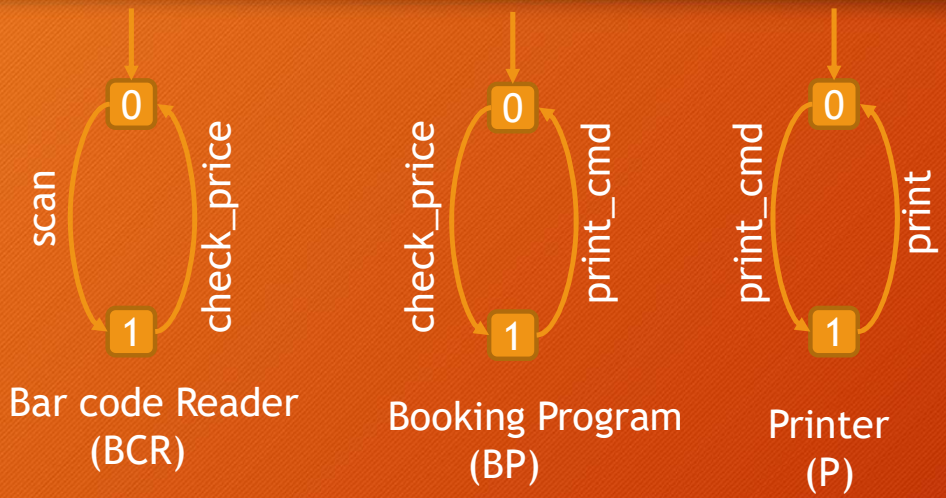
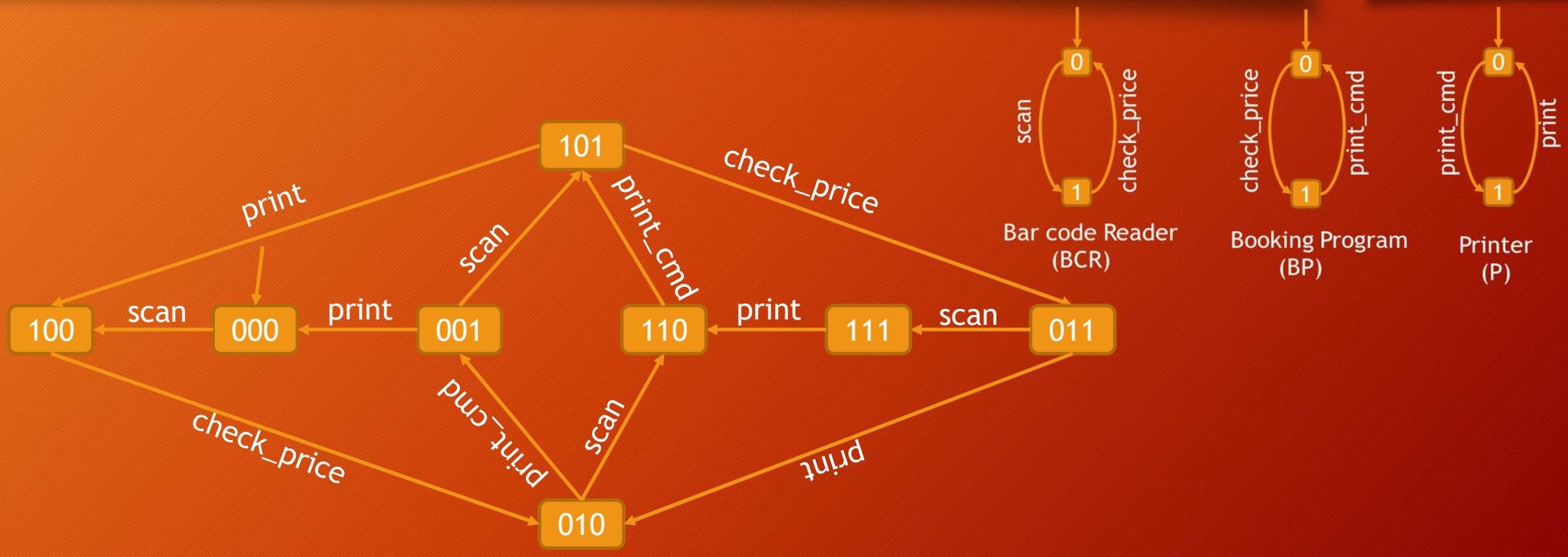
SE2003
Formal Methods in Software Engineering

