Unit-2: Model-checker NuSMV

B. Srivathsan

Chennai Mathematical Institute

NPTEL-course

July - November 2015

Module 4:

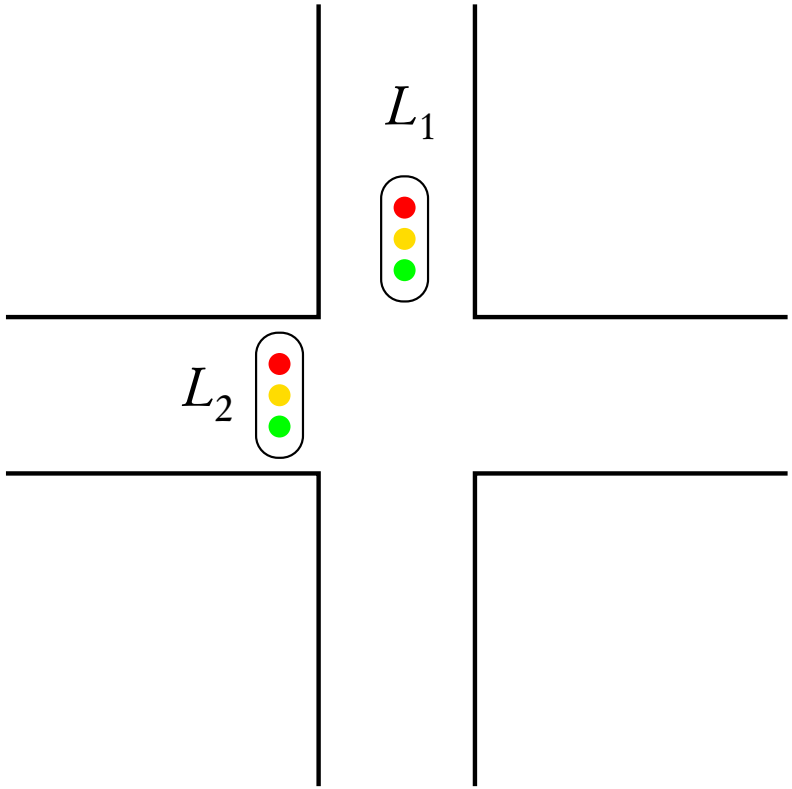
Modeling concurrent systems

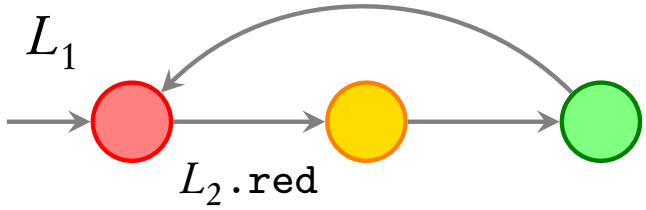
Synchronous vs. Asynchronous systems

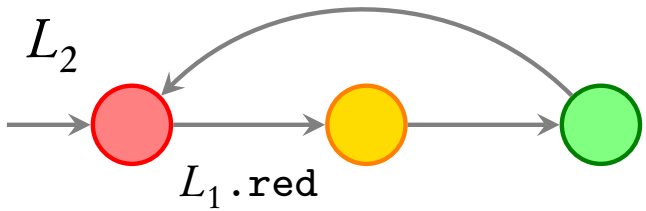
Acknowledgements:

