

# SMV model checking tool

정세진  
김재엽

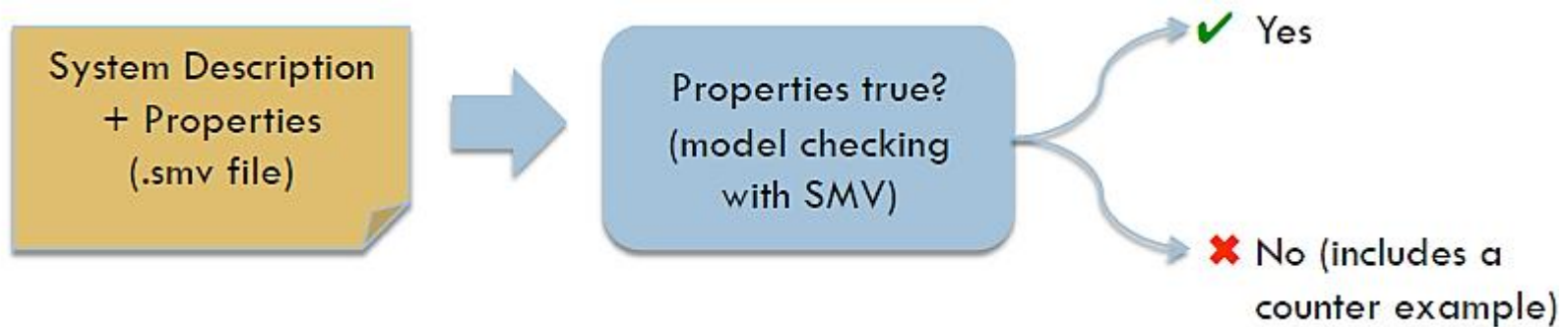
# SMV

---

- SMV is one of the Model checking tool
- Input
  - .smv
  - .v
- Property
  - Temporal logic
- Verifying fail => show counter example

# SMV

- SMV : Symbolic Model Verifier



- Cadence SMV
  - Based on Ken McMillan's work at CMU
  - Can do compositional verification
  - more expressive mode description language ( .smv (2 types) , .v (Verilog) )
  - easy-to-use graphical user interface

# SMV

---

- Model을 code로 작성
- State machine 등의 모델링이 가능
- 모델링 후 확인하고 싶은 property 작성을 통해 verify

# SMV - language

```
module main(req1,req2,ack1,ack2)
{
```

```
  input req1,req2 : boolean;
  output ack1,ack2 : boolean;
```

Type declarations

```
  ack1 := req1; /* priority */
  ack2 := req2 & ~req1;
```

Signal assignments

```
  mutex : assert G ~(ack1 & ack2);
  serve : assert G ((req1 | req2) -> (ack1 | ack2));
  waste1 : assert G (ack1 -> req1);
  waste2 : assert G (ack2 -> req2);
  no_starve : assert G F (~req2 | ack2);
```

Assertions

Prove하고 싶은  
property를 작성

# SMV - language

---

- Data type
  - <signal> : <type>;
  - X : boolean;
  - State : {ready,willing,able};
  - count : 0..7;
  - zip : array 2..0 of boolean;
    - zip[2] : boolean; zip[1] : boolean; zip[0] : boolean;
  - matrix[0] : array [2..0] of boolean;
  - matrix[1] : array [2..0] of boolean;

# SMV – language

---

- Assignment

- $X := y;$

- Delay assignment

- $\text{init}(x) := 0;$

- $\text{next}(x) := y;$

- State machines

```
start, done : boolean;  
state : {idle,cyc1,cyc2};
```

```
default done := 0;
```

```
in switch(state){
```

```
  idle:
```

```
    if start then next(state) := cyc1;
```

```
  cyc1:
```

```
    next(state) := cyc2;
```

```
  cyc2:
```

```
    next(state) := cyc2;
```

```
  done := 1;
```

```
}
```

# SMV – language

---

- `if(<condition>)`
  - `<stmt1>`
- `else`
  - `<stmt2>`
- `default`
  - `<stmt1>`
- `in`
  - `<stmt2>`
- `switch(<expr>)`
  - {
    - `<case1> : <stmt1>`
    - `<case2> : <stmt2>`
    - ...
    - `<casen> : <stmtn>`
    - [`default : <dftlstmt>`]}



