


COMP424/524-06A Topics in Software Engineering

Part I – CTL Model Checking
8. Introduction to NuSMV
Robi Malik



DEPARTMENT OF COMPUTER SCIENCE
TARI ROROHIKO

NuSMV as a Tautology Checker

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

```

MODULE main
VAR
  a: boolean;
  b: boolean;
  c: boolean;
SPEC
  ((a & b) -> c) <-> (a -> (b -> c))
  
```

Comments (pointing to the MODULE line)

Declaration of main module (pointing to the MODULE line)

Declaration of variables (pointing to the VAR line)


A formula to be checked (pointing to the SPEC line)

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 4

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.iirst.itc.it>.



Download possible (in a speech bubble pointing to the homepage link)

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 2

Environment Setup for NuSMV

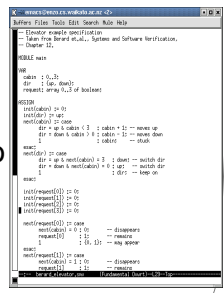
- Add the following line to the file called `.profile` in your home directory:


```
export CS424=y
```
- You may need to login again for this change to take effect.
- Now you can use the NuSMV command.

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 5

SMV Input Language

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



© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 3

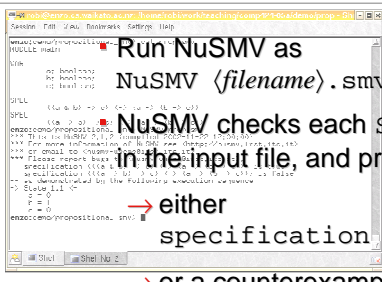
Running NuSMV

Run NuSMV as
`NuSMV <filename>.smv`

NuSMV checks each SPEC section in the input file, and prints:

→ either **specification ... is true**

→ or a counterexample



© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 6

More Types of Variables

Integer range

Enumeration

Array

```

VAR
  count : 0..15;
  level : {empty, partial, full};
  request: array 0..3 of boolean;
  
```

Note: All types are finite.

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 7

Underground Management System

Goal: Move trains safely from A to C

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 10

Operators in NuSMV

High Priority	* / + - mod	Arithmetic
	= != < > <= >=	Comparisons
Low	! & xor <-> ->	Logical

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 8

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;
  
```

Parameters

Initial state

State transition relation

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 11

More ...

The NuSMV language is designed to

- describe **Kripke-structures**;
- define and check **CTL formulas**;
- support **modules** to model complex systems.

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 9

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:


```
⟨var⟩ := ⟨expr⟩;
```
- Cyclical assignments are not possible.

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 12

A More Complex Module

```

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;
    ...
  1: {travelling, on_A, on_B, on_C};
esac;

```

Nondeterministic state change

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 13

Putting Things Together

```

MODULE uturn_section(grant_access, grant_exit,
                    switch_toggle)
VAR
  switch_ABC: switch(switch_toggle);
  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;
...

```

Instantiate modules

Access variables in modules

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 16

Another Way of Doing It

```

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;

```

Not assigning to this variable — can change at will

But assume fairness — must be true sometimes

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 14

Specifying CTL Properties

```

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)

```

- Properties can be added to any module.
- They will be checked for each instance.

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 17

Example Trace

grant_access	0	1	0	0	1	1	1
grant_exit	0	0	1	0	0	0	1
running	1	0	0	0	0	1	1
pos	A						

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 15

Module Structure

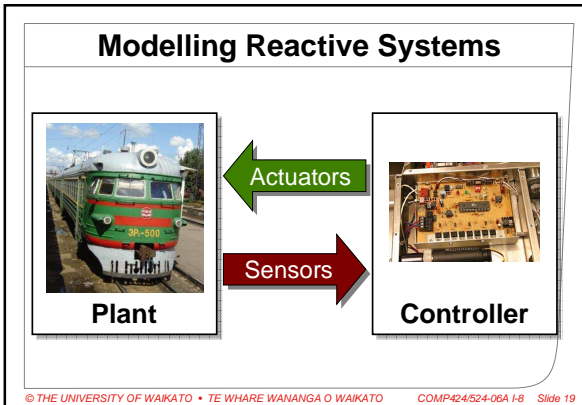
- Module **main** must always be present.
- Several modules are possible.
- Sections **VAR**, **ASSIGN**, etc. may occur multiply and in any sequence of order.

```

MODULE <name>
VAR
  ...
DEFINE
  ...
ASSIGN
  ...
SPEC
  ...

```

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 18



- ### 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.
- © THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 22

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);

```

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 20

Counterexamples

```

-- specification AG (!(train_1.pos = on_A & train_2.pos = on_A))
  IN plant is false
-- as demonstrated by the following execution sequence
-> State: 1.1 <-
...
  plant.switch_ABC.state = straight
-> State: 1.2 <-
  plant.train_1.pos = on_A
  controller.grant_access = 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

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 23

Modelling the Controller

```

MODULE ums
  (detect_A, detect_B, detect_C)
VAR
  switch_toggle: boolean;
  grant_access:  boolean;
  grant_exit:    boolean;

```

Parameters represent inputs from the plant

Local variables include all actuators

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 21

Reading

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

NuSMV Manual & Tutorial:

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

© THE UNIVERSITY OF WAIKATO • TE WHARE WANANGA O WAIKATO COMP424/524-06A I-8 Slide 24