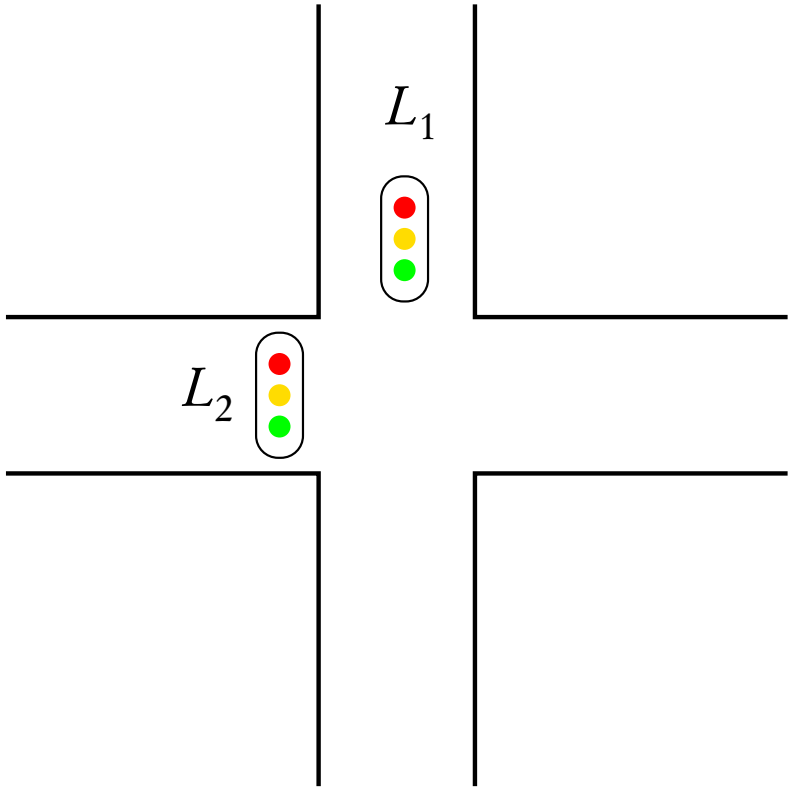
This part of module taken from lecture slides of

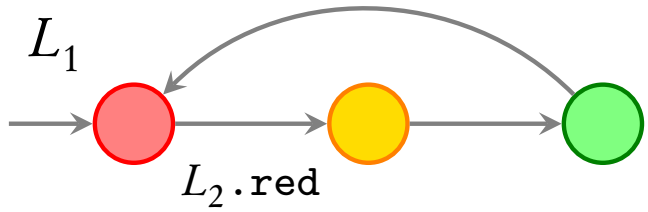
Prof. Supratik Chakraborty, IIT Bombay

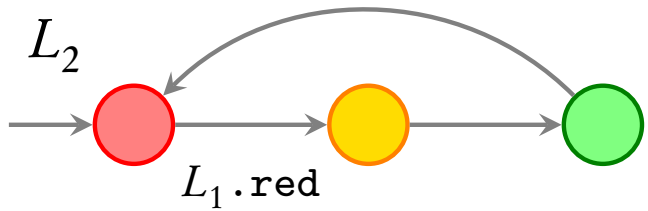

L_1
L_2


L_1
L_2.red


L_2
L_1.red







If a light is **red**, it can **stay red** for an **arbitrary period**

If it goes **yellow**, it should become **green within one cycle**

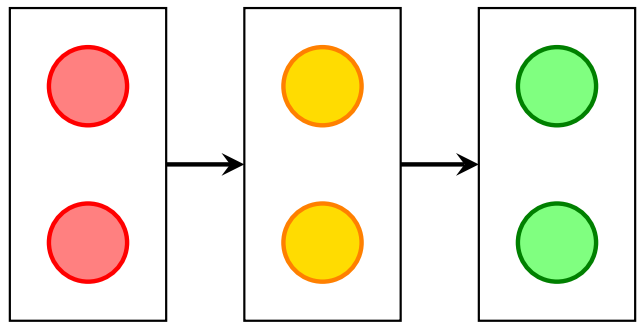
If it is **green**, it can **stay green** for an **arbitrary period**

```
MODULE 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);
```

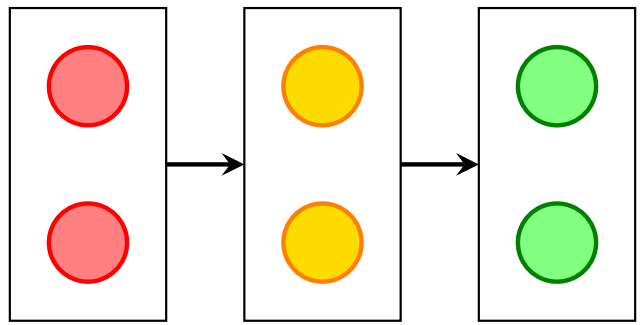
Synchronous composition

```
MODULE 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);
```