# SMV - language

---

- ```
for(i = 0; i < 3; i = i + 1) {  
    x[i] := i;  
}
```
- ```
chain(i = 0; i < n; i = i + 1)  
    if (inp[i]) out := i;
```

# SMV - language

---

::	(concatenation)
-	(unary minus sign)
+, /, <<, >>	(mult, div, left shift, right shift)
+, -	(add, subtract)
mod	(integer mod)
in	(set inclusion)
union	(set union)
=, ~=, <, <=, >, >=	(comparison operators)
~	(not)
&	(and)
, ^	(or, exclusive or)
<->	(iff)
->	(implies)
,	(tuple separator)
?:	(conditional)
..	(integer subrange)

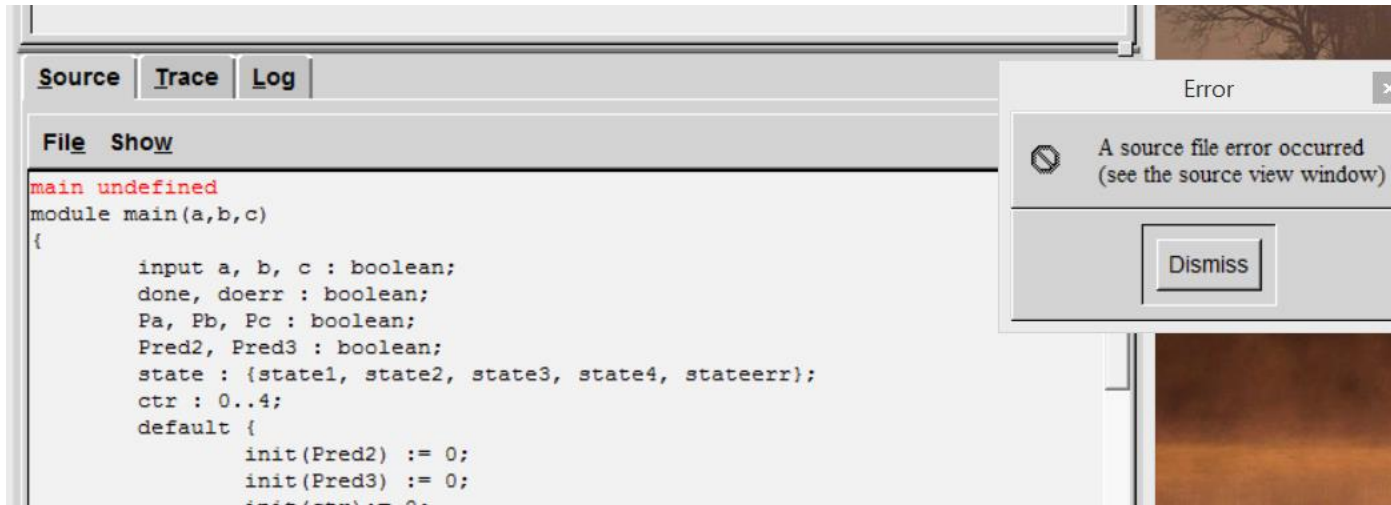
# SMV - language

---

- Assertion
  - foo : assert p;
  - foo : assert AG (( $\sim$ p | q) -> X(p = 1));

# SMV - manual

- 실행
  - 바탕화면에 생성된 바로가기를 통해 실행한 경우 문제



# SMV - manual

- 실행
  - 시작 메뉴의 파일을 통해 실행



# SMV - manual

- Cone

The screenshot shows the SMV software interface. The main window displays the 'Cone' view for a file named 'temp1.smv'. The interface includes a menu bar (File, Prop, View, Goto, History, Abstraction, Help) and a toolbar (Browser, Properties, Results, Cone, Using, Groups). The main window is divided into two sections: a table of signals and a source code view.

