A Practical Demonstration of the Model Checker NuSMV\textsuperscript{1}

Viraj Wijesuriya

Computer-Aided Formal Verification
Week 6, Michaelmas term 2018

\textsuperscript{1}The slides are provided, courtesy of Nathalie Cauchi
What is NuSMV

NuSMV: a symbolic model checker

- the first model checker based on BDDs
- open architecture for model checking, which can be reliably used for the verification of industrial designs, as a core for custom verification tools, as a testbed for formal verification techniques, and applied to other research areas.

² nusmv.fbk.eu
Application

We will perform two tasks:

1. We will first use the tool to encode transition systems and LTL and CTL formulas to be model checked.
2. We will use the tool to perform bounded model checking.
Transition systems in NuSMV

```plaintext
MODULE main
VAR
state :{s0,s1,s2,s3,s4};
ASSIGN
init(state) := {s0};
next(state) := case
state=s0  :  s1;
state=s1  :  {s3, s4};
state=s2  :  s2;
state=s3  :  s2;
state=s4  :  s4;
esac;
DEFINE
a := state=s0 | state=s1;
b := state=s1 | state=s3;
c := state=s2 | state=s3 | state=s4;
```

STARTING STATE: `s0`

```plaintext
diagram
```

**States:**
- `s0`
- `s1`
- `s2`
- `s3`
- `s4`

**Transitions:**
- From `s0` to `s1` on `a`
- From `s1` to `{s3, s4}` on `a` and `b`
- From `s2` to `s2` on `c`
- From `s3` to `s2` on `c`
- From `s4` to `s4` on `c`

Viraj Wijesuriya
A Practical Demonstration of the Model Checker NuSMV
Remark

- The NuSMV code is saved in a text file with extension `.smv`
  
  ```
  TS1.smv
  ```

- Unlike SPIN, NuSMV can handle *multiple initial states* in the verification process. Hence, we only need to run the verification once.

- Can model check both LTL and CTL properties.
An **LTL formula** consists of atomic proposition(s), boolean operator(s) and temporal operator(s)

A **CTL formula** consists of atomic proposition(s), boolean operator(s), temporal operators and **path quantifier(s)**

<table>
<thead>
<tr>
<th>operator</th>
<th>math</th>
<th>NuSMV</th>
</tr>
</thead>
<tbody>
<tr>
<td>not</td>
<td>¬</td>
<td>!</td>
</tr>
<tr>
<td>and</td>
<td>∧</td>
<td>&amp;</td>
</tr>
<tr>
<td>or</td>
<td>∨</td>
<td></td>
</tr>
<tr>
<td>implies</td>
<td>→</td>
<td>-&gt;</td>
</tr>
<tr>
<td>equivalent</td>
<td>⇔</td>
<td>&lt;--&gt;</td>
</tr>
<tr>
<td>always</td>
<td>□</td>
<td>G</td>
</tr>
<tr>
<td>eventually</td>
<td>◊</td>
<td>F</td>
</tr>
<tr>
<td>until</td>
<td>U</td>
<td>U</td>
</tr>
<tr>
<td>next</td>
<td>◯</td>
<td>X</td>
</tr>
<tr>
<td>for all</td>
<td>∀</td>
<td>A</td>
</tr>
<tr>
<td>exist</td>
<td>∃</td>
<td>E</td>
</tr>
</tbody>
</table>
Examples

- Some examples of the translation of LTL /CTL formula from mathematical notations to NuSMV commands

<table>
<thead>
<tr>
<th>LTL/CTL formula</th>
<th>NuSMV</th>
</tr>
</thead>
<tbody>
<tr>
<td>◇□c</td>
<td>FG c</td>
</tr>
<tr>
<td>□◇c</td>
<td>GF c</td>
</tr>
<tr>
<td>(□¬c) → (□□c)</td>
<td>(X ! c) → (X X c)</td>
</tr>
<tr>
<td>□a</td>
<td>G a</td>
</tr>
<tr>
<td>aU□(b v c)</td>
<td>a U (G (b</td>
</tr>
<tr>
<td>(□□b)U(b v c)</td>
<td>(X X b) U (b</td>
</tr>
<tr>
<td>∃◇∀□c</td>
<td>EF AG c</td>
</tr>
<tr>
<td>∀□∃◇¬c</td>
<td>AG EF !c</td>
</tr>
</tbody>
</table>
Preparing a NuSMV file TS1.smv

▶ Attach to the file TS1.smv the following code:

LTLSPEC F G a
CTLSPEC EF AG c
Verification using **NuSMV**

- To verify the transition system against the given specification(s), execute the NuSMV with the parameter name of the smv file:

  **NuSMV TS1.smv**

- NuSMV automatically generates a counter-example when a specification is not satisfied.
Exercise 1

- Verify the transition system used in example (TS1.smv) against the following properties:

  - □⋄ c
  - □□¬b
  - ∀⋄ ∀□c
  - ∃⋄ (a ∧ b ∧ ∀□ b)
  - ∀□(b → ∀□ c)
  - ∀□(a ↔ ¬c)

- In each case, explain why the property was satisfied or not.
Exercise 2

Consider the transition system on the left

Encode the transition system (e.g. TS2.smv)
Exercise 2

- Verify the transition system (TS2.smv) against the following properties:
  - $◊ □ c$
  - $□ ◊ c$
  - $(□ \neg c) \rightarrow □(□ c)$
  - $aU(□(b \lor c))$
  - $∃ ◊ (∀ □ c)$
  - $∀ □ (∃ □ b)$

- In each case, explain why the property was satisfied or not.
Bounded Model Checking

Recall:
- employs a SAT solver for model checker
- focuses on counterexample generation (up to a certain length)

We will now perform bounded model checking on a transition system.
Consider the above transition system

Encode the transition system (e.g. TS3.smv)
Bounded Model Checking: Exercise

- Verify the transition system (e.g. TS3.smv) against the following properties using bounded model checking:
  - □⋄a
  - □□(a → (b → ◊c))
  - □(a ∧ (◊c → ◊a))

- To do bounded model checking:
  - NuSMV -bmc -bmc_length 2 TS3.smv

- Run bounded model checking with different maximum counterexample length and comment on result
Bounded Model Checking: Extra Reading

Read the tutorial on bounded model checking using NuSMV found in the below link (pages 20 - 28):

Bonus Exercise

Determine whether the two formulas are equivalent:

$$\exists \Diamond (\exists \square p) \text{ and } \exists \square (\exists \Diamond p)$$