Synchronous composition



Synchronous composition

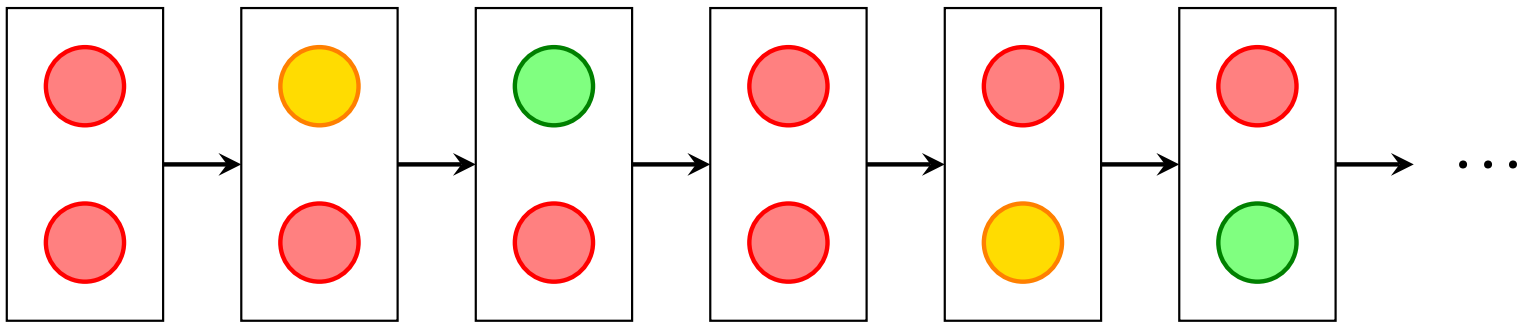


Both lights can **simultaneously become green!**

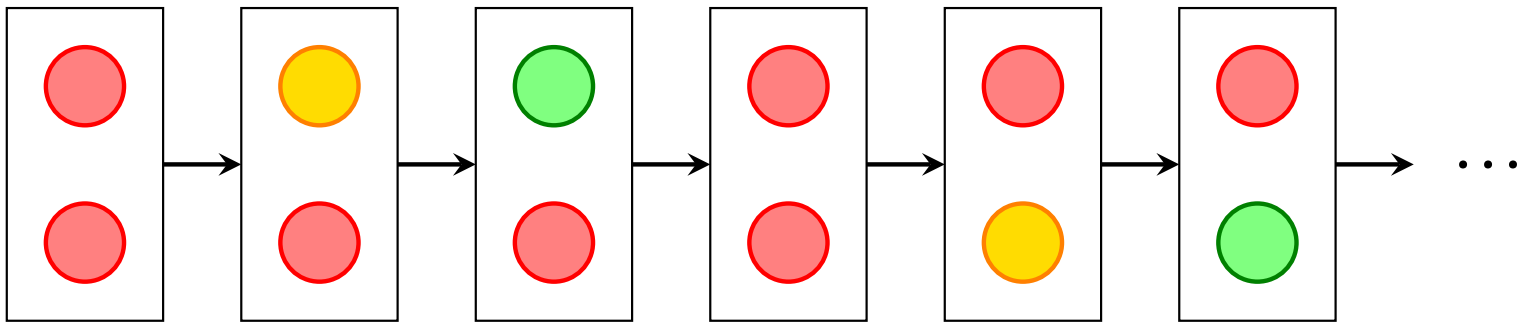
Asynchronous composition

```
MODULE 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



Asynchronous composition



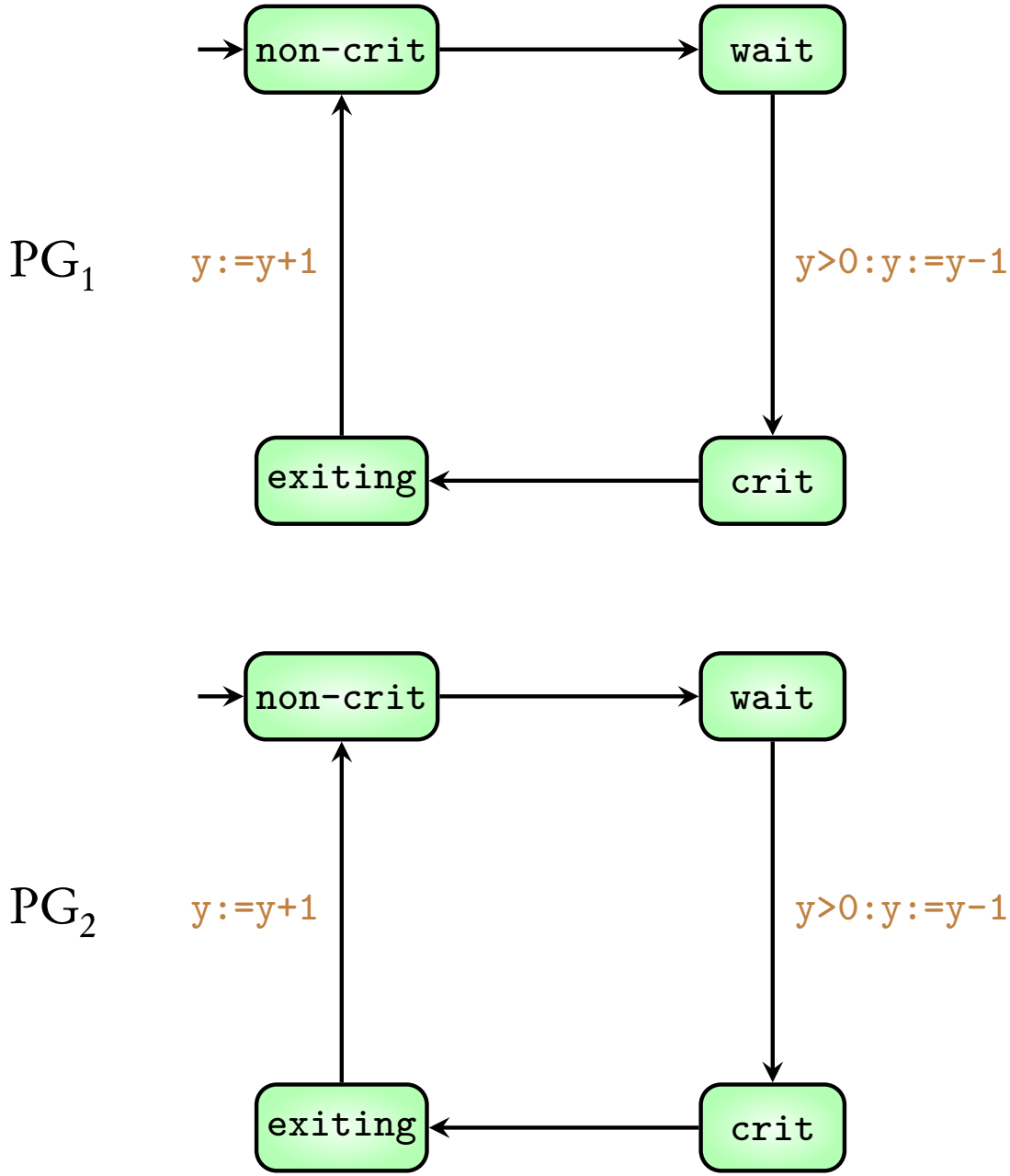
Only one light can become **green** at a time

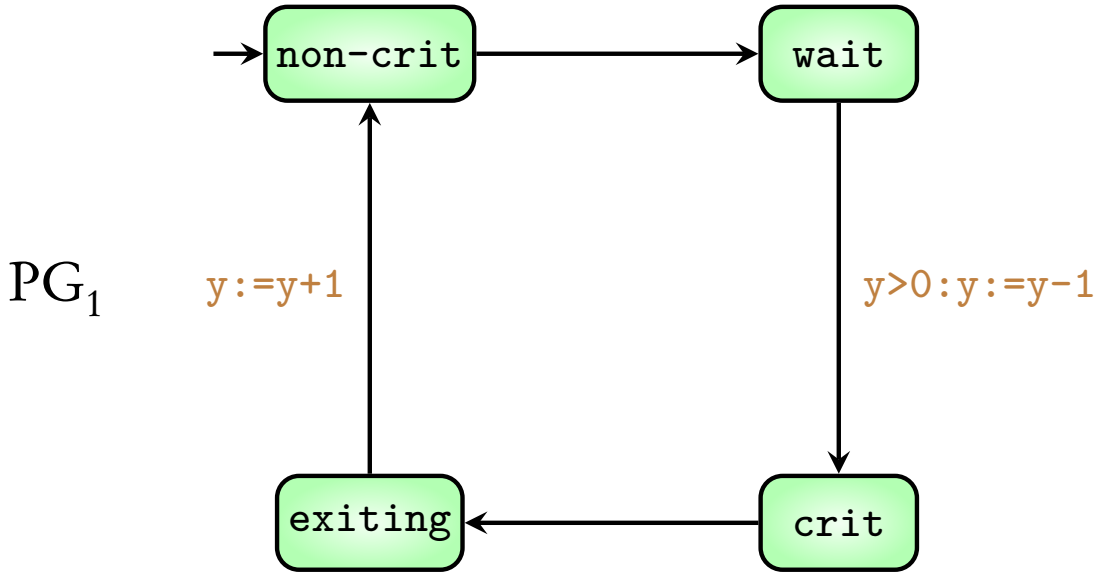
- **Synchronous:**