The 'Entire cone' section shows a table with the following data:

Signal	Layer	Vars	Type
a	free	1	comb
b	free	1	comb
c	free	1	comb
ctr	.	3	state
done	.		
state	.	3	state

The 'Source' section shows a table with the following data:

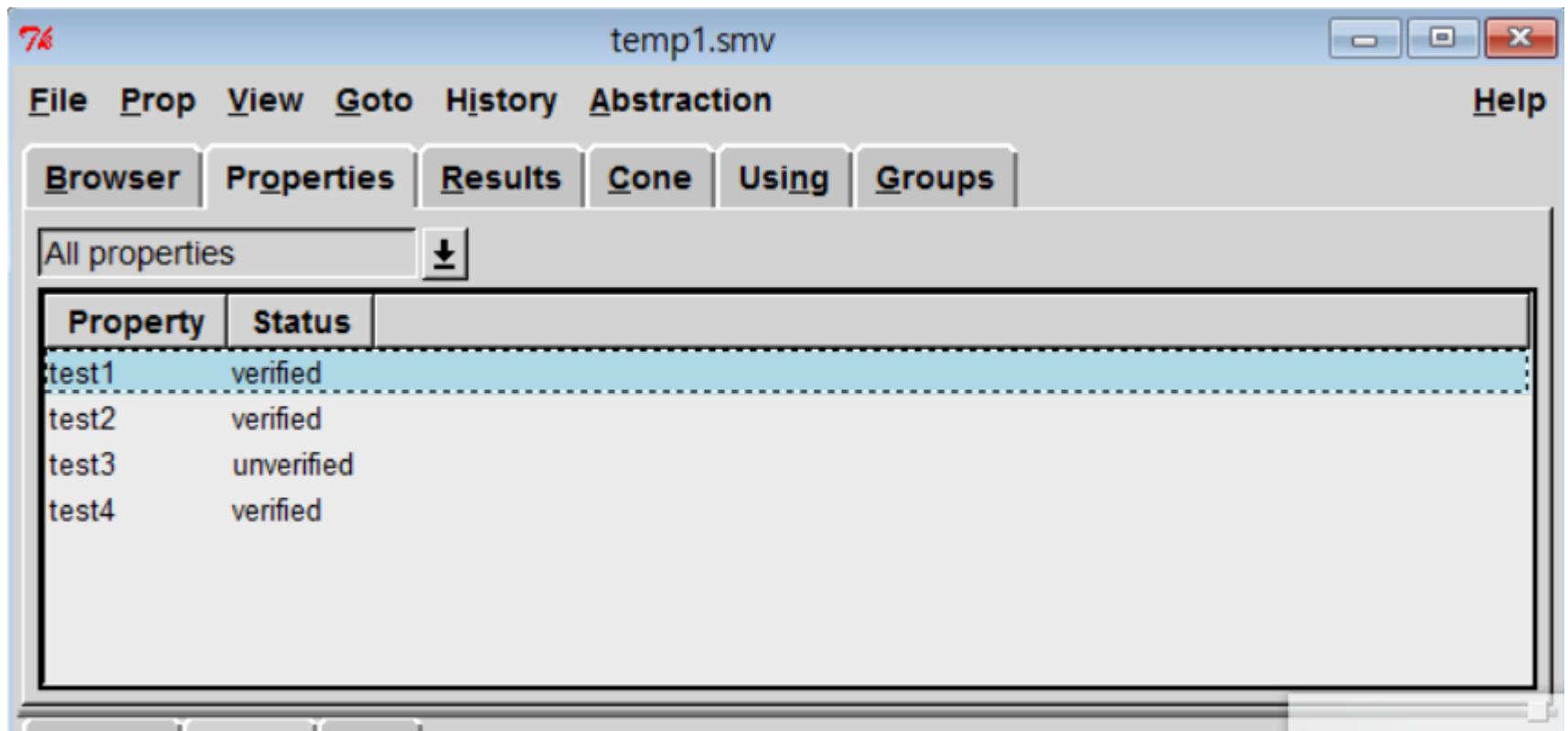
	1	2	3						
a	1	0	1						
b	0	1	0						
c	0	0							
ctr	0	0	0						
done	-	-	-						
state	state1	state2	state3						

The status bar at the bottom shows 'Property: test3' and 'i-search:'.



