \subseteq d_i \cup \{p, \neg p\}$ and $q \in \text{atm}(\gamma)$. As $d_{k+1} = \text{resolve}(d_i, d_j, q)$, we have $d_i, d_j \subseteq d_{k+1} \cup \{q, \neg q\}$.

Note that $\alpha \setminus \{q, \neg q\} \subseteq \alpha \text{ and } \beta \setminus \{q, \neg q\} \subseteq \beta$. Thus, $\alpha \subseteq d_{k+1} \cup \{p, \neg p\}$ and $\beta \subseteq d_{k+1} \cup \{p, \neg p\}$. Or, $\alpha \setminus \beta \subseteq d_{k+1} \cup \{p, \neg p\}$. Thus, we have shown that for each $\gamma \in \delta_{k+1}$, $\gamma \subseteq d_{k+1} \cup \{p, \neg p\}$.

In other words, $\bigcup_{\gamma \in \delta_{k+1}} \gamma \subseteq d_{k+1} \cup \{p, \neg p\}$.

We prove that $\delta_{k+1}$ satisfies (iii):
From the symmetry of $p$ and $\neg p$, we may just consider the case of $p$ being a member of every $\gamma \in \delta_{k+1}$. The symmetric case for $\neg p$ should be similar.

Assume by way of contradiction that $\bigcup_{\gamma \in \delta_{k+1}} \gamma \notin d_{k+1}$. From condition (ii), which has been proved above (independently of the proof of this condition), we have $p \notin d_{k+1}$.

We have $p \neq q$ (as $p \notin \Delta$ and $q \in \Delta$). Thus, $p \notin d_{k+1}$ iff $p \notin d_i$ and $p \notin d_j$.

From the inductive hypothesis (on condition (iii)), there exists a clause $\gamma_i \in \delta_i$ such that $p \notin \gamma_i$ and there exists a clause $\gamma_j \in \delta_j$ such that $p \notin \gamma_j$. Now, if $q \notin \gamma_i$ then $\gamma_i \in \delta_{k+1}$ which is a contradiction as $p \notin \gamma_i$. (Note that $-q \notin \gamma_i$. Thus, $q \in \gamma_i$.

Similarly, $-q \in \gamma_j$.

Let $\gamma'_i$ and $\gamma'_j$ be such that $\gamma_i = q \lor \gamma'_i$ and $\gamma_j = -q \lor \gamma'_j$. Then, $\gamma'_i \lor \gamma'_j \in \delta_{k+1}$. But, $p \notin \gamma'_i \lor \gamma'_j$. Contradiction.

Therefore, if either $p$ is a member of every $c \in \delta_k$ or $-p$ is a member of every $c \in \delta_k$ then $\bigcup_{\gamma \in \delta_k} \gamma \subseteq d_k$.

**Theorem 4** Given a set of atoms $\Delta$ and a theory $S$, a clause $c$ is an independent consequence of $S$ with respect to $\Delta$ iff $\text{ind}_{\Delta}(\text{res}_{\Delta}(S)) \models c$.

**Proof** ($\Leftarrow$) Obvious from Fact 1.

($\Rightarrow$) If $S \models c$ and $c$ is $\Delta$-independent then we will show that $\text{ind}_{\Delta}(\text{res}_{\Delta}(S)) \models c$.

Let $D = \langle d_1, \ldots, d_n \rangle$ be a (resolution-based) derivation of false from $S \cup \{\neg c\}$, we prove that there is a derivation $D'$ of false from $\text{ind}_{\Delta}(\text{res}_{\Delta}(S)) \cup \{\neg c\}$ by induction on the length of $D$.

**Base case:** If $\text{length}(D) = 1$ then there exists $p \in P$ such that $p, \neg p \in S \cup \{\neg c\}$.

If $p \in \Delta$ then $p, \neg p \in S$. Thus false $\in \text{res}_{\Delta}(S)$. Therefore, false $\in \text{ind}_{\Delta}(\text{res}_{\Delta}(S))$. Or, $\text{ind}_{\Delta}(\text{res}_{\Delta}(S)) \models c$.