  - all assignments to all modules made **simultaneously**
  - suitable when all modules are synchronized to a **global clock**
- **Asynchronous:**
  - execution of modules is **interleaved**
  - at a time, **only one** module executes
  - choice of next module to be executed is **non-deterministic**
  - suitable when **no assumptions** can be made **about communication delay** between modules

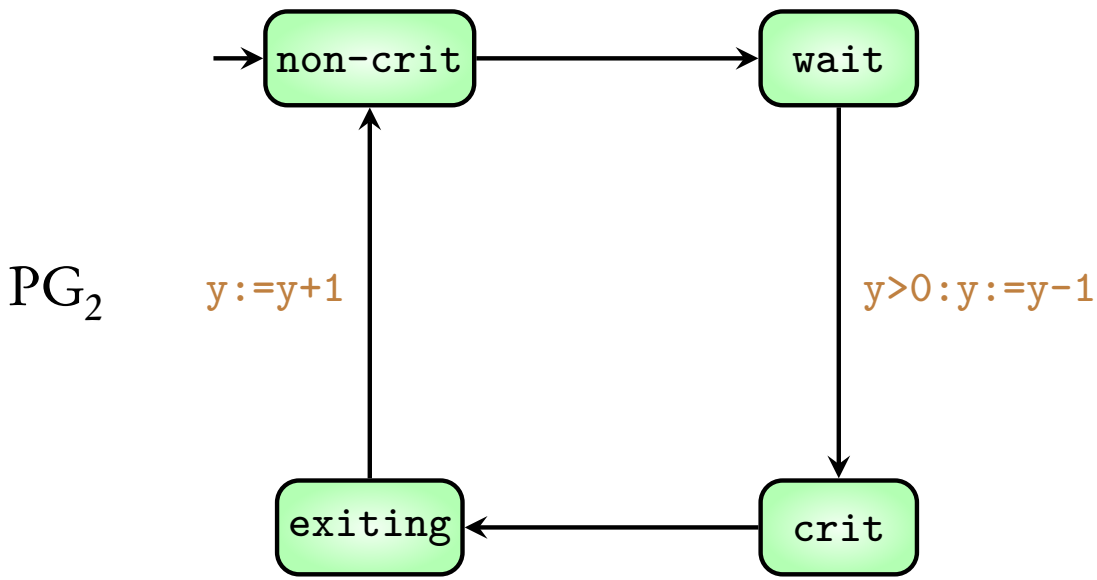
Synchronous
vs.
Asynchronous
systems