# SMV - manual

- Properties





# SMV - manual

- Suggest value

The screenshot shows the SMV GUI with the 'Results' tab selected. The 'All results' table contains one entry: test4, false, Thu Sep 25 19:25:20. Below the table, the 'Source' tab is active, showing a grid with columns 'state1', 'state2', and 'state3'. A context menu is open over the grid, with 'Suggest value' highlighted in a red box. Other menu items include 'Override value', 'Insert row', 'Delete row', and 'Delete all'. The status bar at the bottom shows 'Property: test4' and 'i-search:'.

The screenshot shows the SMV GUI with the 'Source' tab selected. The 'All results' table is the same as in the previous screenshot. The 'Source' tab shows a grid with columns '1', '2', and '3'. The grid contains the following values:

	1	2	3
a	1	0	1
b	0	1	0
c	0	0	0
ctr	0	0	0
state	state1	state2	state3

The status bar at the bottom shows 'Property: test4' and 'i-search:'.

# SMV - manual

- Extend trace

The screenshot shows the SMV GUI with the 'Trace' tab selected. A dialog box titled 'Trace extension' is open, asking 'Extend trace by how many steps?' with the input '3'. The dialog has 'OK' and 'Cancel' buttons. In the background, the 'Trace' table has a red box around the 'Extend trace' button.

Property	Result	Time
test1	true	Thu Sep 25 19:16:54 2014
test4	false	Thu Sep 25 19:16:54 2014

1	2	3	4	5	6
a	1	0	0		
b	0	1	0		
c	0	0	0		
ctr	0	0	0		
state	state1	state2	state3		

The screenshot shows the SMV GUI with the 'Trace' tab selected. The 'Trace' table has a blue dashed box highlighting the cell at row 'a', column '4'. The 'Trace extension' dialog box is no longer visible.

Property	Result	Time
test1	true	Thu Sep 25 19:16:54 2014
test4	false	Thu Sep 25 19:16:54 2014

1	2	3	4	5	6
a	1	0	1		
b	0	1	0		
c	0	0	0		
ctr	0	0	0		
state	state1	state2	state3		

# SMV - manual

- Recalculate trace

The screenshot shows the SMV GUI with the 'Trace' tab selected. A red box highlights the 'Recalculate trace' button. The 'Results' tab shows a table with the following data:

Property	Result	Time
test1	true	Thu Sep 25 19:16:54 2014
test4	false	Thu Sep 25 19:16:54 2014

The 'Trace' tab shows a table with the following data:

	1	2	3	4	5	6			
a	1								
b	0	1	0						
c	0	0							
ctr	0	0	0						
state	state1	state2	state3						

The screenshot shows the SMV GUI with the 'Trace' tab selected. The 'Results' tab shows the same data as the previous screenshot. The 'Trace' tab shows a table with the following data:

	1	2	3	4	5	6			
a	1	0	1	0	0	0			
b	0	1	0	0	0	0			
c	0	0	0	0	0	0			
ctr	0	0	0	0	0	0			
state	state1	state2	state3	state4	state4	state4			

