How LTL Model Checking Works

Paul Jackson
Paul.Jackson@ed.ac.uk

University of Edinburgh

Formal Verification
Autumn 2023

LTL semantics recap

Definition (Transition System)

A transition system $\mathcal{M} = \langle S, S_0, \rightarrow, L \rangle$ consists of

S	set of states
S_0	set of initial states
$\rightarrow \subseteq S \times S$	transition relation
$L : S \rightarrow \mathcal{P}(\mathrm{Atom})$	labelling function

such that $\forall s. \exists t.\ s \rightarrow t$.

Definition (Path)

A path in a model $\mathcal{M} = \langle S, S_0, \rightarrow, L \rangle$ is an infinite sequence of states $s_0, s_1, \ldots$ such that $s_0 \in S_0$ and $\forall i \geq 0.\ s_i \rightarrow s_{i+1}$. We write the path as $s_0 \rightarrow s_1 \rightarrow \ldots$.

The language accepted by a transition system

Take an automata-theoretic viewpoint on transition systems

- Consider
  - the set of states of a transition system as an *alphabet* Σ
  - each state is a *letter*
- Each infinite path π is then a word in Σ^ω
- The set of all paths of a transition system $\mathcal{M}$ is the *language* $\mathcal{L}(\mathcal{M})$ *accepted by* $\mathcal{M}$

Language of a formula

$$\mathcal{L}(\phi) = \{\pi \in S^\omega \mid \pi \models \phi\}$$

- Here ϕ is over the same atomic propositions as $\mathcal{M}$
- Alternate definitions of the language of a transition system and of a formula use $\mathcal{P}(\text{Atom})$ as the alphabet instead of the set of states S (see H&R book).
  - If state has a Boolean component for each element of Atom, definitions are equivalent.
  - In NuSMV, with integer range, array and word types for state components, there is a rich language of atomic propositions and $\mathcal{P}(\text{Atom})$ is usually larger than S.

Alternate presentation of LTL model-checking problem

The proposition

$$\mathcal{M} \models^{0} \phi$$

or equivalently

$$\forall \pi \in \mathsf{Paths}(\mathcal{M}).\ \pi \models^{0} \phi$$

can now be phrased as

$$\mathcal{L}(\mathcal{M}) \ \subseteq\ \mathcal{L}(\phi)$$

or equivalently

$$\mathcal{L}(\mathcal{M}) \ \cap\ \overline{\mathcal{L}(\phi)} \ =\ \emptyset$$

where $\overline{X}$ means $S^{\omega} - X$

Automata with same language as formulas

- In general, for each LTL formula there is not a transition system with the same language
- However, there is a *Büchi Automaton*:
- A *(Non-deterministic) Büchi Automaton* is a tuple

  $$\langle S, \Sigma, \rightarrow, S_0, A \rangle$$

  where
  - S is a set of states
  - Σ is an alphabet
  - $\rightarrow \subseteq S \times \Sigma \times S$ is the transition relation
  - $S_0 \subseteq S$ is the set of initial states
  - $A \subseteq S$ is the set of accepting states
- An infinite word is *accepted* by a BA iff there is some run of the BA for which some accepting state is visited infinitely often

LTL Model checking idea

- Observe $\overline{\mathcal{L}(\phi)} = \mathcal{L}(\neg\phi)$
- Let A_ϕ be a Büchi Automaton such that $\mathcal{L}(\phi) = \mathcal{L}(A_\phi)$
- For a suitable notion of *composition* $\mathcal{M} \otimes A$ of a transition system $\mathcal{M}$ and BA A, we have that

  $$\mathcal{L}(\mathcal{M} \otimes A) = \mathcal{L}(\mathcal{M}) \cap \mathcal{L}(A)$$

  Hence, to check

  $$\mathcal{M} \models^0 \phi$$

  instead check that

  $$\mathcal{L}(\mathcal{M} \otimes A_{\neg\phi}) = \emptyset$$

- *Fair CTL model checking* can be used to check for language emptiness.

Emulating Büchi Automata in NuSMV & nuXmv

Here is a transition system and LTL formula emulating a BA for checking $\mathbf{F}\,\neg p$

```
MODULE formula(sys)
  VAR
    st : {0, 1};
  ASSIGN
    init(st) := 0;
    next(st) := case
                  st = 0 & sys.p  : 0;
                  st = 0 & !sys.p : 1;
                  st = 1          : 1;
                esac;

    -- Accepting states are {1}.
    -- If true, there are no accepting paths
    LTLSPEC ! G F st = 1;

    -- FAIRNESS st = 1;
    -- CTLSPEC EG TRUE -- Does not work as expected
    -- CTLSPEC FALSE   -- Checks ! EG TRUE, as NuSMV only considers
                       -- fair start states, states where EG TRUE
```

Composing BA with a transition system

This composition checks LTL property Gp of `model`

```
MODULE model
  VAR
    st : 0..2;
  ASSIGN
    init(st) := 0;
    next(st) :=
      case
        st = 0 : {1,2};
        st = 1 : 1;
        st = 2 : 2;
      esac;
  DEFINE
    p := st = 0 | st = 1;
--    p := TRUE;

MODULE main
  VAR
    m : model;
    f : formula(m);
```

Model checking results 1

With definition in model

```
p := st = 0 | st = 1;
```

we get

```
Trace Type: Counterexample
-> State: 1.1 <-
  m.st = 0
  f.st = 0
  m.p = TRUE
-> State: 1.2 <-
  m.st = 2
  m.p = FALSE
-- Loop starts here
-> State: 1.3 <-
  f.st = 1
-- Loop starts here
-> State: 1.4 <-
-> State: 1.5 <-
```

Model checking results 2

With definiition in model

```
 p := TRUE;
```

we get

```
-- specification !( G ( F st = 1)) IN f is true
```