Spring-2024

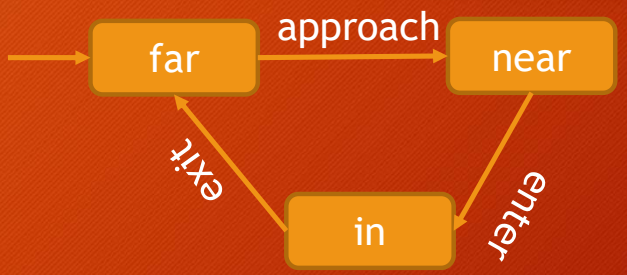
Modelling Concurrent Systems



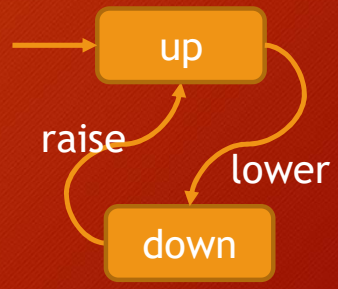
Modelling Concurrent Systems



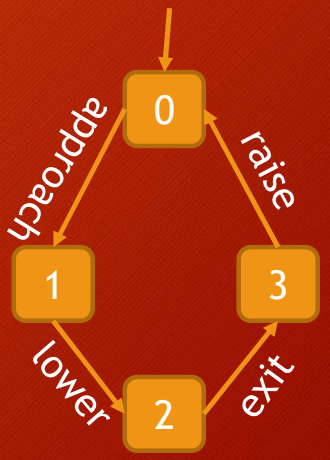
Modelling Concurrent Systems



TRAIN

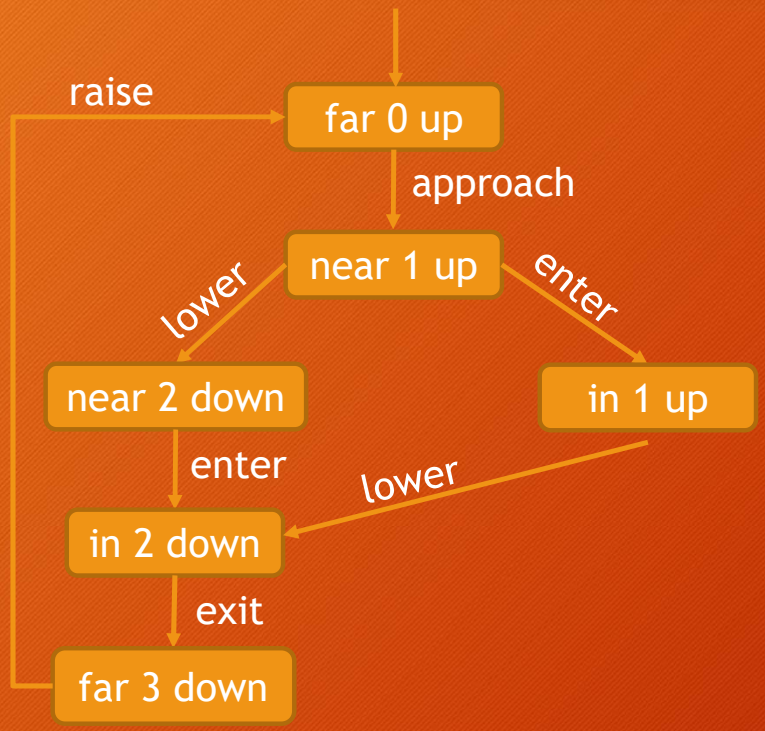


GATE

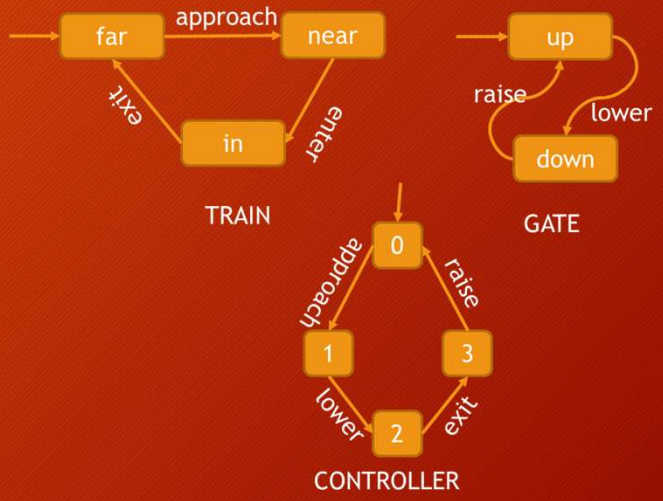


CONTROLLER

Modelling Concurrent Systems

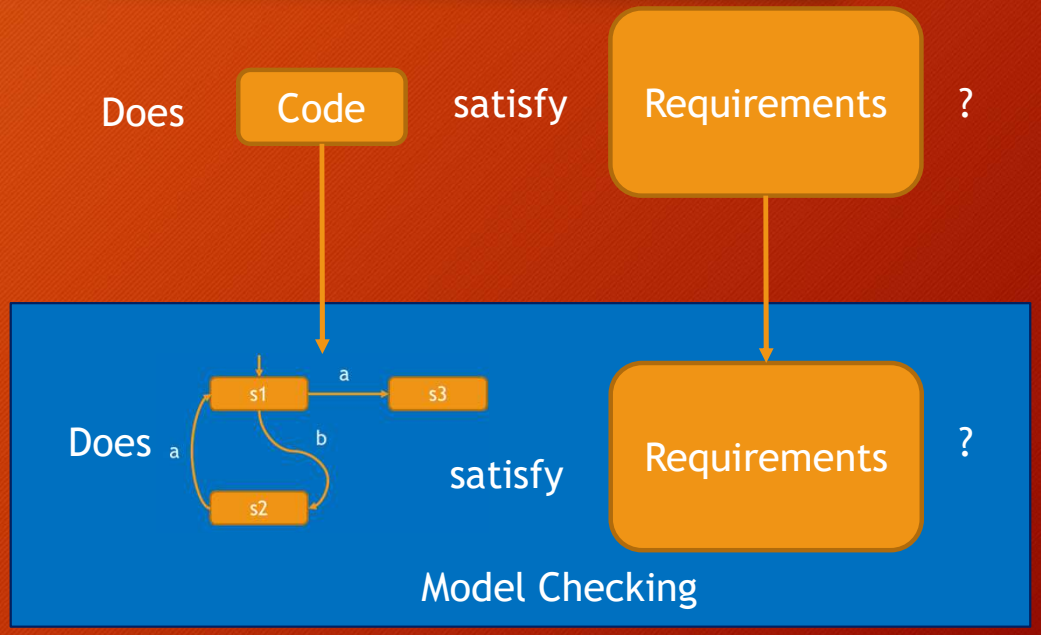


TRAIN || CONTROLLER || GATE



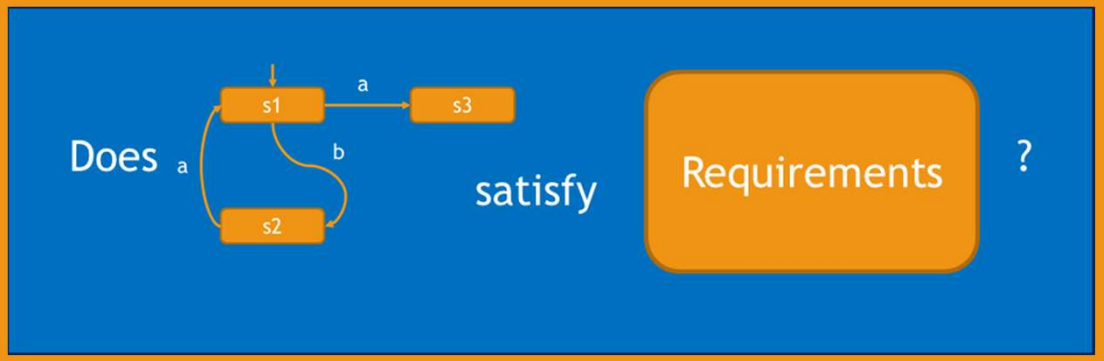
Model Checking

- How reliable is the software?
- Requirements
  - Value of x is always between 1 and 50
  - Value of x is always less than y
  - Whenever x is equal to 50 y should be greater than 200
  - Two programs never in the critical state
- Test cases
  - Size of code or
  - Number of concurrent processes are large



Model Checking Tools

- Model Checkers
- SPIN
  - Well suited for the concurrent systems
- NuSMV
  - Well suited for hardware circuits
  - More requirements can be checked compared to SPIN
  - Install
    - nusmv.fbk.eu

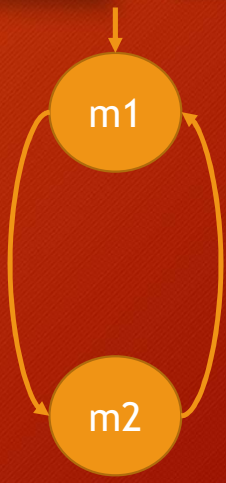


Format of the model checker

Model checker automatically solve the questions

Models in NuSMV

```
MODULE main
VAR
    location: {m1,m2};
ASSIGN
    init(location) := m1;
    next(location) := case
            (location = m1) : m2;
            (location = m2) : m1;
                    esac;
```

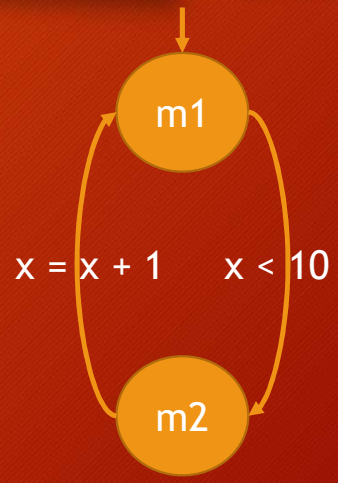



```
>nusmv -int
NuSMV>read_model -i intro-demo.smv
NuSMV>flatten_hierarchy
NuSMV>encode_variables
NuSMV>build_model
NuSMV>pick_state -i
NuSMV>simulate -i -k 10
NuSMV>print_reachable_states -v
```

Models in NuSMV

```
MODULE main
VAR
     location: {m1,m2};
     x : 0 .. 100;
ASSIGN
     init(location) := m1;
     init(x) := 5;
     next(location) := case
     (location = m1) & (x<10) : m2;
     (location = m2) : m1;
     TRUE : location;
               esac;
```

```
next(x):= case
     (location=m2) & (x<50) : x+1;
     TRUE:x;
     esac;
```

