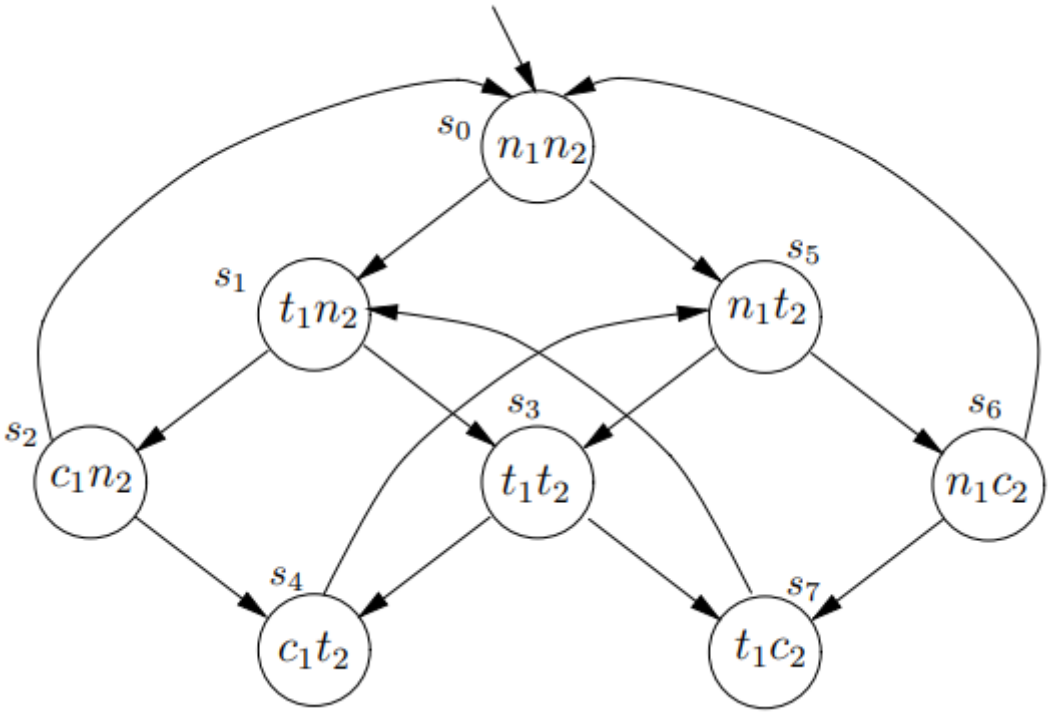
# Lab2

李远航 PB20000137

## 一、实验内容

使用`NuSMV`实现`first-attempt model`，用`CTL`设计`Non-blocking`，`No strict sequencing`，并验证所有四个性质




## 二、模型创建

进程的状态有$\{n,t,c\}$三种，且两个进程不能同时被执行，据此，可以写出如下的状态转换逻辑：

```
init(value) := n;
next(value) := case
                    (value = n) & (other = n) : {t};
                    (value = n) & (other = t) : {t};
                    (value = n) & (other = c) : {t};
                    (value = t) & (other = n) : {c};
                    (value = t) & (other = t) : {c};
                    (value = c) & (other = n) : {n};
                    (value = c) & (other = t) : {n};
                    TRUE : value;
                esac;
```

完成模型的创建，同时需要注意main模块要设置成异步的形式

## 三、性质验证

1. `Safety`：进程不会同时执行

   两个进程不可能同时处于c状态，即：

   $$AG \neg (c_1 \wedge c_2)$$

2. `Liveness`：进程进入临界区总会被执行

   进程在到达状态t之后，总会到达状态c，即：

   $$AG\ (t_1 \rightarrow AF\ c_1)$$

3. `Non-blocking`：进程随时可以请求进入临界区

   进程在n状态的后继状态总有一个后继是t，即：

   $$AG\ (n_1 \rightarrow EX\ t_1)$$

4. `No strict sequencing`：进程不需要按照严格的顺序进入临界区

   进程两次执行之间不会有其他进程，即：

   $$EF\ (c_1 \wedge E[\ c_1\ U\ (\neg\ c_1 \wedge E[\ \neg c_2\ U\ c_1)])$$

## 四、实验结果

```
-- specification AG !(p1.value = c & p2.value = c)  is true
-- specification AG (p1.value = t -> AF p1.value = c)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    p1.value = n
    p2.value = n
  -> Input: 1.2 <-
    _process_selector_ = p1
    running = FALSE
    p2.running = FALSE
    p1.running = TRUE
  -- Loop starts here
  -> State: 1.2 <-
    p1.value = t
  -> Input: 1.3 <-
    _process_selector_ = p2
    p2.running = TRUE
    p1.running = FALSE
  -> State: 1.3 <-
    p2.value = t
  -> Input: 1.4 <-
  -> State: 1.4 <-
    p2.value = c
  -> Input: 1.5 <-
    _process_selector_ = p1
    p2.running = FALSE
    p1.running = TRUE
  -> State: 1.5 <-
  -> Input: 1.6 <-
```

```
      _process_selector_ = p2
      p2.running = TRUE
      p1.running = FALSE
    -> State: 1.6 <-
      p2.value = n
-- specification AG (p2.value = t -> AF p2.value = c)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
    -> State: 2.1 <-
      p1.value = n
      p2.value = n
    -> Input: 2.2 <-
      _process_selector_ = p2
      running = FALSE
      p2.running = TRUE
      p1.running = FALSE
    -- Loop starts here
    -> State: 2.2 <-
      p2.value = t
    -> Input: 2.3 <-
      _process_selector_ = p1
      p2.running = FALSE
      p1.running = TRUE
    -> State: 2.3 <-
      p1.value = t
    -> Input: 2.4 <-
    -> State: 2.4 <-
      p1.value = c
    -> Input: 2.5 <-
      _process_selector_ = p2
      p2.running = TRUE
      p1.running = FALSE
    -> State: 2.5 <-
    -> Input: 2.6 <-
      _process_selector_ = p1
      p2.running = FALSE
      p1.running = TRUE
    -- Loop starts here
    -> State: 2.6 <-
      p1.value = n
    -> Input: 2.7 <-
      _process_selector_ = main
      running = TRUE
      p1.running = FALSE
    -> State: 2.7 <-
-- specification AG (p1.value = n -> EX p1.value = t)  is true
-- specification AG (p2.value = n -> EX p2.value = t)  is true
-- specification EF (p1.value = c & E [ p1.value = c U (!(p1.value = c) & E [ !
(p2.value = c) U p1.value = c ] ) ] )  is true
-- specification EF (p2.value = c & E [ p2.value = c U (!(p2.value = c) & E [ !
(p1.value = c) U p2.value = c ] ) ] )  is true
```

可以看到模型不符合`Liveness`，即进程进入临界区未能成功被执行，出现了死锁现象

## 五、实验收获

1. 对模型检测有了更深的理解
2. 增强了逻辑思维能力