If $p \notin \Delta$ then it’s easy to verify that $p, \neg p \in \text{ind}_{\Delta}(\text{res}_{\Delta}(S)) \cup \{\neg c\}$. Therefore, $\text{ind}_{\Delta}(\text{res}_{\Delta}(S)) \models c$.

**Inductive case:** $\text{length}(D) = n > 1$.

Let $k < n$ be such that $d_k \notin S$ and $d_k = \text{resolve}(d_i, d_j, p)$ for some $d_i, d_j \in S$ and $p \in P$. Consider, a theory $S' = S \cup \{d_k\}$.

Then there is a proof $D'$ of false from $S' \cup \{\neg c\}$ such that $\text{length}(D') = n - 1$. From the inductive hypothesis, $\text{ind}_{\Delta}(\text{res}_{\Delta}(S')) \models c$.

From Lemma 1, $\text{ind}_{\Delta}(\text{res}_{\Delta}(S)) \models \text{ind}_{\Delta}(\text{res}_{\Delta}(S'))$.

Therefore, $\text{ind}_{\Delta}(\text{res}_{\Delta}(S)) \models c$. \qed

**Proposition 6** Given a set of atoms $\Delta$ and a theory $S$, $\text{Cn}(\text{ind}_{\Delta}(\text{res}_{\Delta}(S))) = \text{Cn}(\text{ind}_{\Delta}(\text{res}'_{\Delta}(S)))$.

**Proof** We first introduce some notation to assist in this proof:

Given a derivation $D$ and a clause $c \in D$, we say that a clause $d$ is an ancestor of $c$ in $D$ if $d$ is involved in the derivation of $c$ in $D$. Given a fixed theory $S$ and an atom $p \in P$ such that $D$ is a
derivation in \( S \), we are interested in the set of ancestors of \( c \) which are members of \( S \) whose sets of atoms contain \( p \). We identify two such sets: \( p \)-ancestor\(^+\)(\( c \)) whose members contain the positive occurrence of \( p \) and \( p \)-ancestor\(^-\)(\( c \)) whose members contain the negative occurrence of \( p \).

We prove the non-trivial case: \( \text{ind}_\Delta(\text{res}_\Delta(S)) \subseteq \text{ind}_\Delta(\text{res}'_\Delta(S)) \). Let \( c \in \text{ind}_\Delta(\text{res}_\Delta(S)) \) be given, we prove that \( c \in \text{ind}_\Delta(\text{res}'_\Delta(S)) \):

As \( c \in \text{res}_\Delta(S) \), there exists a sequence \( D_c = (c_1, \ldots, c_n) \) such that:

1. \( c = c_n \), and
2. for each \( k \leq n \), either \( c_k \in S \) or there exist \( i, j < k \) such that \( c_k \) is the result of resolution of \( c_i \) and \( c_j \) on some atom \( p \in \Delta \).

Without loss of generality we can assume that \( D_c \) is minimal and of the following form: for each \( k \leq n \), if there exist \( i, j < k \) such that \( c_k \) is the result of resolution of \( c_i \) and \( c_j \) on some atom \( p \in \Delta \) then \( p \in c_i \) and \( \neg p \in c_j \) and for each \( i \in p \)-ancestor\(^+\)(\( c_k \)), \( l \) is smaller than the index of any \( d \in p \)-ancestor\(^-\)(\( c_k \)) in \( D_c \).

Let \( H \) denote the set of \( \Delta \)-atoms that occur in \( D_c \), i.e. \( H = (\bigcup_{i=1}^n \text{atm}(c_i)) \cap \Delta \). We prove the above proposition by induction on the size of \( H \).

**Base case:** \( H = \emptyset \). Obviously, \( c \in S \). Thus, \( c \in \text{ind}_\Delta(\text{res}'_\Delta(S)) \).

**Inductive case:** Apparently, \( c \in \text{ind}_\Delta(\text{res}'_\Delta(S)) \) iff there exists a sequence \( G_c \) which satisfies similar conditions as those stated for \( D_c \). We construct such a derivation on the basis of \( D_c \) and the inductive hypothesis.