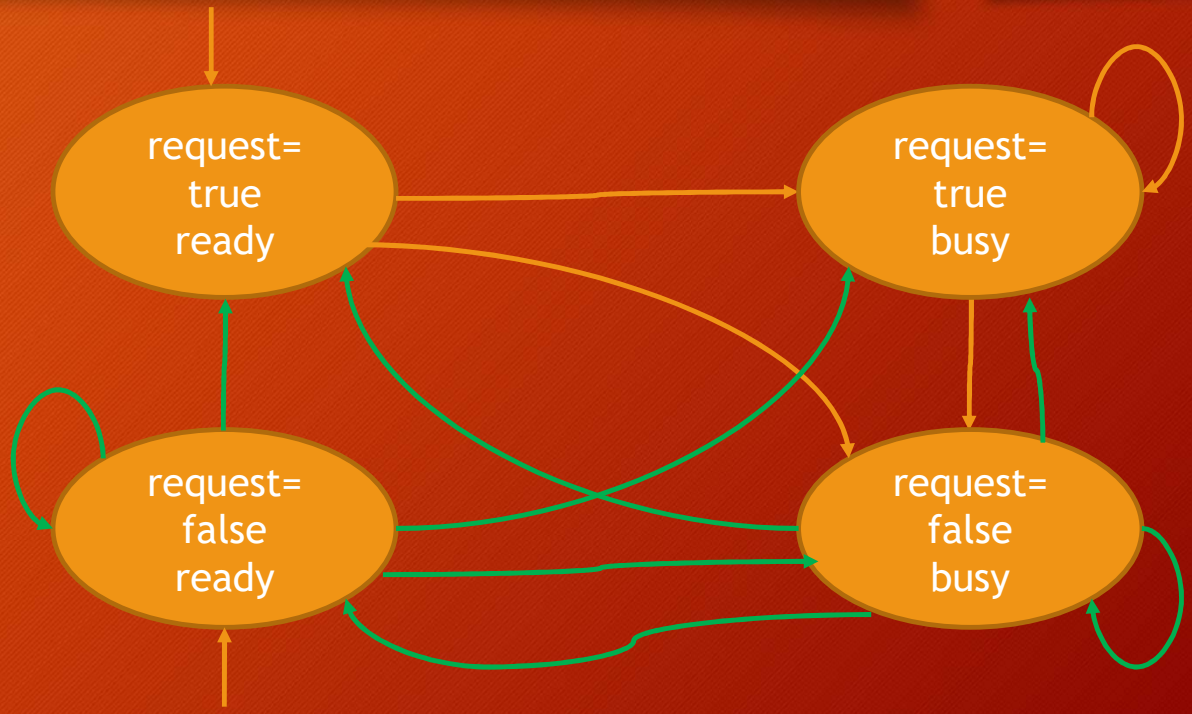



```
>nusmv -int
NuSMV>read_model -i pg-demo.smv
NuSMV>flatten_hierarchy
NuSMV>encode_variables
NuSMV>build_model
NuSMV>pick_state -i
NuSMV>simulate -i -k 10
NuSMV>print_reachable_states -v
```

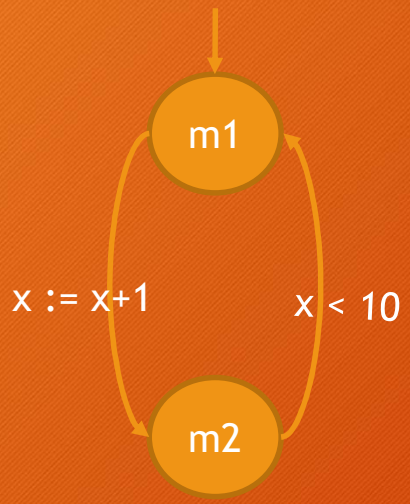
Models in NuSMV

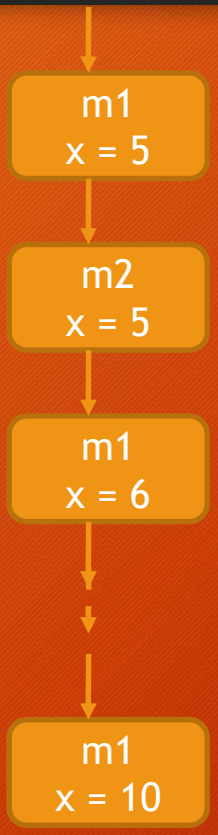
```
MODULE main
VAR
     request : boolean;
     status : {ready, busy};
ASSIGN
     init(status)  := ready;
     next(status) := case
          request = TRUE : busy;
          TRUE : {ready, busy};
     esac;
```

```
>nusmv -int
NuSMV>read_model -i request-busy-demo.smv
NuSMV>flatten_hierarchy
NuSMV>encode_variables
NuSMV>build_model
NuSMV>pick_state -i
NuSMV>simulate -i -k 10
NuSMV>print_reachable_states -v
```



