Achieving Completeness in Bounded Model Checking of Action Theories in ASP

Laura Giordano, Alberto Martelli, Daniele Theseider Dupré
Summary

- The paper deals with the verification of action theories by *Bounded Model Checking* (BMC).
- The action specification is given by a set of *temporal ASP rules* and a set of *constraints* in a temporal logic.
- *Completeness*: exploiting the Büchi automaton construction during the search of a path.
- Action domain and BMC encoded in standard *ASP*. 
Temporal action theory

Domain Specification $\mathcal{D}$

\[
\begin{align*}
[\text{deliv}(E)] & \neg \text{mail}(E) \\
[\text{send}] & \text{mail}(E) \leftarrow \neg \text{[send]} \neg \text{mail}(E) \\
\neg \Diamond & \text{mail}(E) \leftarrow \text{mail}(E), \neg \Diamond \neg \text{mail}(E) \\
\Diamond & \neg \text{mail}(E) \leftarrow \neg \text{mail}(E), \neg \Diamond \text{mail}(E) \\
[\text{deliv}(E)] & \bot \leftarrow \neg \text{mail}(E) \\
[\text{wait}] & \bot \leftarrow \text{mail}(E)
\end{align*}
\]

Constraints $C$ in DLTL (Dynamic Linear Time Temporal Logic)

\[
\begin{align*}
\langle \text{begin} \rangle & \top \\
\Box \langle \text{begin} \rangle & (\text{send}; (\text{deliv}(a) + \text{deliv}(b) + \text{wait}); \text{begin}) \top
\end{align*}
\]
Dynamic Linear Time Temporal Logic (DLTL)

\[ \Sigma \] be a finite non-empty alphabet of actions.
\[ \Sigma^\omega \] the set of infinite words on \( \Sigma \).

\( Prg(\Sigma) \) the set of programs (regular expressions)

\[ Prg(\Sigma) ::= a \mid \pi_1 + \pi_2 \mid \pi_1; \pi_2 \mid \pi^*, \text{ where } a \in \Sigma \]

[[\pi]] represents the set of executions of \( \pi \)

The set of formulas of DLTL(\( \Sigma \)):

\[ DLTL(\Sigma) ::= p \mid \neg \alpha \mid \alpha \lor \beta \mid \alpha \pi^* \beta \]

where \( p \in \mathcal{P} \) are atomic propositions
A model of DLTL(\(\Sigma\)) is a pair \(M = (\sigma, V)\) where \(\sigma \in \Sigma^\omega\) and \(V : prf(\sigma) \rightarrow 2^P\) is a valuation function.

Given a model \(M = (\sigma, V)\), and a prefix \(\tau\) of \(\sigma\), we can define the satisfiability of a formula at \(\tau\) in \(M\). In particular:

\(M, \tau \models \alpha \boxdot \tau \beta\) iff there exists \(\tau' \in [\tau]\) such that:

1. \(\tau \tau' \in prf(\sigma)\)
2. \(M, \tau \tau' \models \beta\)
3. \(M, \tau \tau'' \models \alpha\) for every prefix \(\tau''\) of \(\tau'\)

\[
\begin{array}{cccc}
\tau & \alpha \boxdot \beta & \tau' & \sigma \\
& \alpha & \cdots & \alpha \beta
\end{array}
\]
We can define derived modalities

- $\langle \pi \rangle \alpha \equiv \top U^\pi \alpha$
- $[\pi] \alpha \equiv \neg \langle \pi \rangle \neg \alpha$
- $\bigcirc \alpha \equiv \bigvee_{a \in \Sigma} \langle a \rangle \alpha$ (next)
- $\Diamond \alpha \equiv \top U^{\Sigma^*} \alpha$ (eventually)
- $\Box \alpha \equiv \neg \Diamond \neg \alpha$ (always)
- $\alpha U \beta \equiv \alpha U^{\Sigma^*} \beta$

DLTL has the full expressive power of the monadic second order theory of $\omega$-sequences.
Temporal action theory

Domain Specification $\Pi$

\[
\begin{align*}
[d\text{eliv}(E)] & \neg \text{mail}(E) \\
[s\text{ense}] & \text{mail}(E) \leftarrow \neg [s\text{ense}] \neg \text{mail}(E) \\
\Box & \text{mail}(E) \leftarrow \text{mail}(E), \neg \Box \neg \text{mail}(E) \\
\Box \neg & \text{mail}(E) \leftarrow \neg \text{mail}(E), \neg \Box \text{mail}(E) \\
[d\text{eliv}(E)] & \perp \leftarrow \neg \text{mail}(E) \\
[\text{wait}] & \perp \leftarrow \text{mail}(E)
\end{align*}
\]