Let the sequence \( \langle p_{h(1)}, \ldots, p_{h(m)} \rangle \) be an order on the set \( H \) extracted from the order \( \langle p_1, \ldots, p_n \rangle \). Now we construct a derivation of \( c \) in \( \text{res}_\Delta(S) \) such that all the derivatives don’t contain any instance of \( p_{h(1)} \). Let \( p \) denote \( p_{h(1)} \). As \( c \in \text{ind}_\Delta(\text{res}_\Delta(S)) \), \( \text{atm}(c) \cap \Delta = \emptyset \), there exist a set \( I \subseteq \{1, \ldots, n\} \) such that for each \( k \in I \), there exist \( i, j < k \) such that \( c_k = \text{resolve}(c_i, c_j, p) \), and there does not exists any \( l \in \{1, \ldots, n\} \) such that \( l \) satisfies the above condition. As \( D_c \) is minimal, it’s easy to see that for each \( k \in I \), \( p \not\in \text{atm}(c_k) \).

Now we can proceed to constructing a sequence \( D'_c = \langle d_1, \ldots, d_n \rangle \) as follows:

For each \( k \leq n \),

- if \( p \not\in \text{atm}(c_k) \) then \( d_k = \{c_k\} \).
- otherwise, we consider two cases:

**Case 1:** \( p \in c_k \).

+ if \( c_k \in S \) then, let \( I_k = \{ j \in I : c_k \in p \)-\text{ancestor\(^+\)}(c_j) \}, \( d_k = \bigcup_{j \in I_k} \text{resolve}(c_k, p \)-\text{ancestor\(^-\)}(c_j), p) \).
+ otherwise, let \( i, j < k \) be such that \( c_k = \text{resolve}(c_i, c_j, p) \), then \( d_k = \text{resolve}(d_i, d_j, p) \).

**Case 2:** \( \neg p \in c_k \). From the assumption, \( k \) must be larger than the index of any member of \( D_c \) that satisfies case 1.

+ if \( c_k \in S \) then,\(^{12}\) let \( c_k = \neg p \lor c'_k \), \( d_k = \{d' \lor c'_k\} \).
+ otherwise, let \( i, j < k \) be such that \( c_k = \text{resolve}(c_i, c_j, p) \), then \( d_k = \text{resolve}(d_i, d_j, p) \).

\(^{12}\)essentially, after case 1 has been taken into account, the immediate positive \( p \)-parent of \( d \) for some \( d \) that is resulted as the result of resolving some clauses on \( p \) is exactly the negative \( S \)-ancestor of \( d \) except all instance of \( \neg p \) has been replaced by the clause from the positive branch.
To summarise, the above construction gives us a valid proof for $c$ such that the following is satisfied: All clauses that occur in this proof and contain some occurrence of $p$ is directly derived from clauses in $T$ which are of course must be members of $\text{res}_D^c(S)$. In other words, we have reduced the original derivation $D_c$ to a derivation $D'_c$ such that the only set of clauses that are not members of $\text{res}_D^c(S)$ contain only the atoms from $\{p_{h(2)}, \ldots, p_{h(m)}\}$. From the inductive hypothesis, we have this derivation can be reduced to a derivation $D''_c$ whose all members are in $\text{res}_D^c(S)$. 

Proposition 14 Let $S$ be a theory and $\Phi = \{\varphi_1, \ldots, \varphi_m\}$ a finite set of formulas, 
(i) $\text{ext}(S, \Phi)$ is an $S$-extension, 
(ii) for each theory $\sigma \in \text{ext}(S, \Phi)$, $\sigma$ is complete wrt $\Phi$, and 
(iii) different orders of applying $\varphi_1, \ldots, \varphi_m$ in general result in syntactically different sets of theories $\text{ext}(S, \Phi)$. However, they are logically equivalent.

Proof (i) We prove by induction on the size of $\Phi$:

Base case: $\Phi = \{\varphi\}$. It’s obvious from the definition of $\text{ext}(S, \varphi)$ that $\text{ext}(S, \varphi)$ is equivalent to $S$.

Inductive case: Assume that $\text{ext}(S, \Phi)$ is an $S$-extension for any set of formulas $\Phi$ such that $|\Phi| \leq k$.

