Formal Verification

Lecture 4: Practical Exercise
Model Checking with NuSMV

Jacques Fleuriot    Daniel Raggi

January 27, 2017
Overview

The practical is in 2 parts:

1. Exercises with Temporal Logic
   1.1 Exercises on LTL
   1.2 Exercises on CTL

2. Verifying a traffic-light system
   2.1 Properties to Verify
   2.2 Model bug to fix
   2.3 Principles of LTL Model Checking

This coursework is **not** assessed
Objectives of the Practical

1. Acquire familiarity with the semantics of LTL formulas.
2. Acquire familiarity with the semantics of CTL formulas.
3. Experience in specifying a (toy) system using LTL and CTL.
4. Debugging a system using LTL model checking.
5. Exploring the principles of LTL model checking.
6. In general, gain knowledge of what model checking is capable of.
2.1 LTL

Write this model in NuSMV:

Use NuSMV to analyse some LTL properties ($G\ a,\ a\ U\ b,\ ...$):

1. Determine whether each property is valid (i.e., true for all paths)
2. Get NuSMV to generate a path that satisfies the formula
2.2 CTL

Equivalence of CTL formulas (*CTL will be covered in the next lecture*)

You are given a list of pairs of CTL formulas, you have to determine whether or not each pair is equivalent.

\[ \phi \equiv \psi \iff (\forall M \ s. M, s \models \phi \iff M, s \models \psi) \]

For each pair:

- If they are equivalent, then explain why, with reference to the semantics of CTL.
- If they are not equivalent, then give a NuSMV model that distinguishes them.
An example: the formulas $\textbf{EF} \ \phi$ and $\textbf{EG} \ \phi$.

Not equivalent, this model distinguishes them:

\[
\begin{array}{c}
\neg \phi \\
\rightarrow \\
\phi \\
\end{array}
\]

For this model, $\textbf{EF} \ \phi$ is true, but $\textbf{EG} \ \phi$ is false.
Verifying a traffic-light system for a one-lane bridge

Arrangement of sensors and traffic-lights for a one-lane bridge across a river:

The sensors are

- \texttt{at\_w}: cars are present at the west end, ready to go onto the bridge,
- \texttt{at\_e}: cars are present at the east end, ready to go onto the bridge,
- \texttt{on\_bridge}: cars are present on the bridge.

The traffic-light controller generates outputs with values \texttt{rd} (red) or \texttt{gn} (green) that control the traffic light at each end of the bridge:

- \texttt{w\_light} for the light for traffic coming onto the bridge from the west end,
- \texttt{e\_light} for the light for traffic coming onto the bridge from the east end.
Verifying a traffic-light system for a one-lane bridge

The file `bridge.smv` contains a NuSMV model for the traffic light controller, which has 2 state variables `state` and `tim`. The state machine for the `state` state variable is shown below.

Implicit in the diagram are self-loop transitions for all conditions not explicitly described.
3.2 Properties to Verify

You are given several properties to state in temporal logic, and to verify against the model using NuSMV.

1. 5 properties in LTL
2. 2 properties in CTL

Note: In many cases, you will have to refine your formalized properties with additional assumptions in order to get it to hold for the model.

But remember to stay true to the “spirit” of the property as stated in English.
3.3 Model Bug to Fix

The controller model has a bug. The bug is highlighted by the failure of the following property to verify:

Whenever there is a car at the east light, eventually the east light will go green, under the assumptions

1. if cars are on the bridge, eventually all cars are off the bridge,
2. if a car is at a red light, it stays at that light up to and including the first step (if there is one) at which the light turns green.

You have to:

1. Express this property in LTL, and get NuSMV to demonstrate a counterexample.
2. Explain what the bug is
3. Fix the model

You can use bounded model checking to get shorter counterexamples.
Principles of LTL model checking

Roughly, LTL model checking works by the following process:

1. Start with some formula $\phi$.
2. Negate it: $\neg\phi$.
3. Convert $\neg\phi$ to a finite state automaton: a Büchi automaton.
4. Construct the intersection of the formula’s automaton and the model’s automaton.
5. Check to see whether the resulting automaton’s language is empty.
Summary

- This time: Practical Exercise
  - LTL and CTL
  - Verifying a traffic-light system for a one-lane bridge

- Next time:
  - CTL: Computation Tree Logic
  - A Branching-time temporal logic