

Project with NuSMV

Instructions:

- 1) Your project report should be written in MS word.
 - 2) Your project report should contain all the codes of NuSMV: model, state diagram, properties, and the package running windows.
 - 3) Turn in your report to the D2L by the deadline.
 - 4) Each group should submit one report.
 - 5) Write your email address on the first page of your report.
-

Problem 1.

Follow the instructions and prepare yourself for using NuSMV.

➤ nuSMV 2.5.4 Download:

<http://nusmv.fbk.eu/NuSMV/download/getting-v2.html>

➤ Tutorial:

<http://nusmv.fbk.eu/NuSMV/tutorial/>

➤ Install the binary

Run NuSMV from the command line (NOTICE the -int option):

```
./NuSMV -int
NuSMV > help
NuSMV > read_model -h
usage: read_model [-h] [-i <file>]
-h Prints the command usage.
-i <file> Reads the model from the specified <file>.
NuSMV > quit
```

Problem 2.

Consider the following example.

```
MODULE main
VAR
    request : boolean;
    state : {ready,busy};
ASSIGN
    init(state) := ready;
    next(state) := case
        state = ready & request : busy;
        TRUE : {ready,busy};
    esac;
SPEC
    AG(request -> AF state = busy)
```

Copy the code above in a text file and call it short.smv. Then, call NuSMV and load this file:

```
NuSMV -int
NuSMV > read_model -i short.smv
NuSMV > flatten_hierarchy
```

```
NuSMV > encode_variables
NuSMV > build_model
```

At this point NuSMV is ready to perform simulation and verification. You can verify the formula above with the command `check_ctl`. You can verify more CTL formulas with the command `check_ctl -p "CTL-formula"`. For instance, check whether or not the formula `AG(request -> AX(state=busy))` is true.

Tasks:

- 1) Read the tutorial of NUSMV 2.5 and learn the syntax, draw the state diagram of above code.
- 2) Come up with 3 more CTL properties and check against the model.

Problem 3.

Consider the following 3-bit counter.

```
MODULE counter_cell(carry_in)
VAR
    value : boolean;
ASSIGN
    init(value) := FALSE;
    next(value) := value xor carry_in;
DEFINE
    carry_out := value & carry_in;
MODULE main
VAR
    bit0 : counter_cell(TRUE);
    bit1 : counter_cell(bit0.carry_out);
    bit2 : counter_cell(bit1.carry_out);
```

Without using NuSMV, try to understand what the code does (see tutorial).

Copy the code above in a text file and call it `counter.smv`. Then, call NuSMV and load this file:

```
./NuSMV -int
NuSMV > read_model -i counter.smv
NuSMV > flatten_hierarchy
NuSMV > encode_variables
NuSMV > build_model
```

At this point NuSMV is ready to perform simulation and verification.

Simulation:

At this point you verify that your understanding of the code was correct. To perform simulation, first you have to select a state from the set of initial states:

```
NuSMV > pick_state -i
```

(with this command you select a state interactively (-i). As there is only one initial state, you just have to confirm the only option available). The command `simulate -i <number>` is used to perform simulation. <number> specifies the number of steps you want to simulate,

e.g.,

```
NuSMV > simulate -i 1
```

The evolution of the system is deterministic. Try a few steps and check that your guess was

correct.

Verification:

You can verify CTL formulas with the command `check_spec -p "CTL-formula"`. For instance, check whether or not the formula `AX(bit0.value=0)` is true.

Tasks:

- 1) Draw state diagram for this model
- 2) Come up with 3 more CTL property and check against above model.

Problem 4. Model checking VS Simulation

Consider the following microwave oven controller design written in Verilog.

```
// Microwave oven
module microwave(clk, sys_reset, reset, closeDoor, starOven, done, States);
    input  clk, sys_reset;
    input  reset, closeDoor, starOven, done;
    reg    Start, Close, Heat, Error;
    output reg [3:0] States;

    always @ (posedge clk)
    begin
        if (sys_reset == 1'b1)
            begin
                {Start, Close, Heat, Error} <= 4'b0000;
            end
        else
            begin
                case ({Start, Close, Heat, Error})
                    4'b0000:
                        if (closeDoor) {Start, Close, Heat, Error} <= 4'b0100;
                        else if (starOven) {Start, Close, Heat, Error} <= 4'b1001;
                    4'b1001:
                        if (closeDoor) {Start, Close, Heat, Error} <= 4'b1101;
                    4'b1101:
                        if (~closeDoor){Start, Close, Heat, Error} <= 4'b1001;
                        else if (reset) {Start, Close, Heat, Error} <= 4'b0100;
                    4'b0100:
                        if (~closeDoor){Start, Close, Heat, Error} <= 4'b0000;
                        else if (starOven) {Start, Close, Heat, Error} <= 4'b1100;
                    4'b1100:
                        {Start, Close, Heat, Error} <= 4'b1110;
                    4'b1110:
                        {Start, Close, Heat, Error} <= 4'b0110;
                    4'b0110:
                        if (~closeDoor){Start, Close, Heat, Error} <= 4'b0000;
                        else if (done) {Start, Close, Heat, Error} <= 4'b0100;
                    default:
                        {Start, Close, Heat, Error} <= 4'b0000;
                endcase
            end
        end
    end

    always @ (*)
```

```

begin
    States = {Start, Close, Heat, Error};
end
endmodule // microwave

```

Tasks:

- 1) Understand the design and draw the transition diagram for the design
- 2) Propose 5 new CTL properties. Describe them in English first, then in CTL forms. Your properties should be different from those proposed by other students.
- 3) Create Verilog test benches and run simulation to verify your properties, justify your test result. (show the test result in script form instead of wave form)
- 4) Model the design in NuSMV
- 5) Prove/disprove them by the NuSMV package.
- 6) Explain why the property is true or false and if the test result meets your expectation.
- 7) Compare the model checking to simulation, and discuss the advantages and disadvantages of the above two verification methods.

Sample properties:

- ✧ If an error condition occurs, it must be reset before the oven may heat.
- ✧ $G((\text{Close}=0 * \text{Start}=1) \rightarrow (\text{Error}=0 \text{ R } \text{Heat}=0));$
- ✧ The oven can't be heated if the door is not closed.
- ✧ $\text{Heat}=0 \text{ U } \text{Close}=1;$