We prove that $\Upsilon = \text{ext}(S, \Phi \cup \{\varphi\})$ is an $S$-extension for some set of formulas $\Phi$ such that $|\Phi| = k$. From the above definition, $\Upsilon = \text{ext}(\text{ext}(S, \Phi), \varphi) = \bigcup_{\sigma \in \text{ext}(S, \Phi)} \text{ext}(\sigma, \varphi)$.

From the inductive hypothesis, $\text{ext}(S, \Phi) = \{\theta_1, \ldots, \theta_l\}$ is an $S$-extension. In other words, $\text{Mod}(\theta_1 \lor \ldots \lor \theta_l) = \text{Mod}(S)$.

Then, $\omega \in \text{Mod}(T)$ iff there exists some $\theta_i \in \{\theta_1, \ldots, \theta_l\}$ such that $\omega \in \text{Mod}(\theta_i)$. Moreover, as $\text{ext}(\theta_i, \varphi)$ is an $\theta_i$-extension, $\text{Mod}(\theta_i) = \text{Mod}(\text{ext}(\theta_i, \varphi))$ for any $i = 1, \ldots, l$. Thus, $\omega \in \bigcup_{\sigma \in \text{ext}(T, \Phi)} \text{ext}(\sigma, \varphi)$. Or, $\omega \in \Upsilon$. The converse can be proved in a similar way.

(ii) The proof is also similar to (i) by induction on the size of $\Phi$.

(iii) The proof is an immediate corollary of Theorem 22 to be proved in the following. 

Before proving Theorem 22, we prove the following lemma:

Lemma 33 Let $e \in \text{Evt}$ be an event and $S$ a theory. If $S$ is complete wrt $\text{pre}(e)$ then $\text{post}(e, \omega) = \text{post}(e, \omega)$ for every $\omega \in \text{Mod}(S)$.

Proof As $S$ is complete wrt $\text{pre}(e)$, for each $\alpha = \langle \text{pre}_\alpha, \text{post}_\alpha \rangle \in \text{ED}(e)$, either $S \models \text{pre}_\alpha$ or $S \models \neg \text{pre}_\alpha$. Then, it’s obvious that for every $\omega \in \text{Mod}(S)$: For any $\alpha = \langle \text{pre}_\alpha, \text{post}_\alpha \rangle \in \text{ED}(e)$, $\omega \models \text{pre}_\alpha$ iff $S \models \text{pre}_\alpha$, and thus, $\text{post}(e, S) = \text{post}(e, \omega)$. 

Theorem 22 Let $D = (\text{Evt}, E\text{D})$ be a dynamic domain. Given a theory $S$ and an event $e \in \text{Evt}$, $\text{Mod}(\text{result}(S, e)) = \bigcup_{\omega \in \text{Mod}(S)} \text{nextState}(\omega, e)$.
Case 1: $atm(\varphi) \subseteq \Delta$. Then, $post(e, \sigma) \models \varphi$. Remark that for each partial interpretation $I \in TA(e, \omega)$, there exists some state $s \in nextState(\omega, e)$ such that $s(p) = I(p)$ for all $p \in \Delta$. Moreover, $TA(e, \omega)$ denotes the set of all possible partial interpretations for the set of atoms $\Delta$ that satisfy $post(e, \sigma)$. Thus $result(\sigma, e) \models \varphi$. Contradiction.
Case 2: \( \text{atm}(\varphi) \subseteq \bar{\Delta} \). Then, \( \sigma \models \varphi \). Remark that for each state \( s \in \text{nextState}(\omega, e) \), \( s(p) = \omega(p) \) for all \( p \in \bar{\Delta} \) and \( \bigcup_{\omega \in \text{Mod}(\sigma)} \text{nextState}(\omega, e) \subseteq \text{Mod}(\varphi) \). Thus, \( \text{Mod}(\sigma) \subseteq \text{Mod}(\varphi) \).

But then, \( \varphi \in \text{res}_\Delta(\sigma) \) and \( \text{atm}(\varphi) \cap \Delta = \emptyset \). Thus, \( \text{Mod}(\text{result}(\sigma, e)) \subseteq \text{Mod}(\varphi) \). Contradiction.