Constraints $\mathcal{C}$ in DLTL (Dynamic Linear Time Temporal Logic)

\[
\langle \text{begin} \rangle T \\
\Box[\text{begin}] (s\text{ense}; (d\text{eliv}(a) + d\text{eliv}(b) + \text{wait}); \text{begin}) T
\]
Temporal answer sets of $\Pi$ [TPLP, 2012]

We have introduced a notion of temporal answer set for the set of temporal rules in a domain description $\Pi$.

Definition

Let $\sigma \in \Sigma^\omega$. A partial temporal interpretation $S$ over $\sigma$ is a set of temporal literals of the form $[a_1; \ldots; a_k]/l$, where $a_1 \ldots a_k$ is a prefix of $\sigma$, and it is not the case that both $[a_1; \ldots; a_k]/l$ and $[a_1; \ldots; a_k]^{\neg}/l$ belong to $S$ (namely, $S$ is a consistent set of temporal literals).

To define when a temporal interpretation $(\sigma, S)$ is a temporal answer set, we have extended the notion of answer set [Gelfond&Lifschitz 90].
Temporal answer sets of $\Pi$ satifying $C$

A total answer set of $\Pi$ can be seen as a run of $\Pi$, a linear temporal model $(\sigma,V)$

A transition system can be associated with $\Pi$

Temporal formulas can be evaluated over the transition system runs.

The run above is a run of $\Pi$ satisfying $C$
Verification of temporal action theories

• Given a domain description $\Pi$: Is there a run in the transition system of $\Pi$ satisfying property $\phi$? Does $\phi$ hold on all the runs of $\Pi$?

• Satisfiability and validity checks can be done by model checking techniques

• We can check for the validity of $\phi$ by verifying that $\neg\phi$ is unsatisfiable.

• Here, we exploit Bounded Model Checking (BMC) [Biere et al, 2003]
Bounded Model Checking (BMC)

- Looks for an infinite path in the transition system satisfying $\phi$
- The infinite path is represented as a finite path of length $k$ with a loop back from state $k$ to a previous state $l$: a $(k,l)$-loop
- The search proceeds iteratively, by increasing $k$ until a model satisfying $\phi$ is found, if one exists
Bounded Model Checking (BMC)

- A BMC problem can be efficiently reduced to a propositional satisfiability problem or to an ASP problem [Helianko and Niemela, 2003].
- Its main advantages are:
  - it finds counterexamples very fast;
  - it finds counterexamples of minimal length;
  - it uses much less space than Büchi automata
- If no model exists, the iterative procedure may never stop
Combining BMC and Büchi automata

• Building a path of length $k$ with a back loop of the product automaton, instead of considering paths on the transition system and then checking DLT formulas on them.

• Since the simple paths of the product automaton are finite, we have an upper bound to the value of $k$ given by the length of the longest simple path.
In the paper we show how to build a Büchi automaton for a given
DLTL formula $\phi$ using a tableau-like procedure.

States of the Büchi automaton are labelled by sets of signed
formulas, i.e. formulas prefixed with the symbol $T$ or $F$.
These formulas are expanded according to a set of tableau rules.
We formulate until formulas as

$$\alpha U^{A(q)} \beta$$

instead of $\alpha U^\pi \beta$

where $A(q)$ denotes an automaton $A$ with initial state $q$, such that
$L(A(q)) = [[\pi]]$.  

The rules have the following form:

Tor: \( T(\alpha \lor \beta) \Rightarrow T\alpha | T\beta \)

For: \( F(\alpha \lor \beta) \Rightarrow F\alpha, F\beta \)

Tneg: \( T\neg \alpha \Rightarrow F\alpha \)

Fneg: \( F\neg \alpha \Rightarrow T\alpha \)

\( T\alpha \cup(q) \beta \Rightarrow T(\beta \lor (\alpha \land \bigvee_{a \in \Sigma} (a) \lor_{q' \in \delta(q, a)} \alpha U^A(q') \beta)) \)  
where \( q \) is a final state of \( A \)

\( T\alpha \cup(q) \beta \Rightarrow T(\alpha \land \bigvee_{a \in \Sigma} (a) \lor_{q' \in \delta(q, a)} \alpha U^A(q') \beta) \)  
where \( q \) is not a final state of \( A \)

and similarly for \( F\)until.