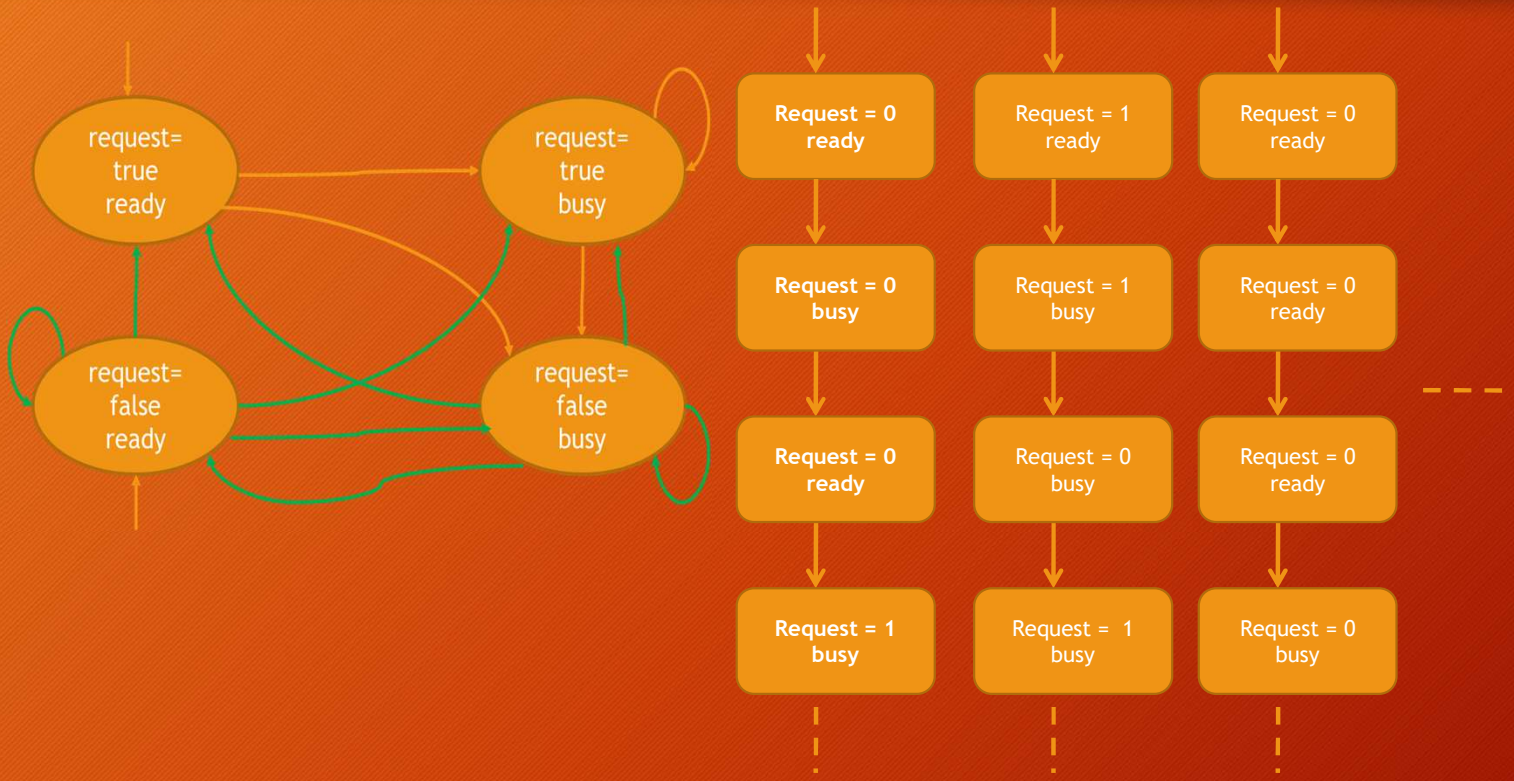
Models in NuSMV





Executions
Initial value of
x = 5

Models in NuSMV



Models in NuSMV

Transition system satisfies a requirement

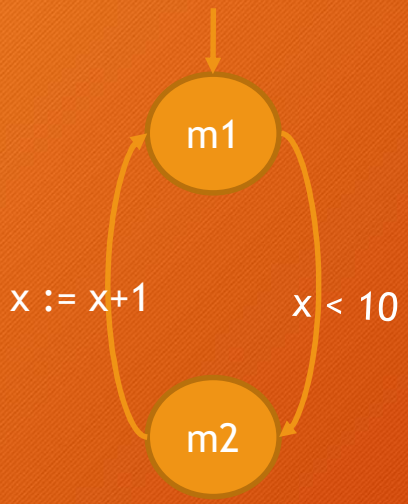
means

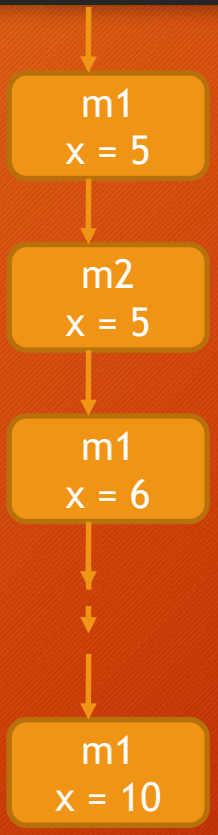
all its executions satisfy the requirement

Models in NuSMV

- Requirement type 1 : G

Models in NuSMV

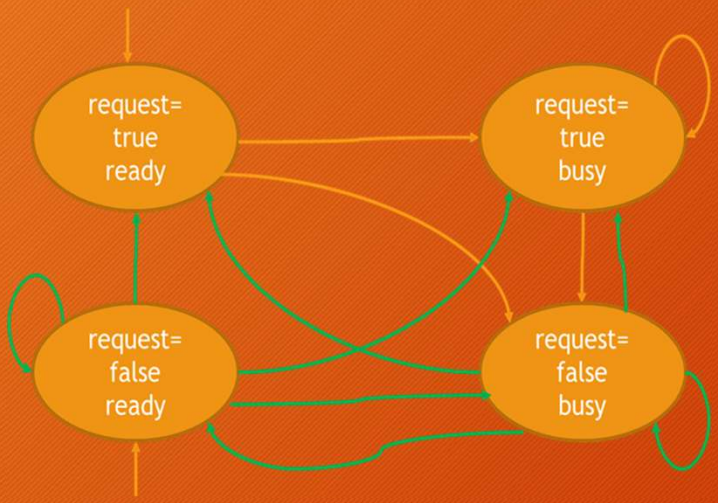


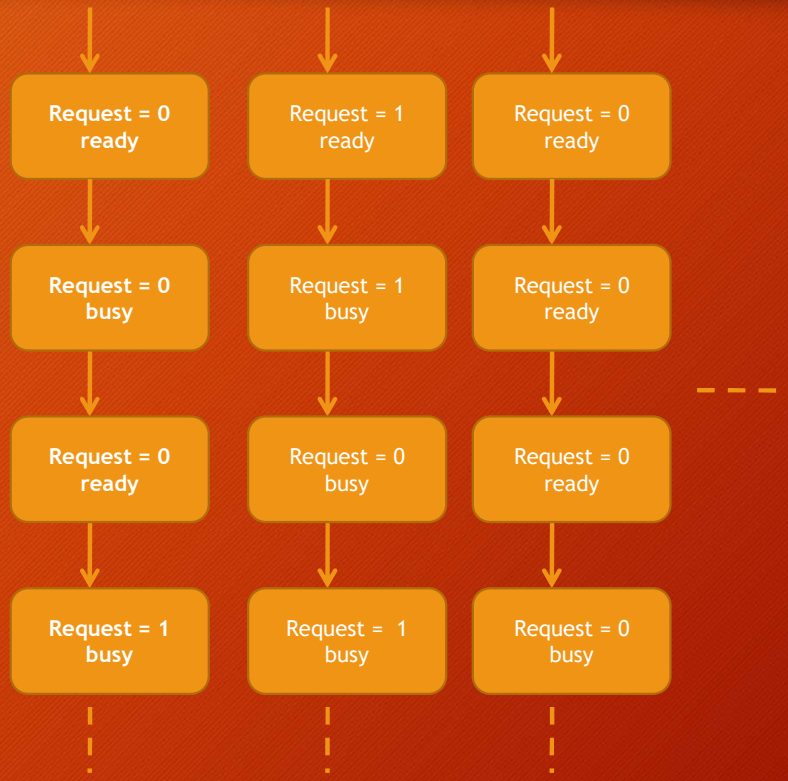


G (x >= 0)

Executions
Initial value of
x = 5

Models in NuSMV

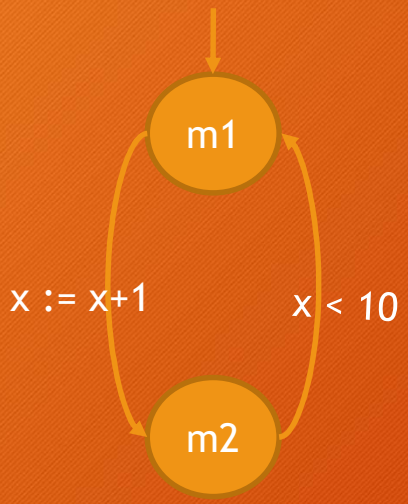


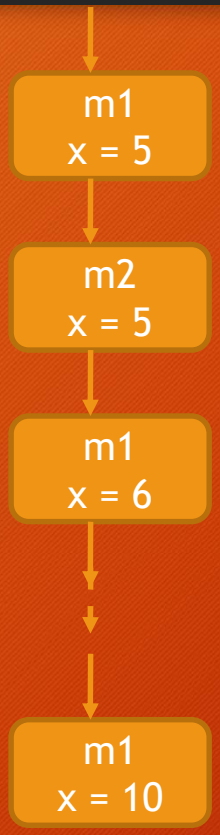


Executions

G (request = false)