# SMV – example 1

```
module main(inc, dec)
{
    input inc, dec : boolean;
    state : {state0, state1, state2};

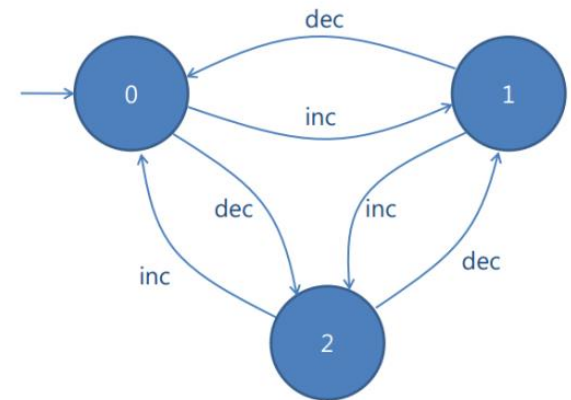
    default ;
    in switch(state){
        state0:
            if(inc) next(state) := state1;
            else if(dec) next(state) := state2;

        state1:
            if(inc) next(state) := state2;
            else if(dec) next(state) := state0;

        state2:
            if(inc) next(state) := state0;
            else if(dec) next(state) := state1;

    };

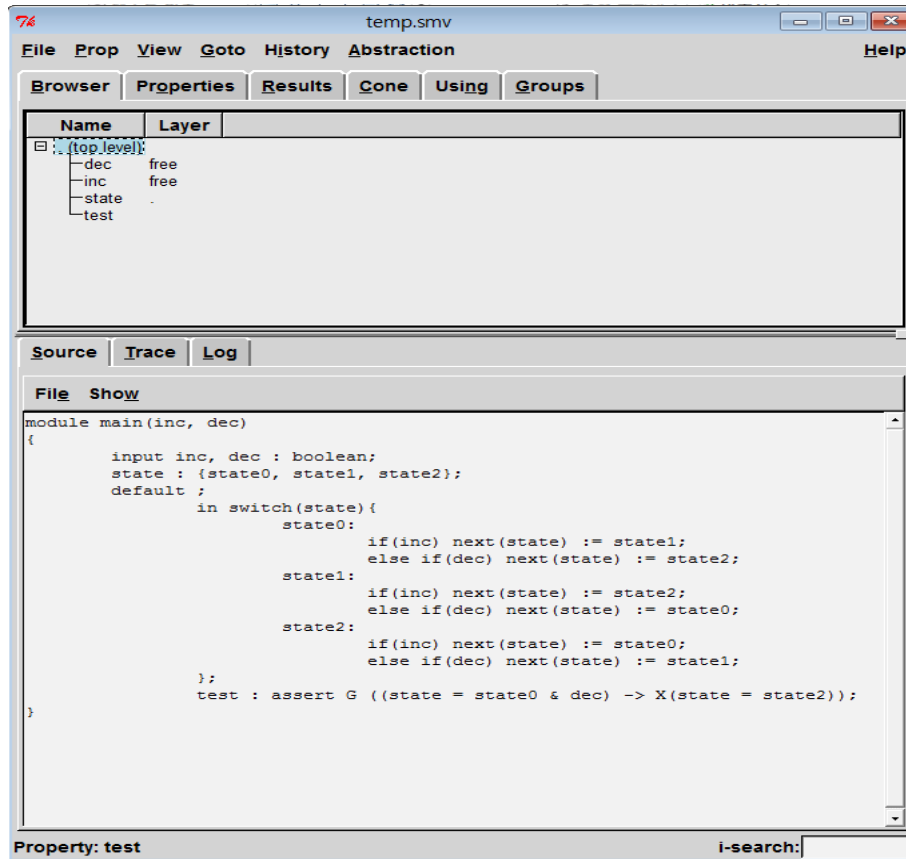
    test : assert G ((state = state0 & dec) -> X(state = state2));
}
```



$A_{c3}$  : a module 3 counter

# SMV – example 1

- 파일 작성 후 오픈



The screenshot shows the SMV editor interface. The top menu bar includes File, Prop, View, Goto, History, Abstraction, and Help. Below the menu is a toolbar with buttons for Browser, Properties, Results, Cone, Using, and Groups. The main window is divided into two panes. The upper pane shows a project tree with a table of components:

Name	Layer
(top_level)	
dec	free
inc	free
state	
test	

The lower pane shows the source code for the module main(inc, dec):

```
module main(inc, dec)
{
    input inc, dec : boolean;
    state : {state0, state1, state2};
    default ;
    in switch(state) {
        state0:
            if(inc) next(state) := state1;
            else if(dec) next(state) := state2;
        state1:
            if(inc) next(state) := state2;
            else if(dec) next(state) := state0;
        state2:
            if(inc) next(state) := state0;
            else if(dec) next(state) := state1;
    };
    test : assert G ((state = state0 & dec) -> X(state = state2));
}
```

At the bottom of the editor, the 'Property: test' is displayed on the left and the 'i-search:' field is on the right.

# SMV – example 1

- verify

The screenshot shows a model checker interface. At the top, a project tree displays a hierarchy under '[top level]':

Name	Layer
[top level]	
-dec	free
-inc	free
-state	.
-test	.