Synchronous vs. Asynchronous systems

Mutual Exclusion


PG1
non-crit
wait
y:=y+1
y>0:y:=y-1
exiting
crit
PG2
non-crit
wait
y:=y+1
y>0:y:=y-1
exiting
crit


PG1
non-crit
wait
y:=y+1
y>0:y:=y-1
exiting
crit


PG2
non-crit
wait
y:=y+1
y>0:y:=y-1
exiting
crit

NuSMV demo

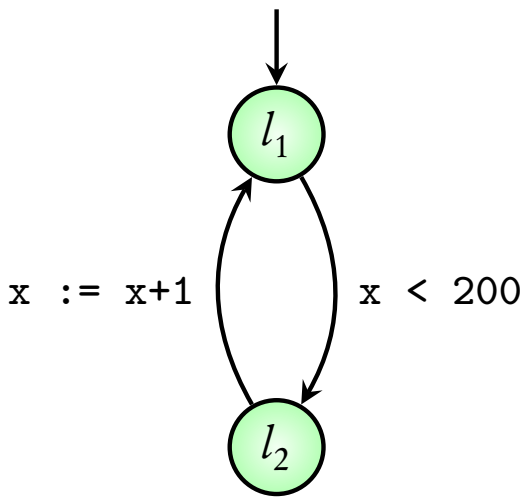
Synchronous
vs.
Asynchronous
systems

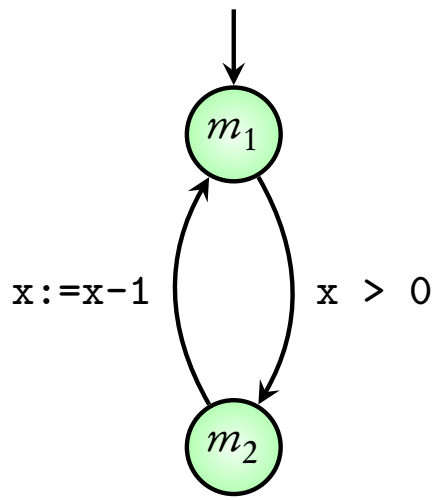
Mutual Exclusion

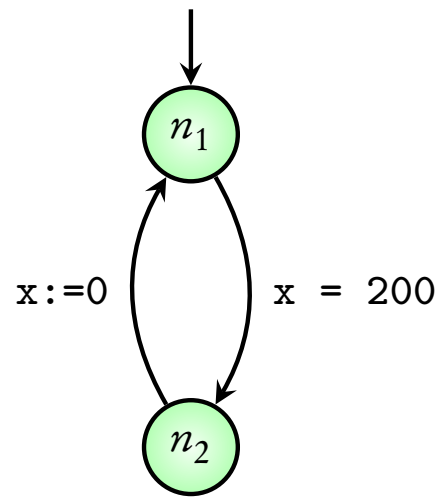
```
while x < 200
    x := x+1
```

