Formal Verification

Lecture 6: How LTL Model Checking Works

(Potted Version)

Jacques Fleuriot
jdf@inf.ac.uk
Recap

Previously:
  ▶ Model Checking CTL formulas

This time:
  ▶ Model Checking LTL
  ▶ Language-theoretic viewpoint
  ▶ From LTL formulas to automata (examples)
LTL Semantics recap

Definition (Transition System, with $S_0$ explicit)

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

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

such that $\forall s_1 \in S. \exists s_2 \in S. s_1 \rightarrow s_2$

Definition (Path)

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

Paths are written as: $\pi = s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \ldots$
The LTL Model Checking Problem

LTL model checking seeks to answer the question (with starting state omitted):

\[ \text{Does } \mathcal{M} \models \phi \text{ hold?} \]

or, equivalently:

\[ \text{Does } \forall \pi \in \text{Paths}(\mathcal{M}). \pi \models^0 \phi \text{ hold?} \]

where (recall) \( \pi \models^i \phi \) means “path at position \( i \) satisfies formula \( \phi \”).

- The universal quantification is over the infinite set of paths and each path is infinitely long
- How can we check infinitely many paths?
- CTL: use a fixed point characterisation of the sets of states
- LTL: sets of paths; a path is a sequence of symbols ...
  ... so use a language-theoretic approach.
The language accepted by a transition system

Fix a transition system $\mathcal{M} = \langle S, S_0, \rightarrow, L \rangle$

Let us consider the set of states $S$ as an *alphabet* $\Sigma$.

Each infinite path $\pi$ is then a word in the set $\Sigma^\omega$.

The set of all paths of $\mathcal{M}$ is the *language* $\mathcal{L}(\mathcal{M})$ accepted by $\mathcal{M}$.
The language accepted by a transition system

Fix a transition system $\mathcal{M} = \langle S, S_0, \rightarrow, L \rangle$

Let us consider the set of states $S$ as an *alphabet* $\Sigma$.

Each infinite path $\pi$ is then a word in the set $\Sigma^\omega$.

The set of all paths of $\mathcal{M}$ is the *language* $\mathcal{L}(\mathcal{M})$ accepted by $\mathcal{M}$.

Example:

<table>
<thead>
<tr>
<th>$\mathcal{M}$</th>
<th>$\mathcal{L}(\mathcal{M})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b \xrightarrow{a} c$</td>
<td>{ $abcccc\ldots$, $ababccccc\ldots$, $ababababcccc\ldots$, $\ldots$, $abababababababab\ldots$ }</td>
</tr>
</tbody>
</table>
Let $\phi$ be an LTL formula, and $S$ be the set of states of a model with the same set of atomic propositions as $\phi$.

Define the language $\mathcal{L}(\phi)$ of $\phi$ as:

$$\mathcal{L}(\phi) = \{ \pi \in S^\omega \mid \pi \models^0 \phi \}$$
Language of an LTL formula

Let $\phi$ be an LTL formula, and $S$ be the set of states of a model with the same set of atomic propositions as $\phi$.

Define the language $\mathcal{L}(\phi)$ of $\phi$ as:

$$\mathcal{L}(\phi) = \{\pi \in S^\omega | \pi \models^0 \phi\}$$

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).

If the state has a boolean component for each element of $\text{Atom}$, then the 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 $S$. 
Recall: LTL model checking seeks to answer the question:

Does $\mathcal{M} \models \phi$ hold?

or, equivalently:

Does $\forall \pi \in \text{Paths}(\mathcal{M}). \pi \models^0 \phi$ hold?

Using the presentation of transitions systems and formulas as languages, this can now be phrased as:

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

or, equivalently:

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

where $\overline{X}$ means $S^\omega - X$. 
Languages via automata

$\mathcal{L}(\mathcal{M})$ is defined in terms of a finite state transition system. Can LTL formulas be described in the same way?
Languages via automata

$\mathcal{L}(\mathcal{M})$ is defined in terms of a finite state transition system. Can LTL formulas be described in the same way?

No. In general, $\mathcal{L}(\phi)$ cannot be represented by a transition system. Can be represented by a related concept called a Büchi Automaton.
Languages via automata

$\mathcal{L}(\mathcal{M})$ is defined in terms of a finite state transition system. Can LTL formulas be described in the same way?

No. In general, $\mathcal{L}(\phi)$ cannot be represented by a transition system. Can be represented by a related concept called a Büchi Automaton.

