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)

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 $\forall s_1. \exists s_2. s_1 \rightarrow s_2$

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:

\[ \mathcal{M} \models^0 \phi \]

or, equivalently:

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

- 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 sequences 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:

\[
\begin{array}{c|c}
\mathcal{M} & \mathcal{L}(\mathcal{M}) \\
\begin{tikzpicture}[baseline=(a.base)]
  \node (a) at (0,0) {$a$};
  \node (b) at (-1,-1) {$b$};
  \node (c) at (1,-1) {$c$};
  \draw[->] (a) to (b);
  \draw[->] (b) to (c);
  \draw[->] (c) to (a);
\end{tikzpicture} & \{ abc\cdots, \\
  ababc\cdots, \\
  abababc\cdots, \\
  ababababc\cdots, \\
  \cdots, \\
  ababababababab\cdots \} \\
\end{array}
\]
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:

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

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

Alternate definitions of the language of a transition system and of a formula use $P(Atom)$ as the alphabet instead of the set of states $S$ (see H&R book).

If the state has a boolean component for each element of $Atom$, 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 $P(Atom)$ is usually larger than $S$. 
Language-theoretic presentation of validity

Recall: LTL model checking seeks to answer the question:

\[ \mathcal{M} \models^0 \phi \] hold?

or, equivalently:

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

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

\[ \mathcal{L}(\mathcal{M}) \subseteq \mathcal{L}(\phi) \]

or, equivalently:

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

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

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

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

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

\( \mathcal{L}(M) \) 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 $a$: 

G $a$: 

$a \text{ U } b$: 

(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 \( \mathcal{M} \models^0 \phi \), 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.
Emulating Büchi automata in NuSMV

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

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: {1}.
LTLSPEC ! G F st = 1; -- If true, no accepting paths

-- How to check EG TRUE for fair paths in NuSMV
-- FAIRNESS st = 1;
-- CTLSPEC FALSE;
Composing Büchi automaton and transition system
This composition checks LTL property $G \, p$ of the 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 this definition in the model:

\[ p := st = 0 \mid st = 1; \]

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 <-
With this definition in the model:

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

we get:

```
-- specification ! ( G ( F \text{st} = 1)) \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:
  - Guest Lecture: Dr. Brian Campbell
    “Automated Reasoning for Testing Specifications and Programs”

- Next Friday: 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”