LTL Model Checking

Sources:


The LTL Model Checking Problem

Same as CTL model-checking problem, except that system specifications are LTL formulas rather than CTL formulas.

**Given**

- Kripke structure $M = \langle S, A, R, \ell, s_I \rangle$
- LTL formula (in PNF) $\phi$

**Determine** Does $M$ sat $\phi$?
Recall sat for LTL

Let $M = \langle S, A, R, \ell, s_I \rangle$ be a Kripke structure.

- $\pi = s_0s_1\ldots \in S^\omega$ is an execution of $M$ if:
  - $s_0 = s_I$
  - For all $i \geq 0$, $\langle s_i, s_{i+1} \rangle \in R$ or for all $s \in S$, $\langle s_i, s \rangle \notin R$ and $s_{i+1} = s_i$.

$E(M)$: set of all executions of $M$.

- $\ell(s_0s_1\ldots) = \ell(s_0)\ell(s_1)\ldots$.

- $M$ sat $\phi$ if for all $\pi \in E(M), \ell(\pi) \models \phi$. 
LTL Model Checking Is Decidable for Finite-State Kripke Structures

Most common approach for checking $M$ sat $\phi$ relies on use of automata.

- $\neg \phi$ translated into finite-state automaton $B_{\neg \phi}$ that accepts sequences satisfying $\neg \phi$

- $B_{\neg \phi}$ is “composed” with $M$ to see if any sequences from $M$ satisfy $\neg \phi$.

This is referred to as the automaton-based approach to model checking.

Automata?
Büchi (Buechi) Automata

- Automata for recognizing *infinite* sequences of symbols. If $A$ is an alphabet:
  - Traditional finite-state automata accept words in $A^*$
  - Büchi automata accept words in $A^\omega$.

A Büchi automaton has form \( \langle S, A, \delta, s_I, F \rangle \), where

- \( S \) is a finite set of states
- \( A \) is an alphabet
- \( \delta \subseteq S \times A \times S \) is the transition relation
- \( s_I \in S \) is the start state
- \( F \subseteq S \) is the set of accepting states
Languages of Büchi Automata

Let $B = \langle S, A, \delta, s_I, F \rangle$ be a Büchi automaton, $w = a_0a_1\ldots \in A^\omega$.

- A run of $B$ on $w$ is a sequence $s_0s_1\ldots \in S^\omega$ such that:
  - $s_0 = s_I$
  - for all $i \geq 0$, $\langle s_i, a_{i+1}, s_{i+1} \rangle \in \delta$

- A run $s_0s_1\ldots$ is accepting if for all $i \geq 0$, there is $j > i$ such that $s_j \in F$.

- $B$ accepts $w$ if there is an accepting run of $B$ on $w$.

- The language, $L(B)$, of $B$ is defined as
  $$L(B) = \{ w \in A^\omega \mid B \text{ accepts } w \}.$$
Example

- Is $aaa... \in L(B)$?
- Is $ababaaaaa... \in L(B)$?
- Is $abababab... \in L(B)$?
- Is $abababbbbb... \in L(B)$? item What is $L(B)$?
Another Example

- Is $aaa\ldots \in L(B)$?
- Is $ababaaaaa\ldots \in L(B)$?
- Is $abababab\ldots \in L(B)$?
- Is $abababbbbb\ldots \in L(B)$?
- What is $L(B)$?
• Give a $w \in L(B)$.

• Give a $w \not\in L(B)$.

• What is $L(B)$?
Facts about Büchi Automata

Definition (Really, a theorem.) A language $W \subseteq A^\omega$ is omega-regular iff there is a Büchi automaton $B$ with $L(B) = W$.

Facts

1. Omega-regular languages are closed with respect to $\cup$, $\cap$, complementation.

2. Let $B$ be a Büchi automaton. It is decidable whether or not $L(B) = \emptyset$.

3. Let $M = \langle S, A, R, \ell, s_I \rangle$ be a Kripke structure. Then there is a Büchi automaton $B_M$ such that $L(B_M) = \{ \ell(\pi) \mid \pi \in E(M) \}$.

4. Let $\mathcal{A}$ be a finite set of atomic propositions, $\phi$ be an LTL formula built from $\mathcal{A}$, and $[\phi] = \{ \pi \in (2^\mathcal{A})^\omega \mid \pi \models \phi \}$. Then there is a Büchi automaton $B_\phi$ such that $L(B_\phi) = [\phi]$.

Note $M$ sat $\phi$ iff $L(B_M) \subseteq L(B_\phi)$ iff $L(B_M) \cap L(B_{\neg \phi}) = \emptyset$!
Fact 1: Closure of Omega-Regular Languages with respect to \( \cap \)

**Given** Büchi automata:
- \( B_1 = \langle S_1, A, \delta_1, s_1, F_1 \rangle \)
- \( B_2 = \langle S_2, A, \delta_2, s_2, F_2 \rangle \)

**Construct** Büchi automaton \( B_{12} = \langle S_{12}, A, \delta_{12}, s_{12}, F_{12} \rangle \) such that
\[
L(B_{12}) = L(B_1) \cap L(B_2).
\]

How? A variant of the *product construction* for finite automata.
- Run automata “in parallel”
- Accept when both automata “accept”
The Product Construction for Finite Automata