A (non-deterministic) Büchi automaton $\langle S, \Sigma, \rightarrow, S_0, A \rangle$ consists of:

- $S$ a finite set of states
- $\Sigma$ an alphabet
- $\rightarrow \subseteq S \times \Sigma \times S$ transition relation
- $S_0 \subseteq S$ set of initial states
- $A \subseteq S$ set of accepting states

An infinite word is accepted by a Büchi automaton iff there is a run of the automaton on which some accepting state is visited infinitely often.
Example Büchi automata

Here, $\neg a$ means “any symbol that isn’t $a$”. States marked with $\odot$ are accepting.

F $a$: 

\[
\begin{array}{c}
\text{States marked with } \odot \text{ are accepting.}
\end{array}
\]

G $a$: 

\[
\begin{array}{c}
\text{States marked with } \odot \text{ are accepting.}
\end{array}
\]

a U b: 

\[
\begin{array}{c}
\text{States marked with } \odot \text{ are accepting.}
\end{array}
\]

(Can also do them without the error paths.)

For the general construction for any formula $\phi$, see H&R, Section 3.6.3.
LTL Model Checking Idea

We reformulated the LTL model checking problem to:

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

Now:

1. Observe that \( \overline{\mathcal{L}(\phi)} = \mathcal{L}(\neg \phi) \)
2. Let \( A_\phi \) be a Büchi automaton such that \( \mathcal{L}(\phi) = \mathcal{L}(A_\phi) \).
3. For a suitable notion of composition \( \mathcal{M} \otimes A \) of a transition system \( \mathcal{M} \) and a Büchi automaton \( A \), we have that

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

4. So, to check \( \mathcal{M} \models \phi \), instead check

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

5. Use *Fair CTL model checking* to check this last property. See H&R.
Example: Model Checking LTL formula $G \ p$

1. Construct an automaton $A_{\neg G \ p} = A_{F \ 
eg p}$ for $F \ 
eg p$, which takes as input infinite paths of states of a model $\mathcal{M}$ and accepts just those paths that satisfy $F \ 
eg p$.

2. Compose $A_{F \ 
eg p}$ and $\mathcal{M}$ and ask whether the language of the composition is empty.

3. If the language is empty, then we know that $G \ p$ is satisfied by $\mathcal{M}$. If not and we exhibit an accepting path, then that path is a counter-example to $G \ p$: it both is a path in $\mathcal{M}$ and it satisfies $A_{F \ 
eg p} = A_{\neg G \ p}$.

The next few slides examine this within the context of NuSMV.
Emulating Büchi automata in NuSMV

Here is a transition system and LTL formula *emulating* a Büchi automaton $A_F \neg p$ for checking $F \neg p$:

-- A 2 state automaton for $F \neg p$.

```plaintext
MODULE formula(sys)
  VAR
    st : { 0, 1 };
  ASSIGN
    init(st) := 0;
    next(st) := case
        -- loop in state 0 if $p$ is always true
        st = 0 & sys.p : 0;
        -- If ever $p$ is false, transition to state 1
        st = 0 & !sys.p : 1;
        -- then loop forever more in state 1
        st = 1 : 1;
    esac;

  -- Accepting states: {1} as $st = 1$ occurs infinitely often

  -- LTL expression of acceptance condition:
  -- Specification is true just when there are no accepting paths

  LTLSPEC ! G F st = 1;
```
Composing Büchi automaton and transition system

This composition checks LTL property $G \ p$ of the model:

-- A model M with 2 alternative definitions of a state property p

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 this definition in the model:

\[ p := \text{st} = 0 \lor \text{st} = 1; \]

we get:

-- specification \( \neg (G (F \text{st} = 1)) \) IN f is false
-- as demonstrated by the following execution sequence
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 <-

The acceptance condition for a run in this composition is just the acceptance condition for a run of the formula automaton.
With this definition in the model:

\[ p := \text{TRUE}; \]

we get:

\[ \text{-- specification } !\left( G \left( F \text{ st } = 1 \right) \right) \text{ IN } f \text{ is true} \]
Summary

- LTL Model Checking (H&R 3.6.2, 3.6.3)
  - Transition systems and formulas as languages
  - Formulas as Büchi automata
  - Simulating Büchi automata in NuSMV

- Next time: Binary Decision Diagrams

  [BDDs are] one of the only really fundamental data structures that came out in the last twenty-five years.
  — Donald Knuth “Fun with Binary Decision Diagrams”