Models in NuSMV



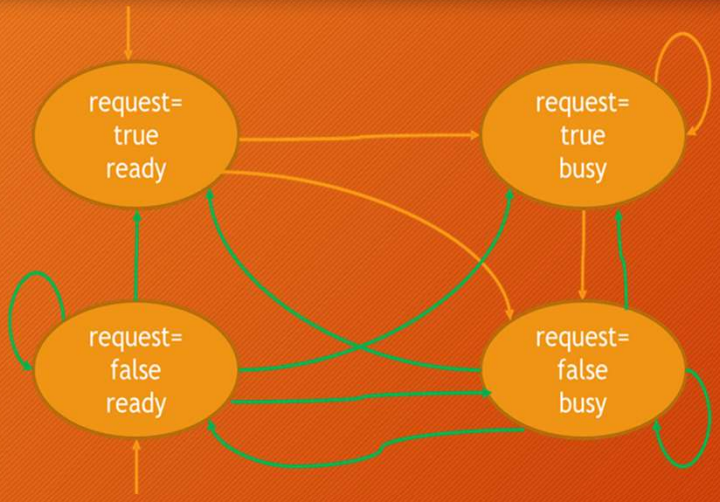


Executions
Initial value of
x = 5

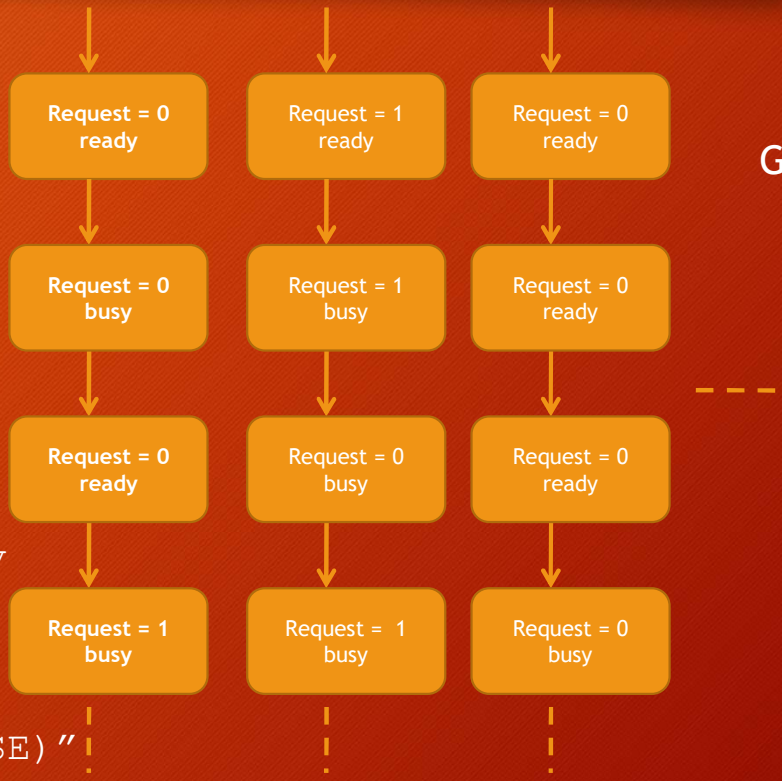
G (x >= 0)

```
>nusmv -int
NuSMV>read_model -i pg-demo.smv
NuSMV>flatten_hierarchy
NuSMV>encode_variables
NuSMV>build_model
NuSMV>check_ltlspec -p "G (x >= 0)"
```

Models in NuSMV




```
>nusmv -int
NuSMV>read_model -i request-busy-demo.smv
NuSMV>flatten_hierarchy
NuSMV>encode_variables
NuSMV>build_model
NuSMV>check_ltlspec -p "G (request = FALSE)"
```



G (request = false)

Models in NuSMV

Transition system satisfies a requirement

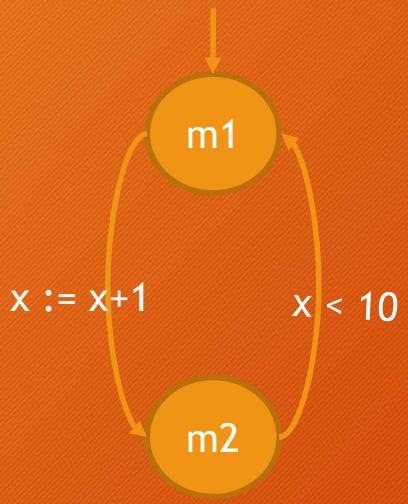
means

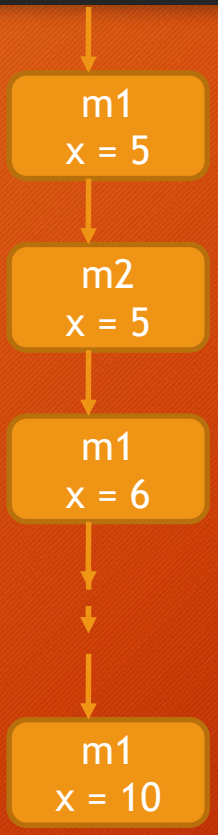
all its executions satisfy the requirement

Models in NuSMV

- Requirement type 2 : F

Models in NuSMV

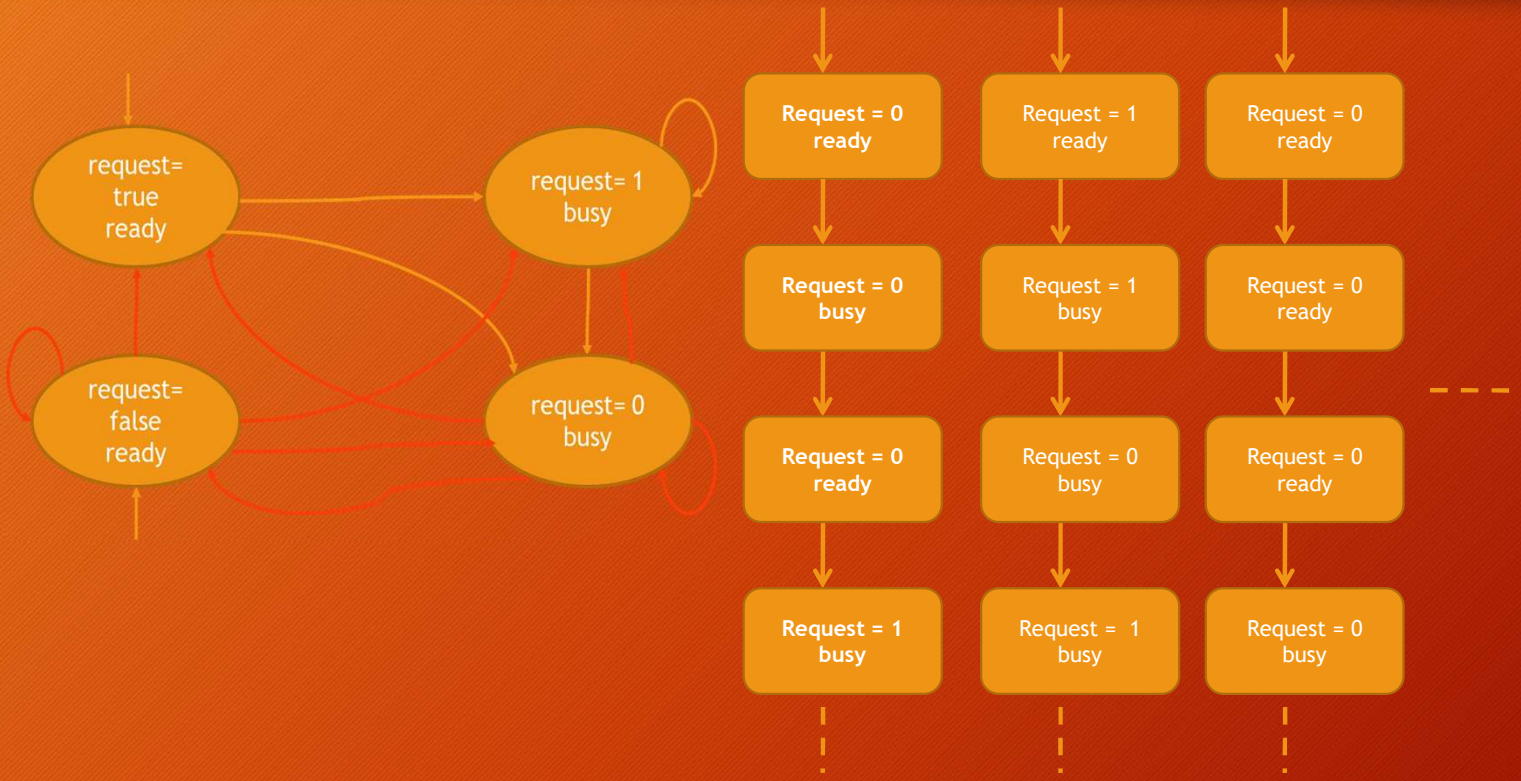




F (x >= 7)

