# 形式化方法导引第七次作业

## 两个CTL实现

- Non-blocking的CTL实现：$AG(t1 \rightarrow AF(c1))$，即对于任何一条路径的任意点，如果这个点满足t1，则后面任意一条路径都会通过c1，这样就使得c1一定会实现。
- Non strict sequencing的CTL实现$EF(E[c1\ U\ E[(\neg c1 \& \neg c2)\ U\ c1]])$，即整个图中存在一条路径，它从状态c1切换到非c1及非c2的状态后又切换回c1，保证c1c2不是严格按序切换的。

转换成代码分别如下表示：

- Non-blocking: `CTLSPEC AG((pr1.st = t) -> AF (pr1.st = c))`
- Non strict sequencing: `CTLSPEC EF(E[pr1.st=c U E[(pr1.st!=c&pr2.st!=c)U(pr1.st=c)]])`

## 完整代码

以下是完整实验代码：

```
MODULE main
    VAR
        pr1:process prc(pr2.st,turn,FALSE);
        pr2:process prc(pr1.st,turn,TRUE);
        turn:boolean;
    ASSIGN
        init(turn):=TRUE;

    LTLSPEC G!((pr1.st = c) & (pr2.st = c))
    LTLSPEC G((pr1.st = t) -> F (pr1.st = c))
    CTLSPEC AG((pr1.st = t) -> AF (pr1.st = c))
    CTLSPEC EF(E[pr1.st=c U E[(pr1.st!=c&pr2.st!=c)U(pr1.st=c)]])
MODULE prc(other-st,turn,myturn)
    VAR
        st:{n,t,c};
    ASSIGN
        init(st):=n;
        next(st):=
            case
                (st=n):{t,n};
                (st=t)&(other-st=n):c;
                (st=t)&(other-st=t)&(turn=myturn):c;
                (st=c):{c,n};
                TRUE:st;
            esac;
        next(turn):=
            case
                turn=myturn&st=c:!turn;
                TRUE:turn;
            esac;
    FAIRNESS running
    FAIRNESS !(st=c)
```

## 运行结果

(前面欢迎信息已省去)

```
-- specification AG (pr1.st = t -> AF pr1.st = c)  is true
-- specification EF E [ pr1.st = c U E [ (pr1.st != c & pr2.st != c) U pr1.st = c ]
]   is true
-- specification  G !(pr1.st = c & pr2.st = c)  is true
-- specification  G (pr1.st = t ->  F pr1.st = c)  is true
```

四个条件均满足。