If $B_1 = \langle S_1, A, \delta_1, s_1, F_1 \rangle$, $B_2 = \langle S_2, A, \delta_2, s_2, F_2 \rangle$ are finite (not Büchi) automata, the product automaton $B_{12} = \langle S_{12}, A, \delta_{12}, s_{12}, F_{12} \rangle$ is given by:

- $S_{12} = S_1 \times S_2$
- $\langle \langle t_1, t_2 \rangle, a, \langle t'_1, t'_2 \rangle \rangle \in \delta_{12}$ if and only if $\langle t_1, a, t'_1 \rangle \in \delta_1$ and $\langle t_2, a, t'_2 \rangle \in \delta_2$
- $s_{12} = \langle s_1, s_2 \rangle$
- $F_{12} = F_1 \times F_2$

Does this work for Büchi?
Almost, But Not Quite

\[ L(B_1) = L(B_2) = \{aaaa \ldots \} \]

\[ L(B_{12}) = \emptyset \]

**Problem**

In product automaton, do not need to hit accepting states of two component automata at same time; only need to ensure that each is hit infinitely often.
A Büchi Product Construction: Idea

- Same intuition as production construction for finite automata: “run automata in parallel”
- For acceptance, flip back and forth between checking acceptance of two component automata
- Use a flag in the composite state to indicate which component’s acceptance is under investigation
Formalizing the Büchi Product Construction

Let $B_1 = \langle S_1, A, \delta_1, s_1, F_1 \rangle$, $B_2 = \langle S_2, A, \delta_2, s_2, F_2 \rangle$ be Büchi automata. Then the product automaton $B_{12} = \langle S_{12}, A, \delta_{12}, s_{12}, F_{12} \rangle$ is given by:

- $S_{12} = S_1 \times S_2 \times \{1, 2\}$

- $\langle \langle t_1, t_2, i \rangle, a, \langle t'_1, t'_2, i' \rangle \rangle \in \delta_{12}$ if and only if $\langle t_1, a, t'_1 \rangle \in \delta_1$, $\langle t_2, a, t'_2 \rangle \in \delta_2$, and

  $$i' = \begin{cases} 
3 - i & \text{if } t_i \in F_i \\
 i & \text{otherwise}
\end{cases}$$

- $s_{12} = \langle s_1, s_2, 1 \rangle$

- $F_{12} = \{ \langle t_1, t_2, i \rangle \in S_{12} \mid t_i \in F_i \}$
Unreachable states in $B_{12}$ are not shown.

**Facts**

1. $L(B_{12}) = L(B_1) \cap L(B_2)$

2. The time complexity for building $B_{12}$ is $O(|B_1| \cdot |B_2|)$, where

$$|B_1| = |S_1| + |\delta_1| + |F_1|$$
Fact 2: Emptiness Checking of Büchi Automata

When is a Büchi automaton non-empty?

This can be checked in $O(|B|)$ time!

1. Remove unreachable states from $B$.
2. Compute strongly connected components (SCCs) of remaining states.
3. Check if a nontrivial SCC contains an accepting state.
Fact 3: From Kripke Structures to Büchi Automata via Dualization

- Turn edges into states, states into edges.
- Add new start state.
- Every state is accepting state
- The alphabet consists of subsets of atomic propositions!
Fact 4: From LTL to Büchi Automata: The Tableau Construction

Goal  Convert PNF (= “positive normal form”) formulas into automata accepting the sequences making the formula true.

- Convert formulas into equivalent ones consisting of:
  - disjunctions of
  - conjunctions of
  - (negated) atomic propositions and X formulas

- Introduce transition for each disjunct

- Label of transition is set of atomic propositions “consistent with” atomic propositions in disjunct

- Target corresponds to set of formulas in X

- Accepting states used to track unsatisfied U
Recursive Characterizations of LTL Operators

... the basis of the tableau construction!

**Definition** \( \phi_1 \equiv \phi_2 \) means: \([\phi_1] = [\phi_2]\)

**Facts**

1. \( \phi_1 \cup \phi_2 \equiv \phi_2 \lor (\phi_1 \land X(\phi_1 \cup \phi_2)) \)

2. \( \phi_1 \lor \phi_2 \equiv \phi_2 \land (\phi_1 \lor X(\phi_1 \lor \phi_2)) \)

**Note** Right-hand sides consist of disjunctions, conjunctions of subformulas of left-hand side, \(X\) applied to left-hand side
Example Tableau Construction for $a \cup (b \cup c)$

Using recursive characterizations:

$$a \cup (b \cup c) \equiv c \lor (b \land X(b \cup c)) \lor (a \land X(a \cup (b \cup c)))$$

$$b \cup c \equiv c \lor (b \land X(b \cup c))$$
Facts about Tableau Construction

1. Construction can be defined to ensure that $|B_\phi|$ is $O(3^{|\phi|})$, where $|\phi|$ is number of subformulas in $\phi$.

2. Automaton-based model checking can be performed in $O(|M| \cdot 3^{|\phi|})$, where $|M|$ is size of Kripke structure.

Complexity seems much worse than for CTL. However, $|M| \gg |\phi|$, in general. Symbolic methods also possible. Tools like SPIN use automaton-based approach.