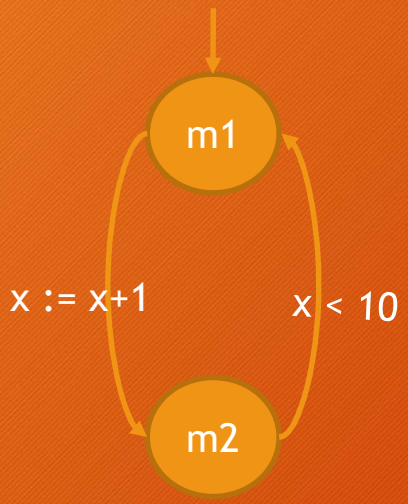
Executions
Initial value of
x = 5

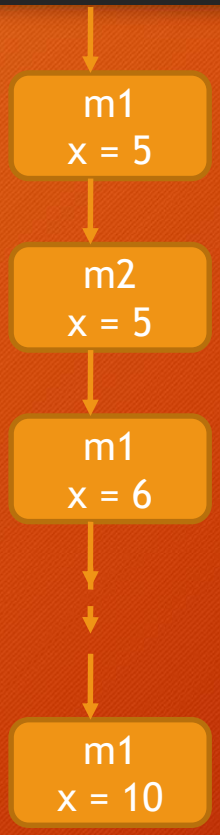
Models in NuSMV



F (request = true)

Models in NuSMV



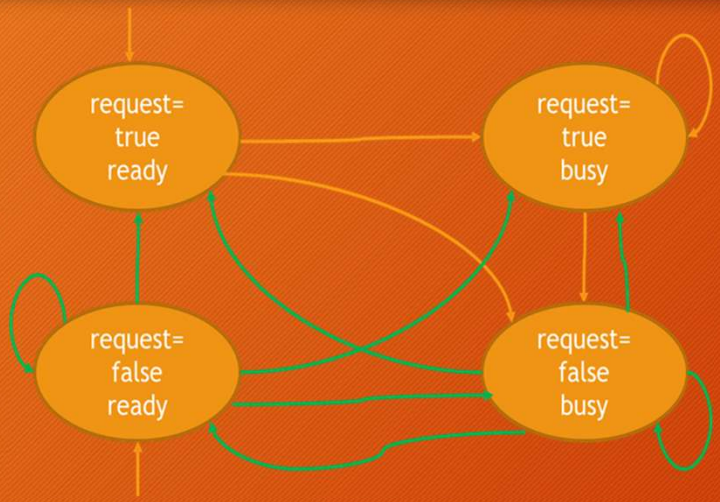


Executions
Initial value of
x = 5

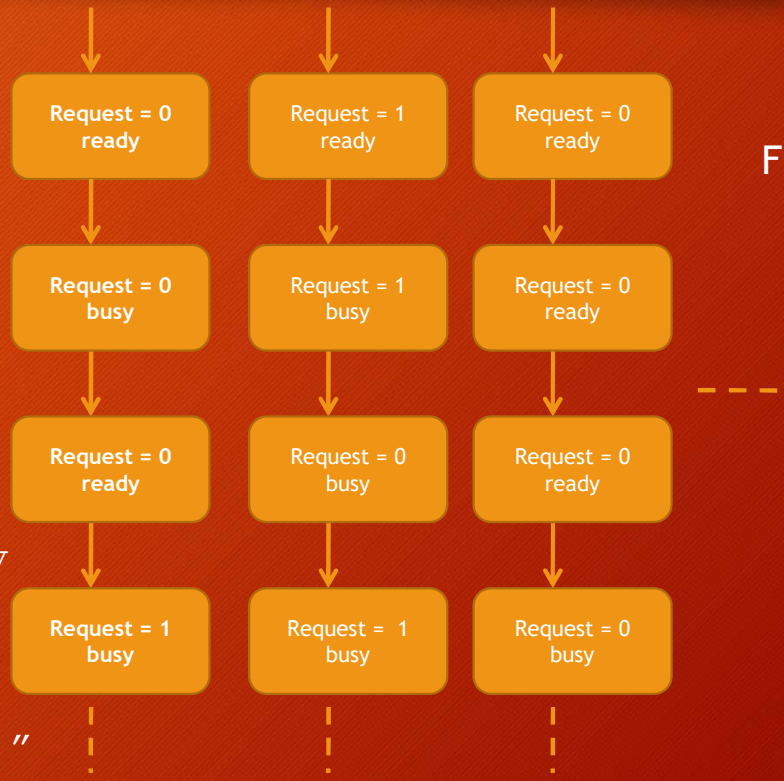
F (x >= 7)

```
>nusmv -int
NuSMV>read_model -i pg-demo.smv
NuSMV>flatten_hierarchy
NuSMV>encode_variables
NuSMV>build_model
NuSMV>check_ltlspec -p "F (x >= 7)"
```

Models in NuSMV

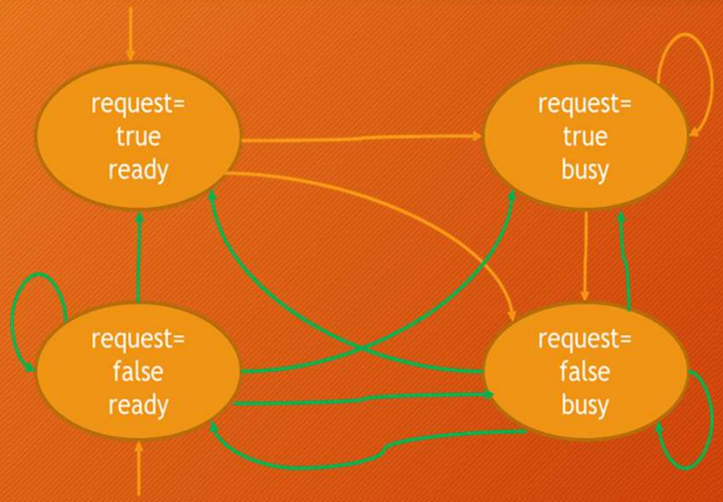



```
>nusmv -int
NuSMV>read_model -i request-busy-demo.smv
NuSMV>flatten_hierarchy
NuSMV>encode_variables
NuSMV>build_model
NuSMV>check_ltlspec -p "F (request =TRUE)"
```

