**COMP424/524-06A**

**Topics in Software Engineering**

Part 1 – CTL Model Checking

8. Introduction to NuSMV

Robi Malik

---

**NuSMV as a Tautology Checker**

-- NuSMV can be used to check
-- propositional tautologies.

**MODULE** main

Declaration of
main module

**VAR**

a: boolean;
b: boolean;
c: boolean;

**DECLARATION OF VARIABLES**

A formula to
be checked

**SPEC**

\(((a \land b) \rightarrow c) \leftrightarrow ((a \rightarrow (b \rightarrow c))\)

---

**The NuSMV Model Checker**

SMV – Symbolic Model Verifier

- Originally developed by K. L. McMillan and E. M. Clarke at Carnegie Mellon University.
- Model checking of CTL formulas.
- The first powerful model checker.
- Homepage:
  http://nusmv.irst.itnict.it.

---

**Environment Setup for NuSMV**

- Add the following line to the file called
  .profile in your home directory:
    ```
    export CS424=y
    ```
- You may need to login again for this change to take effect.
- Now you can use the NuSMV command.

---

**SMV Input Language**

- Textual input language.
- Create files with text editor and run SMV to check them.
- Preferred file extension: .smv.

---

**Running NuSMV**

- Run NuSMV as
  ```
  NuSMV <filename>.smv
  ```
- NuSMV checks each SPEC section in the input file, and prints:
  ```
  either
 specification ... is true
  or a counterexample
  ```
More Types of Variables

<table>
<thead>
<tr>
<th>VAR</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>0..15;</td>
</tr>
<tr>
<td>level</td>
<td>(empty, partial, full);</td>
</tr>
<tr>
<td>request</td>
<td>array 0..3 of boolean;</td>
</tr>
</tbody>
</table>

**Note:** All types are finite.

Underground Management System

**Goal:** Move trains safely from A to C

Operators in NuSMV

<table>
<thead>
<tr>
<th>Operators</th>
<th>Priority</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>* / mod</td>
<td>High</td>
<td>Arithmetic</td>
</tr>
<tr>
<td>+ -</td>
<td>High</td>
<td>Comparisons</td>
</tr>
<tr>
<td>&lt; &gt; &lt;= =</td>
<td>Low</td>
<td>Logical</td>
</tr>
<tr>
<td>!</td>
<td>Low</td>
<td>Logical</td>
</tr>
<tr>
<td>&amp;</td>
<td>Low</td>
<td>Logical</td>
</tr>
<tr>
<td>xor</td>
<td>Low</td>
<td>Logical</td>
</tr>
<tr>
<td>&lt;-&gt;</td>
<td>Low</td>
<td>Logical</td>
</tr>
</tbody>
</table>

A Simple Module

```
MODULE switch(toggle)
VAR
state: (straight, curved);
ASSIGN
init(state) := straight;
next(state) := case
    !toggle: state;
    state = straight: curved;
    state = curved: straight;
esac;
```

Notes on ASSIGN Blocks

- For each variable, the initial and next state can be assigned at most once.
- Alternatively, you can assign the variable for the current state:

  \[
  \langle \text{var} \rangle := \langle \text{expr} \rangle;
  \]

- Cyclical assignments are not possible.

More ...

The NuSMV language is designed to
- describe Kripke-structures;
- define and check CTL formulas;
- support modules to model complex systems.
**A More Complex Module**

```plaintext
MODULE train(grant_access, grant_exit, switch_state)
VAR
  pos: {travelling, on_A, on_B, on_C};
ASSIGN
  init(pos) := travelling;
  next(pos) := case
    pos = on_A & !grant_access : on_A;
    pos = on_A & grant_access : on_B;
  esac;
```

**Another Way of Doing It**

```plaintext
MODULE train(grant_access, grant_exit, switch_state)
VAR
  pos: {travelling, on_A, on_B, on_C};
  slow: boolean;
ASSIGN
  init(pos) := travelling;
  next(pos) := case
    pos = on_A & !grant_access : on_A;
  esac;
JUSTICE
  !slow
```

**Putting Things Together**

```plaintext
MODULE uturn_section(grant_access, grant_exit, switch_togg)
VAR
  switch_ABC: switch(switch_togg);
  train_1: train(grant_access, grant_exit, switch_ABC.state);
  train_2: train(grant_access, grant_exit, switch_ABC.state);
DEFINE
  detect_A := train_1.pos = on_A | train_2.pos = on_A;
```

**Specifying CTL Properties**

```plaintext
MODULE uturn_section(
VAR ...
DEFINE ...
-- No collision on any track section:
SPEC AG !(train1.pos=on_A & train2.pos=on_A)
SPEC AG !(train1.pos=on_B & train2.pos=on_B)
SPEC AG !(train1.pos=on_C & train2.pos=on_C)
```

**Example Trace**

<table>
<thead>
<tr>
<th>grant_access</th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>grant_exit</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>running</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>pos</td>
<td>A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Module Structure**

- Module main must always be present.
- Several modules are possible.
- Sections VAR, ASSIGN, etc. may occur multiply and in any sequence of order.
Modelling Reactive Systems

**Controller**

**Plant**

**Actuators**

**Sensors**

### Rules

*Not enforced by NuSMV, but …*

- Do not assign to any variables declared outside of your controller.
- Do not read variables declared outside of your controller, unless they are sensors from the parameter list.
- Do not modify the given sets of sensors and actuators.
- Do not modify the plant.

### Reactive Systems in NuSMV

**MODULE main**

**VAR**

- `plant: uturn_section` (controller.grant_access, controller.grant_exit, controller.switch_toggle);
- `controller: ums` (plant.detect_A, plant.detect_B, plant.detect_C);

### Counterexamples

--- specification: AG (¬(train_1.pos = on_A & train_2.pos = on_B))
- In plant is false
- as demonstrated by the following execution sequence
- `State: 1.1 <-` plant.switch_BCC.state = straight
- `State: 1.2 <-` plant.train_1.pos = on_A controller.grant_exit = 1
- `State: 1.3 <-` plant.train_1.pos = on_B controller.switch_toggle = 1 controller.grant_exit = 1
- `State: 1.4 <-` plant.train_1.pos = on_A plant.train_2.pos = on_A

Only shows variables that have changed

### Modelling the Controller

**MODULE ums**

**VAR**

- `switch_toggle: boolean;`
- `grant_access: boolean;`
- `grant_exit: boolean;`

Parameters represent inputs from the plant

Local variables include all actuators

### Reading

Béard et. al.:
Chapter 12 – Symbolic Model Checking

**NuSMV Manual & Tutorial:**

http://nusmv.irst.itc.it/NuSMV/userman/
http://nusmv.irst.itc.it/NuSMV/tutorial/