Case 3: \( \text{atm}(\varphi) \cap \Delta \neq \emptyset \) and \( \text{atm}(\varphi) \cap \bar{\Delta} \neq \emptyset \):

\[ \Rightarrow \text{the following three conditions hold:} \]

(i) for all \( s \in \bigcup_{\omega \in \text{Mod}(\sigma)} \text{nextState}(\omega, e) \) : \( s \cap \varphi \neq \emptyset \), and

(ii) there exists \( \rho \in \bigcup_{\omega \in \text{Mod}(\sigma)} \text{nextState}(\omega, e) \) such that \( \rho \cap \varphi_\Delta = \emptyset \), otherwise it would collapse to Case 1, and

(iii) there exists \( \varpi \in \bigcup_{\omega \in \text{Mod}(\sigma)} \text{nextState}(\omega, e) \) such that \( \varpi \cap \varphi_\bar{\Delta} = \emptyset \), otherwise it would collapse to Case 2.

Let \( \varpi \in \bigcup_{\omega \in \text{Mod}(\sigma)} \text{nextState}(\omega, e) \) be such that \( \varpi \cap \varphi_\bar{\Delta} = \emptyset \).

Let \( \omega \in \text{Mod}(\sigma) \) be such that \( \varpi \in \text{nextState}(\omega, e) \).

\[ \Rightarrow \omega \cap \varphi_\Delta = \emptyset \] (as \( \omega(p) = \varpi(p) \) for any \( p \in \bar{\Delta} \)).

\[ \Rightarrow \text{for all } s \in \text{nextState}(\omega, e) : s \models \varphi \] (iii) holds.

\[ \text{But, for all } s \in \text{nextState}(\omega, e) : s \models \varphi. \]

\[ \Rightarrow \text{for all partial interpretation } I \in TA(e, \omega), I \models \varphi \neq \emptyset. \] But then, for all \( s \in \bigcup_{\omega \in \text{Mod}(\sigma)} \text{nextState}(\omega, e) \): \( s \cap \varphi_\Delta \neq \emptyset \). Contradiction (with (ii) above). \qed

**Proposition 25** Let a deterministic dynamic domain \( D = (\text{Evt}, ED) \) be given. If \( D \) is an \( A \)-action description obtained from \( D \) by the above translation and \( \langle S, V, R \rangle \) is the transition system described by \( D \) then, for any theory \( T \) and any event \( E \in \text{Evt} \), \( \langle \omega, E, \omega' \rangle \in R \) if and only if \( \omega' \in \text{Mod}(\text{result}(T, E)) \) provided \( \omega \) is a model of \( T \).

**Proof** Assume that \( \omega \models T \), we prove that: \( \langle \omega, E, \omega' \rangle \in R \) (if and only if) \( \omega' \in \text{Mod}(\text{result}(T, E)) \)

iff \( \omega' \in \bigcup_{\omega \in \text{Mod}(T)} \text{nextState}(\omega, E) \) (from Theorem 22) iff there exists \( \varpi \in \text{Mod}(T) \) such that \( \omega' \in \text{nextState}(\varpi, E) \).

Since \( \omega \models T \), it suffices to prove that \( \langle \omega, E, \omega' \rangle \in R \) iff \( \omega' \in \text{nextState}(\omega, E) \):

We have \( \langle \omega, E, \omega' \rangle \in R \) iff \( \text{Res}(E, \omega) \subseteq \omega' \subseteq \text{Res}(E, \omega) \cup \omega \) (from Definition 24). On the other hand, we also have \( \omega' \in \text{nextState}(\omega, E) \) iff there exists \( \tau \in TA(\omega, \omega) \) such that \( \omega' = \text{update}(\omega, \tau) \). Thus, we only need to prove that \( \text{Res}(E, \omega) \subseteq TA(\omega, \omega) \).

Every event description \( \langle \text{pre}, \{l_1, \ldots, l_n\} \rangle \in ED(E) \) corresponds to the set of \( A \)-propositions \( \{E \text{ causes } l_i \text{ if } \text{pre} \}_n \). Thus, for \( i \in \{1, \ldots, n\} \): \( l_i \in \text{post}(E, \omega) \) iff \( \omega \models \text{pre} \) iff \( l_i \in \text{Res}(E, \omega) \).