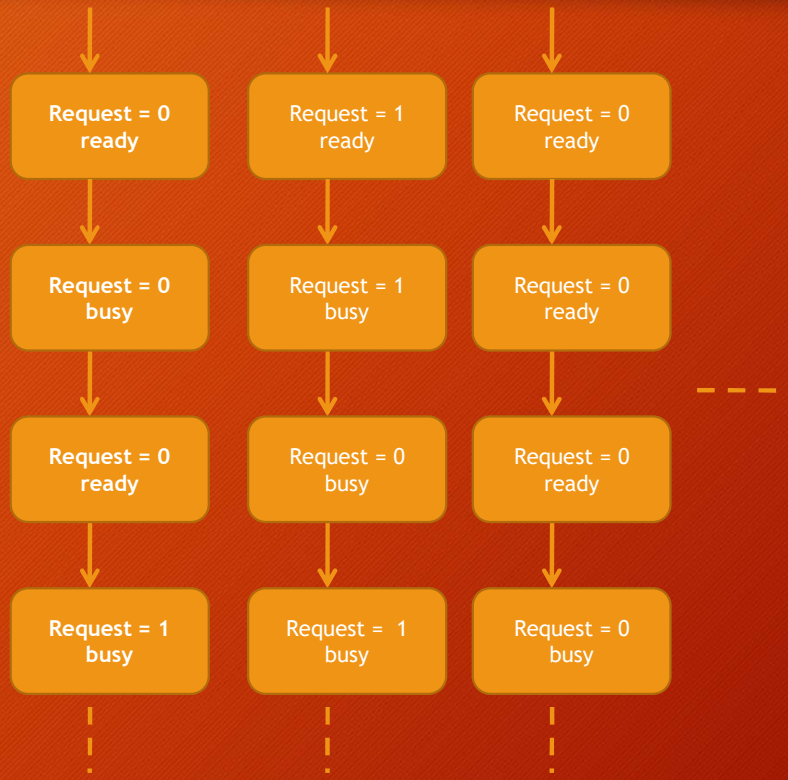


F (request = true)

Models in NuSMV

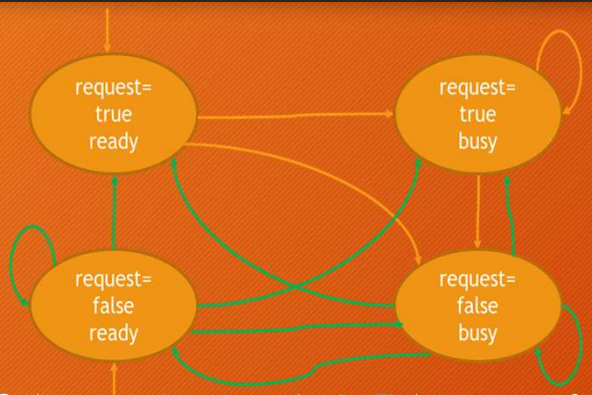


G (request = true) → F (status = busy)

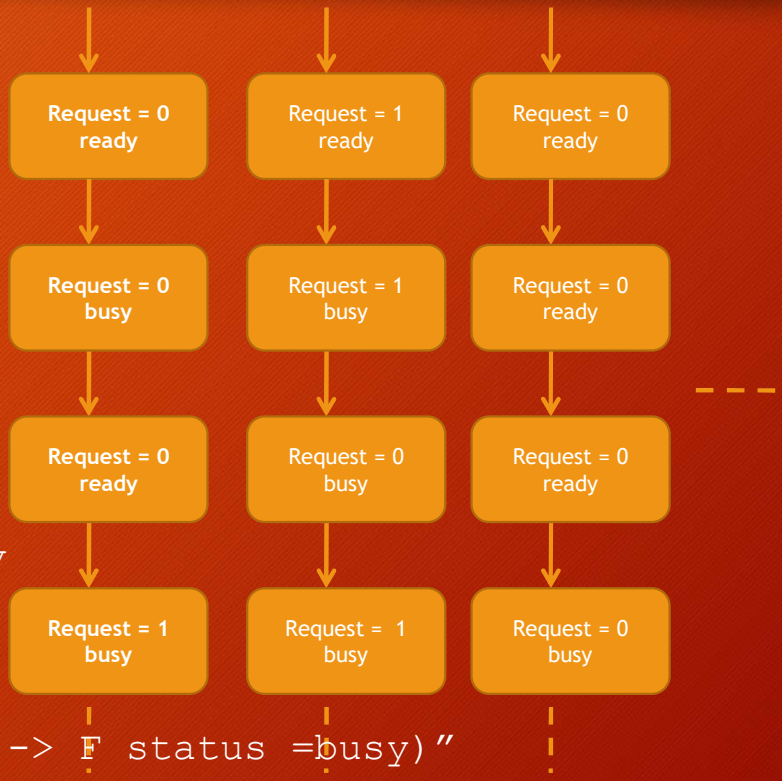


Executions

Models in NuSMV



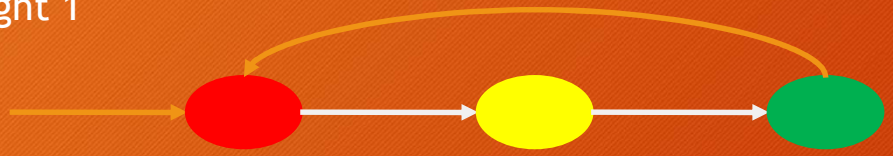
G (request = true) → F (status = busy)




```
>nusmv -int
NuSMV>read_model -i request-busy-demo.smv
NuSMV>flatten_hierarchy
NuSMV>encode_variables
NuSMV>build_model
NuSMV>check_ltlspec -p "G (request =TRUE -> F status =busy)"
```

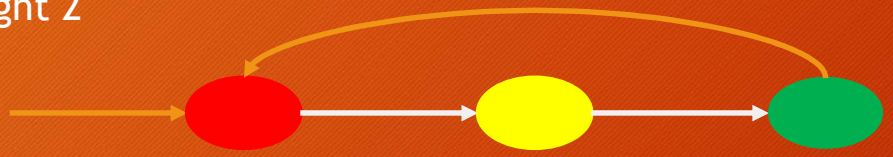
Modelling in NuSMV

Light 1



Light 2.red

Light 2

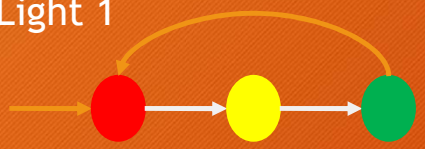


Light1.red

If a light is red, it can stay red
If it goes yellow, it should become green
If it is green, it can stay green

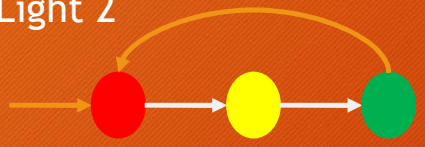
Modelling in NuSMV

Light 1



Light 2.red

Light 2



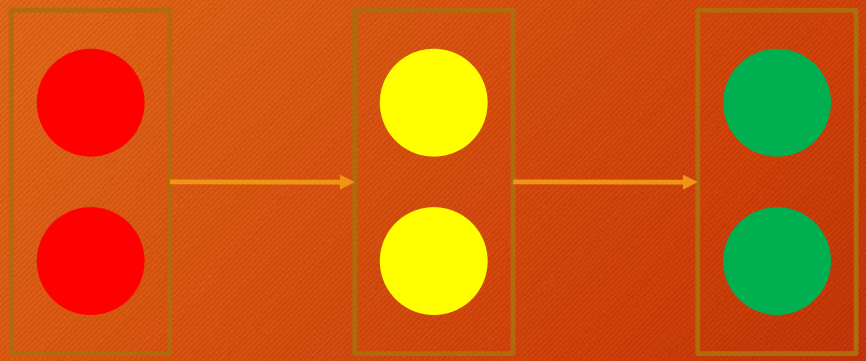
Light1.red

```
MODUE light(other)
VAR
state:{r, y, g};
ASSIGN
init(state) := r;
next(state) := case
     state = r & other = r : {r, y};
     state = y  : g;
     state=g : {g, r};
     TRUE : state;
     esac;

MODULE main
VAR
tl1 : light(tl2.state);
tl2 : light(tl1.state);
```