where \( \alpha | \beta \) means: make two copies of the set of formulas and add \( \alpha \) to one of them and \( \beta \) to the other.
Building a path of the product automaton

The first states of the path of the product automaton for a domain description $\Pi$ and a formula $\varphi$ consists of:

- the set of fluents contained in the initial state of $\Pi$
- the signed formula $T\varphi$ and all formulas derived from it by the tableau rules

The next state is obtained by:

- building a set of fluents by applying the rules of $\Pi$ to the previous state
- propagating the temporal DLTL formulas (for instance, if the previous state contains $T(a)\alpha$, the next state will be reached through a transition $a$, and will contain $T\alpha$)

The last state $k$ must be connected to a previous state.
An Example

To achieve completeness, the upper bound is given by the length the longest simple path.

\[ F_0.1: T(\text{begin}) \top \]
\[ F_0.2: T(\text{begin})(A(y_0)) \top \]
\[ F_0.3: F(\text{begin})(A(y_0)) \top \]
\[ F_0.4: T(\text{begin})(A(y_0)) \top \]
\[ F_0.5: T \circ \Box_{\text{(mail(a)} \top \}
\[ F_0.6: F \circ \Box_{\text{(mail(a)} \top \}

\[ F_1.1: T(A(y_0)) \top \]
\[ F_1.2: T(A(y_0)) \top \]
\[ F_1.3: F(\text{begin})(A(y_0)) \top \]
\[ F_1.4: T(\text{begin})(A(y_0)) \top \]
\[ F_1.5: T(\text{begin})(A(y_0)) \top \]
\[ F_1.6: T(\text{begin})(A(y_0)) \top \]
\[ F_1.7: F(\text{mail(a)} \top \}
\[ F_1.8: F(\text{mail(a)} \top \]
\[ F_1.9: F(\text{mail(a)} \top \]
\[ F_1.10: F(\text{mail(a)} \top \]
Translation to ASP: the laws in $\pi$

The above procedure for building a path of the product automaton can be easily encoded in ASP. We have used the DLV-Complex extension of DLV. States are represented in ASP as integers from 0 to $K$.

For instance, the action law

$$[\text{go up}] \text{current}_\text{floor}(N) \leftarrow \text{next}_\text{floor}(N)$$

can be translated in the obvious way to:

\[
\text{holds(current}_\text{floor}(N),S') : - \\
\quad \text{next(S,S')}, \ \text{occurs(go-up,S)}, \ \text{holds(next}_\text{floor}(N),S).
\]
Translation to ASP: temporal formulas

LTL formulas are represented as ASP terms. The expansion of signed formulas can be formulated by means of ASP rules corresponding to the rules given in the tableau. For instance:

\[
\begin{align*}
\text{tt}(F_1,S) \lor \text{tt}(F_2,S) & : \text{tt}(\text{or}(F_1,F_2),S). \\
\text{ff}(F_1,S) & : \text{ff}(\text{or}(F_1,F_2),S). \\
\text{ff}(F_2,S) & : \text{ff}(\text{or}(F_1,F_2),S). \\
\text{or} & : \text{fluent}(F), \text{tt}(F,S), \text{not holds}(F,S).
\end{align*}
\]
Related work: Action Languages

Apart from the presence of the temporal constraints, the action language of this paper has strong relations with the languages $\mathcal{K}$ and $\mathcal{C}$.

- The logic programming based planning language $\mathcal{K}$ [Eiter et al., 2000, Eiter et al., 2004] is well suited for planning under incomplete knowledge and allows concurrent actions.
- The languages $\mathcal{C}$ and $\mathcal{C}^+$ [Giunchiglia et al., 1998, Giunchiglia et al., 2004] provide an account of causality and deal with actions with indirect and non-deterministic effects and with concurrent actions.

While our action language does not deal with concurrent actions and incomplete knowledge, our proposal extends the ASP approach for reasoning about dynamic domains with infinite computations by means of a temporal logic.
Related work: Bounded Model Checking

[Cimatti et al. 2003] developed an approach based on SAT.

Heljanko and Niemelä developed a compact encoding of bounded model checking of LTL formulas as the problem of finding stable models of logic programs.

In this paper, to achieve completeness, we follow a different approach to BMC which exploits the Büchi automaton construction.

- [Clarke et al. 2004] first proposed the use of Büchi automata in BMC. Our encoding in ASP does not assume the Büchi automaton is computed in advance.