Therefore, \( \text{Res}(E, \omega) \in TA(\omega, \omega) \). \qed

**Proposition 26** Let \( D \) be a (propositional) \( A \)-action description over the action signature \( \{\text{false, true}\}, F, A \) and the transition system \( \langle S, V, R \rangle \) be described by \( D \) (see [5] for the definitions of these concepts). If \( D = (\text{Evt}, ED) \) is the dynamic domain obtained by the above translation then for any \( \omega \in S \) and \( A \in A \),

\[ \text{Mod}(\text{result}(\omega, A)) = \{ \omega' \in S \mid \langle \omega, A, \omega' \rangle \in R \}. \]

**Proof** From Theorem 22, it suffices to prove that \( \omega' \in \text{nextState}(\omega, A) \) iff \( \langle \omega, A, \omega' \rangle \in R \). Similar to the proof of Proposition 25, we then only need to prove that \( \text{Res}(A, \omega) \in TA(A, \omega) \). Observe
that \( L \in \text{Res}(A, \omega) \) iff there is an \( A \)-proposition \( A \text{ causes } L \) if \( F \) and \( \omega \models F \) iff \( \langle F, L \rangle \in \text{ED}(A) \) (according to our translation) and \( L \in \text{post}(A, \omega) \). Thus, \( \text{Res}(A, \omega) \) satisfies the conditions of a partial interpretation over \( \text{atm}(\text{post}(A, \omega)) \) and \( \text{Res}(A, \omega) \models \text{post}(A, \omega) \). Therefore, \( \text{Res}(A, \omega) \in \text{TA}(A, \omega) \).

\[ Q.E.D \]

**Theorem 30** The proposed syntax-based update operator \( \circ_{ab} \) satisfies Katsuno-Mendelzon’s update postulates (U1), (U3), (U5), and (U7)–(U8).

**Proof** We introduce some notations to be used in the following proof. Let \( \Delta \subseteq P \) be a set of atoms and \( \omega \) an interpretation, we define: \( \omega|_\Delta \overset{\text{def}}{=} \{ \omega(p) : p \in \Delta \} \). In other words, \( \omega|_\Delta \) denotes the partial interpretation which is a restriction of \( \omega \) over the set of atoms \( \Delta \).

That \( \circ_{ab} \) satisfies (U1) is trivial as \( \mu \equiv \text{post}(E_\mu) \) is added to the theory after removing the affected formulas from the updated theory.

We will show that \( \circ_{ab} \) satisfies Katsuno-Mendelzon’s update postulate (U3). That is, if \( \text{Mod}(\psi) \) and \( \text{Mod}(\mu) \) are not empty then neither is \( \text{Mod}(\psi \circ_{ab} \mu) \). From Theorem 22, \( \text{Mod}(\psi \circ_{ab} \mu) = \text{Mod}(\text{result}(\psi, E_\mu)) = \text{Mod}(\psi) \cup \text{Mod}(\mu) \). From Theorem 22, it suffices to prove that \( \text{nextState}(\omega, E_\mu) \neq \emptyset \), for some interpretation \( \omega \). Since \( \text{pre}(\omega, E_\mu) = \text{true}, \mu \equiv \text{post}(\omega, E_\mu) \). Thus, \( \text{Mod}(\text{TA}(E_\mu, \omega)) \neq \emptyset \). Let \( \tau \in \text{TA}(E_\mu, \omega) \): update(\( \omega, \tau \)) \in \text{nextState}(\omega, E_\mu) \). Therefore, \( \text{nextState}(\omega, E_\mu) \neq \emptyset \).