If a light is red, it can stay red
If it goes yellow, it should become green
If it is green, it can stay green

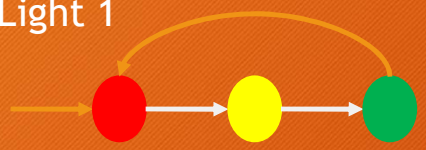
Synchronous Composition



Both lights can simultaneously become green

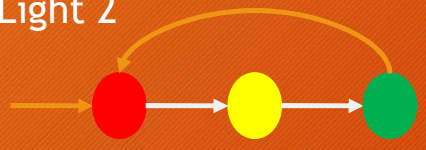
Asynchronous Composition

Light 1



Light 2.red

Light 2



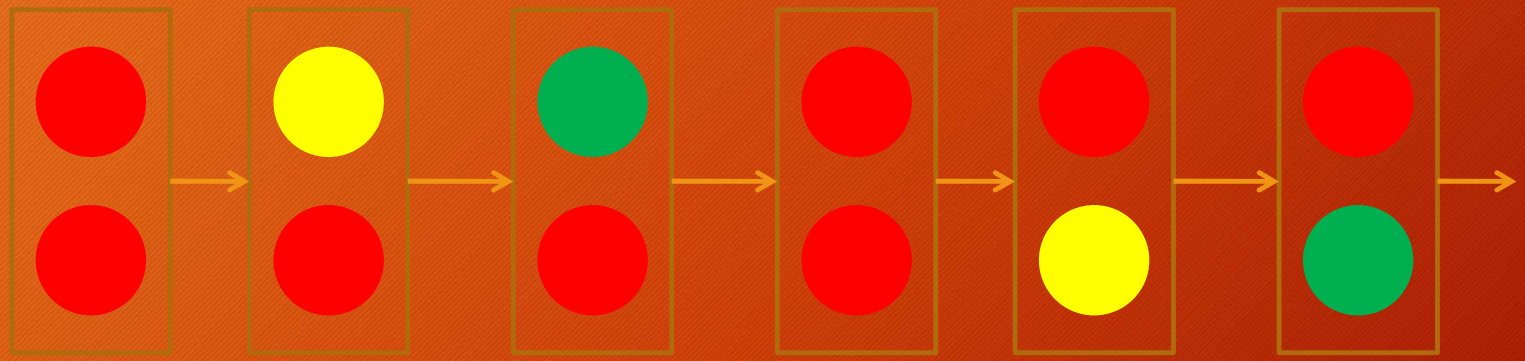
Light1.red

If a light is red, it can stay red
If it goes yellow, it should become green
If it is green, it can stay green

```
MODUE light(other)
VAR
state:{r, y, g};
ASSIGN
init(state) := r;
next(state) := case
     state = r & other = r : {r, y};
               state = y   : g;
     state = g : {g, r};
     TRUE : state;
     esac;


MODULE main
VAR
tl1 : process light(tl2.state);
tl2 : process light(tl1.state);
```

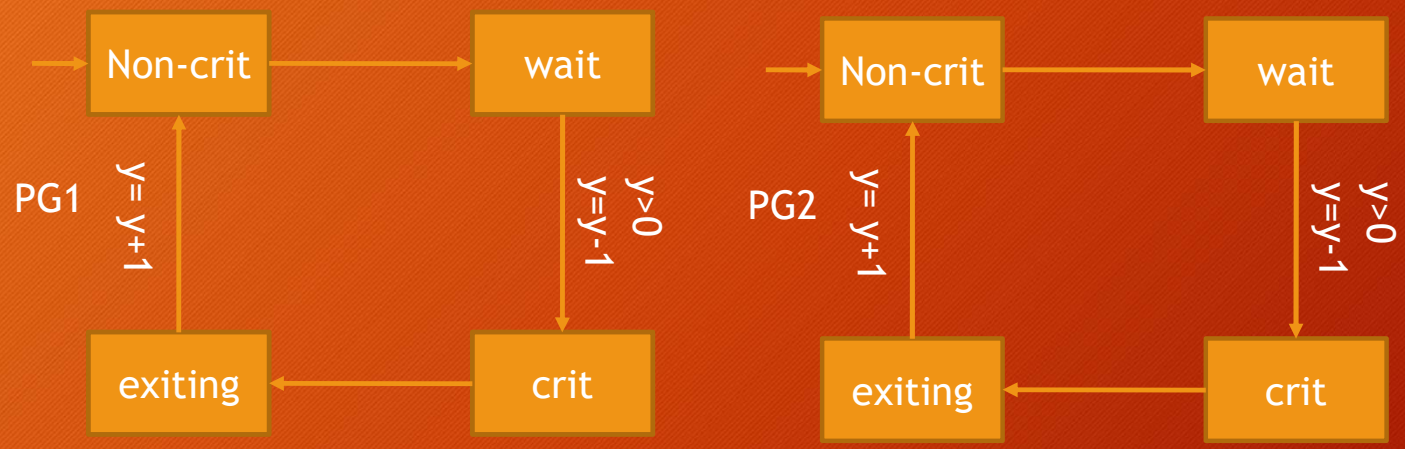
Asynchronous Composition



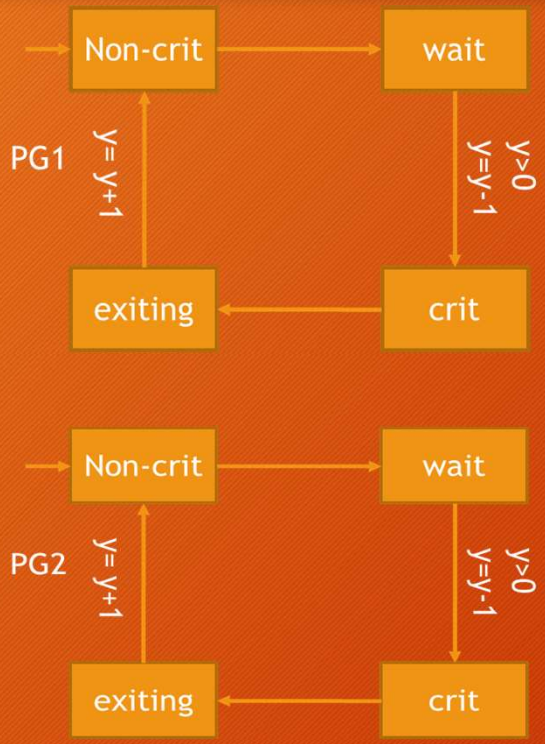
Only one lights can become green

check_ltlspec –p “! F (tl1.state = g & tl2.state = g) ”

Asynchronous Composition



Asynchronous Composition




```
MODULE thread(y)
VAR
location:{nc, w, c, e};
ASSIGN
init(location):= nc;
next(location):= case
     location = nc : {nc, w};
     location = w & y > 0: c;
     location = c : {c,e};
     location = e: nc;
     TRUE: location;
     esac;
next(y) := case
     location = w & y > 0 : y-1;
     location = e & y = 0 : y+1;
     TRUE: y;
     esac;
```

```
MODLE main
VAR
y-main : 0..1;
Prg1 : process thread(y-main);
Prg2 : process thread(y-main);
ASSIGN
init(y-main) := 1;
```