Iterated Belief Change Implementation

Joel Faber
jfaber@cs.sfu.ca
School of Computing Science
Simon Fraser University
Burnaby, B.C.
Canada V5A 1S6

Abstract

We describe BE, a belief evolution solver for the modal action language $\mathcal{A}_\Box$. The action language $\mathcal{A}_\Box$ is an epistemic extension of the action language $\mathcal{A}$. This action language extends previous epistemic extentions of $\mathcal{A}$ by allowing erroneous initial beliefs and non-Markovian belief change. We describe our implementation using an AGM revision operator called the *topological revision operator* as described in (Hunter, 2006).

1 Introduction

The action language $\mathcal{A}_\Box$, first described in (Hunter, 2006), is an epistemic extension to the high level action language $\mathcal{A}$. In this paper we present an implementation, named `BE`, of a belief evolution solver for a simplified version of the $\mathcal{A}_\Box$ action language.

In section 2 we will briefly review the $\mathcal{A}_\Box$ action language. The general algorithm will be presented in section 3. In section 4 we will present details of the implementation using `smodels`. In section 5 we will discuss how to use the program `BE`.

2 Action Language $\mathcal{A}_\Box$

In this section we review the action language $\mathcal{A}_\Box$ as described in (Hunter, 2006). As $\mathcal{A}_\Box$ is an extension of the action description language $\mathcal{A}$, we refer the reader to (Gelfond and Lifschitz, 1998) for a definition of $\mathcal{A}$. This section recapitulates material taken from (Hunter, 2006).

2.1 Propositions

Let **A** be a fixed set of action symbols and let **F** be a fixed set of fluent symbols.

Definition 1 *A proposition of $\mathcal{A}_\Box$ is an expression of the form*

$$A \textbf{ causes } \phi \textbf{ if } F_1 \wedge \cdots \wedge F_p$$

where $A \in \mathbf{A}$, each F_i is a literal and ϕ is either a formula of the form $\Box\psi$ for some non-modal formula ψ.

Since we are interested in an epistemic modality, it is natural to think of propositions of the form

$$O \textbf{ causes } \Box\phi \textbf{ if } F_1 \wedge \cdots \wedge F_p$$

as descriptions of sensing action effects. We use the terms *sensing action* and *non-sensing action* to refer to actions with modal and non-modal effects, respectively. The symbol O will range over sensing actions and the symbol A will range over non-sensing actions. An *action description* is a set of propositions. The semantics of $\mathcal{A}_\Box$, defined by associating a transition function Φ_{AD} with every action description AD, is intended specifically for belief change with respect to a fixed AGM revision operator. …

2.2 World View

(Hunter, 2006) defines observation and action trajectories as the following.

Definition 2 *An observation trajectory of length n is an n-tuple $\bar{\alpha} = \langle \alpha_1, \ldots, \alpha_n \rangle$ where each $\alpha_i \in 2^S$.*

Definition 3 *An action trajectory of length n is an n-tuple $\bar{A} = \langle A_1, \ldots, A_n \rangle$ where each $A_i \in \mathbf{A}$.*

Definition 4 *Let $\bar{\alpha} = \langle \alpha_1, \ldots, \alpha_n \rangle$ be an observation trajectory and let $\bar{A} = \langle A_1, \ldots, A_n \rangle$ be an action trajectory. We say that $\bar{A}$ is consistent with $\bar{\alpha}$ if and only if there is a belief trajectory $\langle \kappa_0, \ldots, \kappa_n \rangle$ such that, for all i,*

1. *$\kappa_i \subseteq \alpha_i$*
2. *$\kappa_i = \kappa_{i-1} \diamond A_{i-1}$.*

If $\bar{A}$ is consistent with $\bar{\alpha}$, we write $\bar{A}||\bar{\alpha}$.

