Exercise 1: Modeling

Vis

Call vis and model check the examples from the lecture.
See the document “How to get started with VIS on ensisun” on how to get started.
The examples and this document can be downloaded from:

`http://www-verimag.imag.fr/~jobstman/teaching/`

Mutex

Draw the Kripke structure defined by the following Verimag model. Give two CTL properties the model satisfies and two it does not satisfy.

```
typedef enum{s0,s1,s2} stateT;

module mutex(clk,state,y,x);
    input clk;
    input x;
    output state;
    output y;
    wire clk;
    wire x;
    reg y;
    stateT reg state;

    initial begin
        state=s0;
        y=0;
    end

    always @(posedge clk) begin
        case(state)
        s0: begin state = s1; y = x+1; end
        s1: if (x == 0 || y == 0) state = s2;
        s2: begin state = s0; y = 0; end
        endcase // case (state)
    end // always @ (posedge clk)

endmodule // mutex
```

Sabbath Elevator

Consider the following Verilog code of a simple controller for an elevator with four floors (0,1,2,3). The elevator start in floor 0 (`initial position = 0`) moves one by one up to floor 3 and then moves floor by floor down to floor 0 again. This pattern repeat. Currently, the controller does not react to the four input signals indicating that a floor was requested.

```
module elevator(position,clk,button0,button1,button2,button3);
       input clk;
       input button0, button1, button2, button3;
       output position;
       reg [1:0] position;
       reg up;

       initial begin
          position = 0;
          up = 1;
       end

       always @(posedge clk) begin
           if (up)
               if (position < 3) position = position + 1;
               else up = 0;
           else  //down
                 if (position > 0) position = position - 1;
                 else up = 1;
       end
endmodule
```

1) Write CTL formulas that show that the elevator can always reach any for the four floors. (See `/home/perms/jobstmab/Vis/doc/ctl.pdf` for the CTL syntax in VIS. E.g., `position[1:0]=2` or `position<1>=1 * position<0>=0` refer to the elevator being in floor 2.)
2) Modify the controller (meaning the Verilog code) to react to the requests sent my users of the elevators with the four input signals button0, button1, button2, button3.
3) Write a CTL property showing that the elevator does not move if none of the buttons is pressed and check if your controller satisfies this property.
4) Write a CTL formula describing that floor 2 is eventually reached if button2 was pressed and check if your controller satisfies this property.