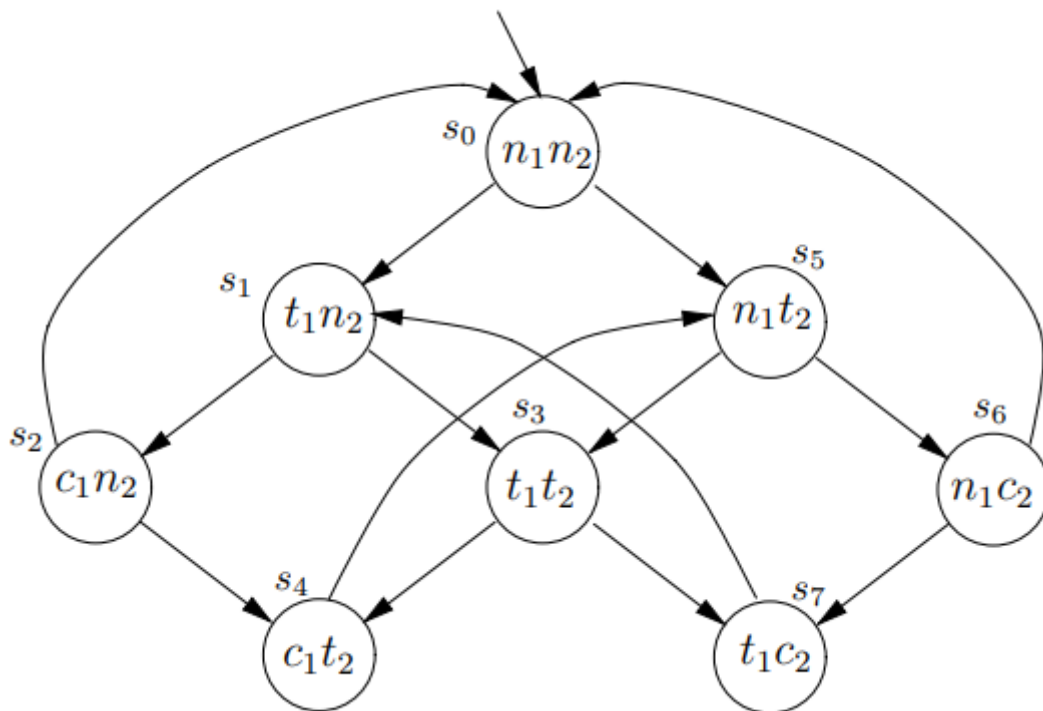


Lab2

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：

```
1  init(value) := n;
2  next(value) := case
3      (value = n) & (other = n) : {t};
4      (value = n) & (other = t) : {t};
5      (value = n) & (other = c) : {t};
6      (value = t) & (other = n) : {c};
7      (value = t) & (other = t) : {c};
8      (value = c) & (other = n) : {n};
9      (value = c) & (other = t) : {n};
10     TRUE : value;
11     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

```
1  -- specification AG !(p1.value = c & p2.value = c) is true
2  -- specification AG (p1.value = t -> AF p1.value = c) is false
3  -- as demonstrated by the following execution sequence
4  Trace Description: CTL Counterexample
5  Trace Type: Counterexample
6  -> State: 1.1 <-
7      p1.value = n
8      p2.value = n
9  -> Input: 1.2 <-
10     _process_selector_ = p1
11     running = FALSE
12     p2.running = FALSE
13     p1.running = TRUE
14  -- Loop starts here
15  -> State: 1.2 <-
16     p1.value = t
17  -> Input: 1.3 <-
18     _process_selector_ = p2
19     p2.running = TRUE
20     p1.running = FALSE
21  -> State: 1.3 <-
22     p2.value = t
23  -> Input: 1.4 <-
24  -> State: 1.4 <-
25     p2.value = c
26  -> Input: 1.5 <-
27     _process_selector_ = p1
28     p2.running = FALSE
29     p1.running = TRUE
30  -> State: 1.5 <-
31  -> Input: 1.6 <-
```

```

32     _process_selector_ = p2
33     p2.running = TRUE
34     p1.running = FALSE
35     -> State: 1.6 <-
36     p2.value = n
37     -- specification AG (p2.value = t -> AF p2.value = c) is false
38     -- as demonstrated by the following execution sequence
39     Trace Description: CTL Counterexample
40     Trace Type: Counterexample
41     -> State: 2.1 <-
42     p1.value = n
43     p2.value = n
44     -> Input: 2.2 <-
45     _process_selector_ = p2
46     running = FALSE
47     p2.running = TRUE
48     p1.running = FALSE
49     -- Loop starts here
50     -> State: 2.2 <-
51     p2.value = t
52     -> Input: 2.3 <-
53     _process_selector_ = p1
54     p2.running = FALSE
55     p1.running = TRUE
56     -> State: 2.3 <-
57     p1.value = t
58     -> Input: 2.4 <-
59     -> State: 2.4 <-
60     p1.value = c
61     -> Input: 2.5 <-
62     _process_selector_ = p2
63     p2.running = TRUE
64     p1.running = FALSE
65     -> State: 2.5 <-
66     -> Input: 2.6 <-
67     _process_selector_ = p1
68     p2.running = FALSE
69     p1.running = TRUE
70     -- Loop starts here
71     -> State: 2.6 <-
72     p1.value = n
73     -> Input: 2.7 <-
74     _process_selector_ = main
75     running = TRUE
76     p1.running = FALSE
77     -> State: 2.7 <-
78     -- specification AG (p1.value = n -> EX p1.value = t) is true
79     -- specification AG (p2.value = n -> EX p2.value = t) is true
80     -- specification EF (p1.value = c & E [ p1.value = c U (!(p1.value = c) & E [ !
      (p2.value = c) U p1.value = c ] ) ] ) is true
81     -- 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入的理解
2. 增强了逻辑思维能力