Definition 5 *A world view of length n is a pair $W = \langle \bar{A}, \bar{\alpha} \rangle$, where $\bar{\alpha}$ is an observation trajectory and $\bar{A}$ is an action trajectory, each of length n. We say W is consistent if $\bar{A}||\bar{\alpha}$.*

Definition 6 *Let T be a deterministic transition system, let $\bar{A} = \langle A_1, \ldots, A_n \rangle$ and let α be an observation. Then $\alpha^{-1}(\bar{A})$ denotes the set of all w such that there is a path from w to an element of α following the edges $A_1, \ldots, A_n$.*

If a world view W is inconsistent we need to remove observations such that we obtain a new world view that is consistent and contains the most reliable observations. From (Hunter, 2006),

Definition 7 *Let $W = \langle \bar{A}, \bar{\alpha} \rangle$ be a world view of length n. Define $\tau(W) = \langle \bar{A}, \bar{\alpha}' \rangle$, where $\bar{\alpha}' = \langle \alpha'_1, \ldots, \alpha'_n \rangle$ is defined by the following recursion.*

- *If $\alpha_n^{-1}(\bar{A}) \neq \emptyset$ then $\alpha'_n = \alpha_n$,*
  otherwise $\alpha'_n = 2^{\mathbf{F}}$.

- *For $i < n$, if*

$$\alpha_i^{-1}(\bar{A}_i) \cap \bigcap_{i<j} (\alpha'_j)^{-1}(\bar{A}_j) \neq \emptyset$$

  then $\alpha'_i = \alpha_i$,
  otherwise $\alpha'_i = 2^{\mathbf{F}}$.

3 Algorithm

It has been demonstrated that belief evolution under topological revision can be reduced to finding shortest paths in an underlying transition system. This subsection is a summary of the algorithm given in (Hunter, 2006).

If we consider action trajectories and observation trajectories of fixed finite length, the procedure can be described as follows. Let $\bar{A} = \langle A_1, \dots, A_n \rangle$ be an action trajectory of length n and let $\bar{\alpha} = \langle \alpha_1, \dots, \alpha_n \rangle$ be an observation trajectory of length n.

1. Determine $\alpha_{PRE} = \bigcap_i \alpha_i^{-1}(A_1, \dots, A_i)$.
2. Let $PATH$ denote the set of shortest paths from κ to α_{PRE}.
3. Let κ_0 be the set of terminal nodes on paths in $PATH$.
4. For $i \geq 1$, $\kappa_i = \kappa_0 \diamond A_1 \diamond \cdots \diamond A_i$.

In step 1 we determine α_{PRE}. If $P = \langle v_1, \dots, v_n \rangle$ is a path in the transition function Φ_{AD} such that each v_i is a vertex of Φ_{AD}, $v_i \in \alpha_i$, and v_{i+1} is obtained by executing A_i in state v_i then $v_1 \in \alpha_{PRE}$. Step 2 requires some mechanism to find the set of shortest paths from κ to α_{PRE}. Steps 3 and 4 are straightforward.

4 Implementation with `Smodels`

In this section we will discuss `BE`, a solver for the $\mathcal{A}_{\Box}$ action language. This implementation uses `smodels` API (Simons, 2000) but could be adapted to other answer set solvers such as `DVI`.

4.1 Action Description

Let $\mathbf{F}$ denote a set of fluent symbols and let $\mathbf{A}$ denote a fixed set of action symbols. An effect proposition is an expression of the form

$$A \textbf{ causes } L \textbf{ if } F$$

where $A \in \mathbf{A}$ and $F \in \mathbf{F}$. An action description is a set of effect propositions.