Below the tree are tabs for 'Source', 'Trace', and 'Log'. The 'File' tab is active, showing a command window with the following output:

```
iteration 0.....5
[optimized conjunctive relation
Optimized transition relation s
user time.....0 s
system time.....0.046875 s
Model checking time: 0.000000
user time.....0 s
system time.....0.046875 s

Model checking results
=====
test.....false

user time.....0 s
system time.....0.046875 s

Resources used
=====
user time.....0 s
system time.....0.046875 s
BDD nodes allocated.....96
data segment size.....0
```

A 'Confirm' dialog box is overlaid on the command window, containing an information icon, the text 'Verification finished', and a button labeled '확인' (Confirm).

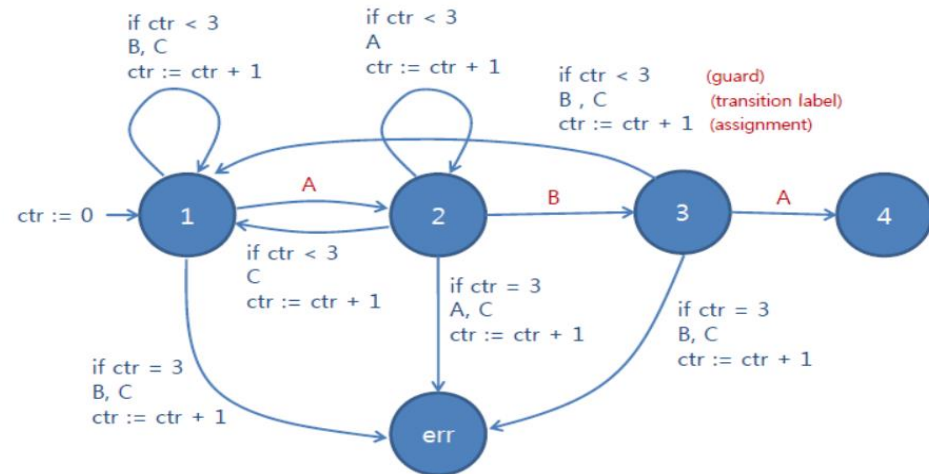
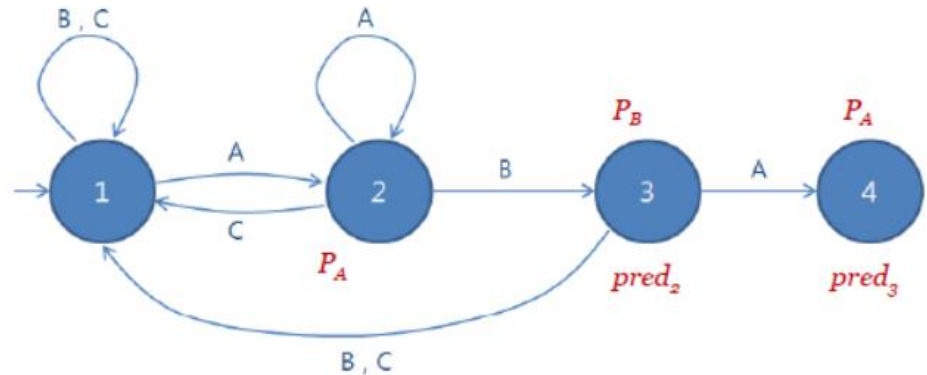


# SMV – example2

```

module main(a,b,c)
{
  input a, b, c : boolean;
  done, doerr : boolean;
  Pa, Pb, Pc : boolean;
  Pred2, Pred3 : boolean;
  state : {state1, state2, state3, state4, stateerr};
  ctr : 0..4;
  default {
    init(Pred2) := 0;
    init(Pred3) := 0;
    init(ctr) := 0;
    init(state) := state1;
    init(Pa) := 0;
    init(Pb) := 0;
    init(Pc) := 0;
  }
  in switch(state) {
    state1:
      if(a){
        next(Pa) := 1;
        next(Pb) := 0;
        next(state) := state2;
      } else if(ctr < 3 & (b | c)){
        next(state) := state1;
        next(ctr) := ctr + 1;
      } else if(ctr = 3 & (b | c)){
        next(state) := stateerr;
        next(ctr) := ctr + 1;
      }
  }
}