```
while x>0
    x := x-1
```

```
while x=200
    x := 0
```



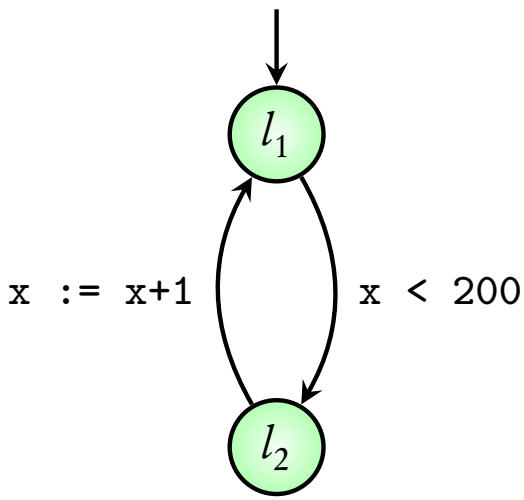


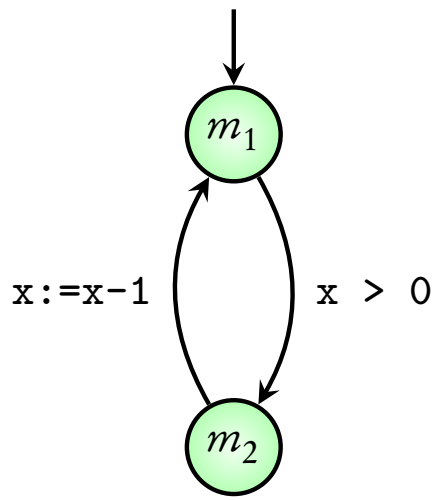


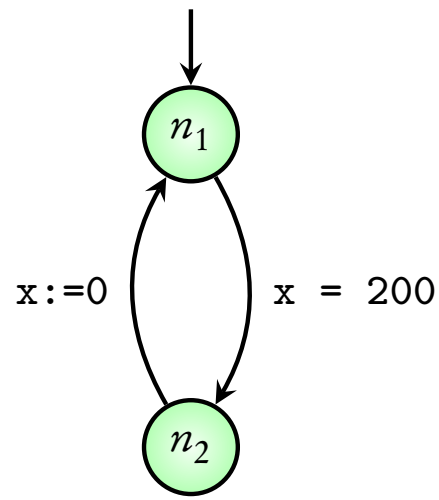
```
while x < 200
    x := x+1
```

```
while x>0
    x := x-1
```

```
while x=200
    x := 0
```







NuSMV demo

Synchronous vs. Asynchronous systems

Mutual Exclusion

Concurrent programs example