A logic program $\tau_n(AD)$, where AD is an action description, has answer sets that correspond to plans of length n. For each fluent F and $i \leq n$ the logic program $\tau_n(AD)$ contains the atoms $F_{pos}(i)$ and $F_{neg}(i)$. For each action A and $i < n$ the logic program contains the atoms $A_{pos}(i)$ and $A_{neg}(i)$. The logic program $\tau_n(AD)$ is composed of the following rules:

1. if B is an action or fluent symbol, then for each $i \leq n$ when B is a fluent symbol or each $i < n$ when B is an action symbol, $\tau_n(AD)$ contains the rule
$$\bot \leftarrow B_{pos}(i), B_{neg}(i)$$
2. for every proposition in AD of the form
$$A \textbf{ causes } F \textbf{ if } G_1 \wedge \cdots \wedge G_p \wedge \neg H_1 \wedge \cdots \wedge \neg H_q$$
and every $i < n$, $\tau_n(AD)$ contains the rules
$$\begin{array}{rcl} F(i+1) & \leftarrow & A(i), G_{1pos}(i), \ldots, G_{p_{pos}}(i), \\ & & H_{1neg}(i), \ldots, H_{q_{neg}}(i) \end{array}$$
3. if B is an action symbol and $i < n$ or if B is a fluent symbol and $i = 0$, then $\tau_n(AD)$ contains the rules
$$B_{neg}(i) \leftarrow \textit{not } B_{pos}(i)$$
$$B_{pos}(i) \leftarrow \textit{not } B_{neg}(i)$$
4. for every fluent symbol F and $i < n$, $\tau_n(AD)$ contains
$$F_{pos}(i+1) \leftarrow \textit{not } F_{neg}(i+1), F_{pos}(i)$$
$$F_{neg}(i+1) \leftarrow \textit{not } F_{pos}(i+1), F_{neg}(i)$$
5. for every $i < n$, and every distinct action symbols A_1 and A_2, $\tau_n(AD)$ contains the rules
$$A_{1neg}(i) \leftarrow A_{2pos}(i).$$

The rules in (1) are to simulate classical negation because `smodels` only supports negation as failure natively. Rules in (2) express the causal relationship between the fluent F the action A and the preconditions $G_1, \ldots, G_p, \neg H_1, \ldots, \neg H_q$. Rules in (3) state that all actions can be true or false and all fluents can initially be either true or false. Rules in (4) state that all fluents are inertial, and rules in (5) state that no two actions occur at the same time.

Intuitively, a literal F is *true* at time i if an and only if $F_{pos}(i)$ is in the answer set. Likewise, a literal $\neg F$ is *true* if and only if $F_{neg}(i)$ is in the answer set. Similarly, an action A is executed at time i if and only if $A_{pos}(i)$ is in the answer set. For any action or fluent B, $B_{pos}(i)$, called the positive atom of B at time i, is in the answer set if and only if B is *true* at time i. Likewise, $B_{neg}(i)$, called the negative atom of B at time i, is in the answer set if and only if B is $false$ at time i.

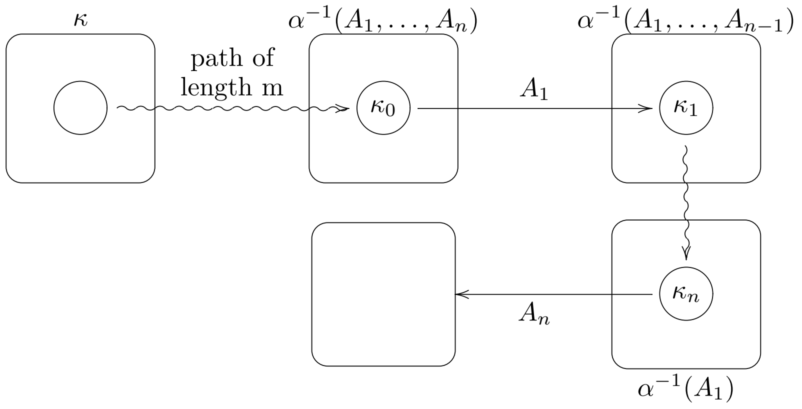

Figure 1: Visualizing Topological Revision

4.2 Extending $\tau_n(AD)$

We will extend $\tau_n(AD)$ to a new logic program $\tau_{n,m}(AD, K, W)$, where AD is an action description, K is a set of initial beliefs and W is a world view of the form $\langle \bar{A}, \bar{\alpha} \rangle$. The answer sets of $\tau_{n,m}(AD, K, W)$ correspond to plans in the action domain AD of length $n + m$ with the following properties.