We now show that \( \circ_{ab} \) satisfies (U5). From Theorem 22, it suffices to prove that:

\( \{ \omega \in \text{Mod}(\psi) : \text{nextState}(\omega, E_\mu) \cap \text{Mod}(\phi) \neq \emptyset \} \)

Let \( m \in \{ \omega \in \text{Mod}(\psi) : \text{nextState}(\omega, E_\mu) \cap \text{Mod}(\phi) \neq \emptyset \} \) and \( m \in \text{Mod}(\phi) \) and there exists \( \omega_0 \in \text{Mod}(\psi) \) such that \( m \in \text{nextState}(\omega_0, E_\mu) \). Therefore, \( m|_{\text{atm}(\phi)} \models \phi \) and \( m|_{\text{atm}(\mu)} \models \mu \) (for \( m \) is a model of \( \omega_0, E_\mu \) which is essentially the same as \( \mu \)). Thus, \( m|_{\text{atm}(\phi)} \cup m|_{\text{atm}(\mu)} \models \phi \land \mu \).

But, \( m|_{\text{atm}(\phi)} \cup m|_{\text{atm}(\mu)} = m|_{\text{atm}(\phi) \cup \text{atm}(\mu)} \). Therefore, \( m|_{\text{atm}(\phi) \cup \text{atm}(\mu)} \models \phi \land \mu \). Hence, \( m \in \text{nextState}(\omega, E_{\mu \land \phi}) \).

Before showing that \( \circ_{ab} \) satisfies (U7), we prove the following lemma:

**Lemma 34** Let \( \Delta, \Delta' \subseteq P \) be sets of atoms and \( S \) a theory. If \( \Delta \subseteq \Delta' \) then \( \text{ind}_\Delta(\text{res}_\Delta(S)) \models \text{ind}_{\Delta'}(\text{res}_{\Delta'}(S)) \).

**Proof (of the lemma)** Following Fact 1, \( \text{res}_\Delta(S) \) and \( \text{res}_{\Delta'}(S) \) are logically equivalent. However, as \( \Delta \subseteq \Delta' \), \( \text{ind}_\Delta(\text{res}_\Delta(S)) \models \text{ind}_{\Delta'}(\text{res}_{\Delta'}(S)) \).

Now, to prove that \( \circ_{ab} \) satisfies (U7), we only need to observe that \( \text{atm}(\mu_1) \subseteq \text{atm}(\mu_1 \lor \mu_2) \). Moreover, we also have \( \psi \circ_{ab} \mu_1 = \text{ind}_{\text{atm}(\mu_1)}(\text{res}_{\text{atm}(\mu_1)}(\psi)) \cup \{ \mu_1 \} \) and \( \psi \circ_{ab} (\mu_1 \lor \mu_2) = \text{ind}_{\text{atm}(\mu_1 \lor \mu_2)}(\text{res}_{\text{atm}(\mu_1 \lor \mu_2)}(\psi)) \cup \{ \mu_1 \lor \mu_2 \} \).

But \( \text{ind}_{\text{atm}(\mu_1)}(\text{res}_{\text{atm}(\mu_1)}(\psi)) \models \text{ind}_{\text{atm}(\mu_1 \lor \mu_2)}(\text{res}_{\text{atm}(\mu_1 \lor \mu_2)}(\psi)) \) (following the above lemma) and \( \{ \mu_1 \} \models \{ \mu_1 \lor \mu_2 \} \). Thus, we have \( \psi \circ_{ab} \mu_1 \) implies \( \psi \circ_{ab} (\mu_1 \lor \mu_2) \). In fact, \( \circ_{ab} \) satisfies the following postulate (U7') which is much stronger than KM postulate (U7):

\( \psi \circ_{ab} \mu_1 \) implies \( \psi \circ_{ab} (\mu_1 \lor \mu_2) \) and \( \psi \circ_{ab} \mu_2 \) implies \( \psi \circ_{ab} (\mu_1 \lor \mu_2) \).

We now show that \( \circ_{ab} \) satisfies (U8). From Theorem 22, it suffices to prove that:

\( \bigcup_{\omega \in \text{Mod}(\phi_1 \lor \phi_2)} \text{nextState}(\omega, E_\mu) = \bigcup_{\omega \in \text{Mod}(\phi_1)} \text{nextState}(\omega, E_\mu) \cup \bigcup_{\omega \in \text{Mod}(\phi_2)} \text{nextState}(\omega, E_\mu) \).

However, as \( \text{Mod}(\phi_1 \lor \phi_2) = \text{Mod}(\phi_1) \cup \text{Mod}(\phi_2) \), it’s easy to see that the above equation holds.\[Q.E.D]