```





# SMV – example2

```
state2:
  if(b) {
    next(Pb) := 1;
    next(Pa) := 0;
    next(Pred3) := 0;
    next(Pred2) := 1;
    next(state) := state3;
  } else if(ctr < 3 & a) {
    next(ctr) := ctr + 1;
  } else if(ctr < 3 & c) {
    next(state) := state1;
    next(ctr) := ctr + 1;
  } else if(ctr = 3 & (a | c)) {
    next(state) := stateerr;
    next(ctr) := ctr + 1;
  }
}
```

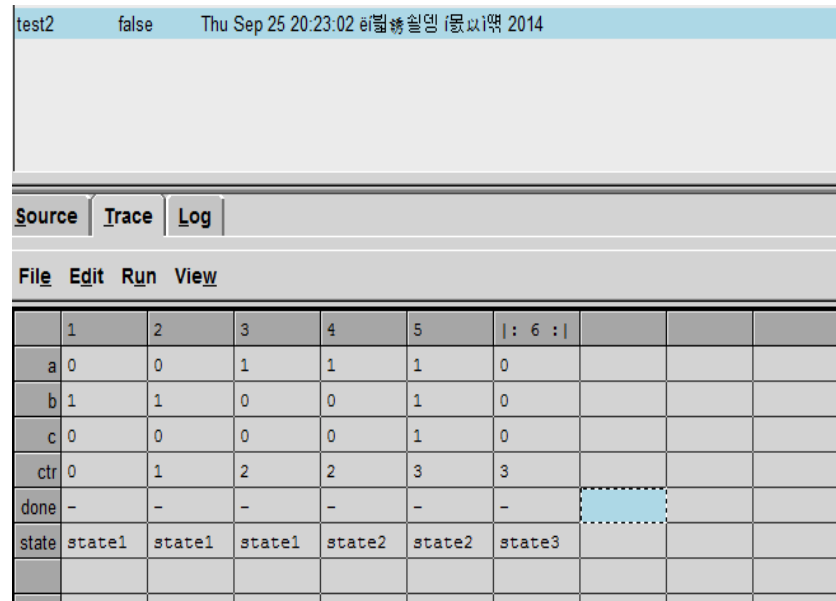
```
state3:
  if(a) {
    next(Pred3) := 1;
    next(Pa) := 1;
    next(Pb) := 0;
    next(state) := state4;
  } else if(ctr < 3 & (b | c)) {
    next(state) := state1;
    next(ctr) := ctr + 1;
  } else if(ctr = 3 & (b | c)) {
    next(state) := stateerr;
    next(ctr) := ctr + 1;
  }
}
state4: {
  done := 1;
}
stateerr: {
  doerr := 1;
}
};
```

```
test : SPEC AG(Pa = 1 & EX(Pb = 1 & EX(Pa = 1)) -> EF(done = 1));
test1 : SPEC AF((done = 1) -> EF(Pred2 = 1 & Pred3 = 1));
test2 : SPEC AG((ctr = 3 & state = state2) -> AF(done = 1) );
```



# SMV - problem

- 예제로 작성한 모델의 문제점
  - 모델을 모든 상황에 대해 검사 하므로 동시 입력의 문제
  - 아무 입력이 없는 상황에 대한 문제



The screenshot shows a model checker interface with a table of variables and their values. The table has columns labeled 1 through 5, and a column labeled |: 6 :|. The rows are labeled a, b, c, ctr, done, and state. The 'done' row has a dashed blue box around the cell containing '-'. The 'state' row has cells containing state1, state1, state1, state2, state2, and state3.

	1	2	3	4	5	: 6 :			
a	0	0	1	1	1	0			
b	1	1	0	0	1	0			
c	0	0	0	0	1	0			
ctr	0	1	2	2	3	3			
done	-	-	-	-	-	-			
state	state1	state1	state1	state2	state2	state3			

Q & A

**END**