1. m is the length of the subpath between some state s that satisfies the initial conditions K, and some state $s' \in \alpha^{-1}(A_1, \ldots, A_n)$,
2. n is the length of the world view W,
3. K is satisfied at time 0,
4. A_i executes at time $m + i - 1$,
5. α_i is satisfied at time $m + i - 1$.

Figure 1 illustrates the answer sets of $\tau_{n,m}(AD, K, W)$ graphically. The figure shows a large box representing κ; these are the states that satisfy the initial conditions K. The large box labeled $\alpha^{-1}(A_1, \ldots, A_n)$ is the set of states that can reach α by executing the actions $A_1, A_2, \ldots, A_n$. The boxes representing $\alpha^{-1}(A_1, \ldots, A_{n-1})$ and $\alpha^{-1}(A_1)$ have similar interpretations. The small circle inside $\alpha^{-1}(A_1, \ldots, A_n)$ represents the subset that is minimally distant from κ, which, in the context of topological revision, means the elements that can be reached from κ by a minimal path length.

4.2.1 Consistent with Initial Beliefs

Let K, the agent's initial beliefs, be the conjunction of literals $K_1 \wedge \cdots \wedge K_p$. To ensure that property (3) is satisfied we include the rules $K_{i pos}(0)$ for each $K_i \in \{K_1, K_2, \ldots, K_p\}$.

4.2.2 Consistent with Action Trajectory

Given an action trajectory $\bar{A} = \langle A_1, A_2, \ldots, A_n \rangle$ we need to add some rules to $\tau_{n,m}(AD, K, W)$ to ensure that all stable models correspond to plans that contain a subpath with edges $A_1, A_2, \ldots, A_n$. This is enforced using the rules $A_{i_{pos}}(m + i - 2)$ for every $i \leq n$.

4.2.3 Consistent with Observation Trajectory

Given an observation trajectory $\bar{\alpha} = \langle \alpha_1, \alpha_2, \ldots, \alpha_n \rangle$ we need to add some rules to $\tau_{n,m}(AD, K, W)$ to ensure that all stable models correspond to plans that are consistent with all the observations in $\bar{\alpha}$.

Observations are represented as well formed formulas using only conjunction, disjunction and negation connectives. Let F be a formula in disjunctive normal form that is equivalent to an observation α_i. Let the atom $F(m + i - 1)$ be in the stable model when the formula F at time $m + i - 1$ is asserted to be true. We also let the atoms $D_j(m + i - 1)$ represent the truth values of each of the j disjuncts of F. Our strategy is not to have an atom that is equivalent to the truth value of the formula but, rather to have the ability to assert that a formula must be true to be part of an answer set. Models that do not satisfy the formula will create an inconsistency and therefore will not be part of the answer set. We can ensure that all stable models have true observations at the appropriate time in the plan by adding the following rules to $\tau_{n,m}(AD, K, W)$ for each disjuctive formula F equivalent to an observation α_i:

1. for each disjunct D_j of F we add the rules

$$\begin{aligned} D_j(m + i - 1) \quad \leftarrow \quad & B_{1_{pos}}, \ldots, B_{k_{pos}}, \\ & B_{k+1_{neg}}, \ldots, B_{n_{neg}}, \end{aligned}$$

   where each B_h is a conjunct of D_j

2. the rules

$$\perp \leftarrow F(t), not\ D_1(m + i - 1), not\ D_2(m + i - 1), \ldots, not\ D_k(m + i - 1)$$

3. We assert that the formula must be true at time $m + i - 1$

$$F(m + i - 1)$$

4.3 Fallible Observations

If the world view W is inconsistent then the logic program $\tau_{n,m}(AD, K, W)$ has no answer sets. From definition 7 we can develop an algorithm using answer set programming to detect observations that are not consistent with newer observations.

Let $\tau_n(AD, W_p)$ be a logic program where AD is an action description and $W_p = \langle \bar{A}, \langle \alpha_p, \alpha'_{p+1}, \ldots, \alpha'_n \rangle \rangle$ is a world view such that α_p is an observation. Answer sets of $\tau_n(AD, W_p)$ correspond to plans of the action language AD of length n with the following properties, shown graphically in figure 2.

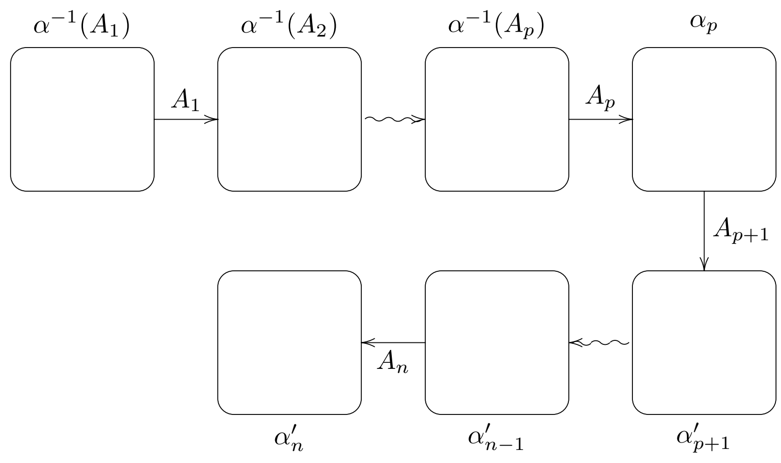

Figure 2: Visualizing Consistency Check

1. A_i executes at time i.
2. α_p is satisfied at time p.
3. α'_i is satisfied at time i for each $i > p$.

We can detect and remove inconsistent observations using the following algorithm.

1. Set $i = n$
2. Determine whether $\tau_n(AD, W_p)$ has any valid answer sets.
3. If there are no valid answer sets, set $\alpha'_i = \top$.
4. Set $i = i - 1$ and goto 2 if $i >= 0$.

The action trajectory $\bar{A}$ is consistent with the observation trajectory $\bar{\alpha'} = \langle \alpha'_1, \ldots, \alpha'_n \rangle$ and the logic program $\tau_{n,m}(AD, K, W')$, where $W' = \langle \bar{A}, \bar{\alpha'} \rangle$, has valid answer sets.

4.4 Belief Evolution

The solution to the logic program $\tau_{n,m}(AD, K, W')$ with minimal m is the solution to belief evolution using topological revision. Given AD, K, and W' as input we can find a solution minimal m using the following algorithm.

1. Set $m = 0$
2. Determine all answer sets to $\tau_{n,m}(AD, K, W')$.
3. Let $PATH$ be the corresponding set of paths.

   (a) If $PATH = \emptyset$, set $m = m + 1$ and goto 2
   (b) If $PATH \neq \emptyset$, then continue.

4. Let κ_i denote the set of states in $PATH$ at time $m + i$.
5. Return $\langle \kappa_0, \kappa_1, \ldots, \kappa_n \rangle$.

5 Usage

In this section we will discuss how to use `BE`. `BE` can be obtained at The SFU Computational Logic Lab's website `http://www.cs.sfu.ca/~cl/`.

5.1 Input

`BE` reads all input from standard in. Strings that contain only alphanumeric characters and the underscore ('_') character are accepted as action and fluent names.

`BE` uses the logical connectives '`-`' to represent classical negation, '`|`' to represent logical disjunction and '`&`' for logical conjunction. A literal is defined as a fluent symbol that, if negated, is immediately preceded by a negation symbol. A conjunction of literals is represented as a sequence of one or more literals connected with conjunction symbols.

A formula is similar to a conjunction of literals except it allows disjunctions and parenthesis to enforce the preferred order of operation. Formulae must be well formed or else an error message is printed and the program is terminated. For example, the formula

$$(A \wedge \neg B \wedge \neg C) \vee \neg(\neg D \wedge E \wedge F)$$

should be represented in `BE` syntax as the following (sans quotes)

"`(A & -B & -C) | -(-D & E & F)`".

An effect proposition of an action description AD is a statement of the form

A **causes** L,

where A is an action and L is a literal, or

A **causes** L **if** F,

where A and L are the same as above and F is a conjunction of literals. Effect propositions are represented as a line of input of the form

"`A causes L`"

or

"`A causes L if G1 &...& Gp`",

where `G1...Gp` are literals.

Command statements of the form

$$\kappa \circ \langle\langle A_1, \ldots, A_n\rangle, \langle\alpha_1, \ldots, \alpha_n\rangle\rangle,$$

where κ is the set of all possible initial worlds, $\langle A_1, \ldots, A_n\rangle$ is an action trajectory of ontic actions, and $\langle\alpha_1, \ldots, \alpha_n\rangle$ is an observation trajectory, is represented as a line of input of the form

“`|K1 & ...& Km| o <<A1, ..., An>, <OBS1, ..., OBSn>>`”.

Observations are represented as formulas. Only one command statement may be given per program execution.

Whitespace, except for line breaks, is ignored.

5.2 Output

The type of output can be controlled by using command line arguments.

- `-h` Display help and exit.
- `-p` Display all paths.
- `-t` Display transition system.
- `-k` Display new knowledge set.

The default is `-k`. The arguments `-p`, `-t`, and `-k` can be used together to display multiple type of output.

If there are no valid solutions to the problem (or the path length is greater than 100), then the only output is a warning message telling the user than no solution was found.

5.2.1 Display Paths

Each path, as illustrated in figure 1, are printed in the following format.

$$\begin{array}{rcl} \{State_0\} & <> & A_1 \\ \{State_1\} & <> & A_2 \\ \ldots & & \\ \{State_{n+m-1}\} & <> & A_{n+m} \\ \{State_{n+m}\} & & \end{array}$$

where each $State_i$ is a comma separated list of fluents that are true in that state it represents in the transition system at time i. Each path is separated by an empty line.

5.2.2 Display Transition System

The transition system is printed in the form

$$\{StartState\}\ Action\ \{EndState\},$$

where *StartState* and *EndState* are comma separated list of fluents that are true in that state it represents in the transition system.

5.2.3 Display New Knowledge Set

The new knowledge set is printed in the form

$$\begin{array}{l} k0\{ \\ \qquad \{k0_1\} \\ \qquad \{k0_2\} \\ \qquad \ \ \ldots \\ \ \ \} \\ \ldots \\ kn\{ \\ \qquad \{kn_1\} \\ \qquad \{kn_2\} \\ \qquad \ \ \ldots \\ \ \ \} \end{array}$$

where each kx_i is a comma separated list of fluents that are true in the state it represents.

6 Conclusion

We have presented an implementation of the $\mathcal{A}_{\Box}$ action language using answer set planning. Our implementation makes use of a topological revision operator as described in (Hunter, 2006). In future work we would like to generalized this revision operator to provide support for more complex revision operators. For example, one extension could be to minimize a path of weighted edges rather than find a path of shortest length. This extension would subsume the current implementation.

References

Michael Gelfond and Vladimir Lifschitz. Action languages. *Linköping Electronic Articles in Computer and Information Science*, 3(16), 1998. URL `http://www.ep.liu.se/ea/cis/1998/016/`.

Aaron Hunter. *Belief Change in the Presence of Actions and Observations: A Transition System Approach.* PhD thesis, Simon Fraser University, Burnaby, B.C. Canada, July 2006.

Aaron Hunter and James P. Delgrande. Iterated Belief Change: A Transition System Approach. *The Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, page 460, 2005.

Patrik Simons. *Extending and Implementing the Stable Model Semantics.* PhD thesis, Helsinki University of Technology, Otaniemi, Finland, 2000.