实验报告

PB20061338 柯志伟

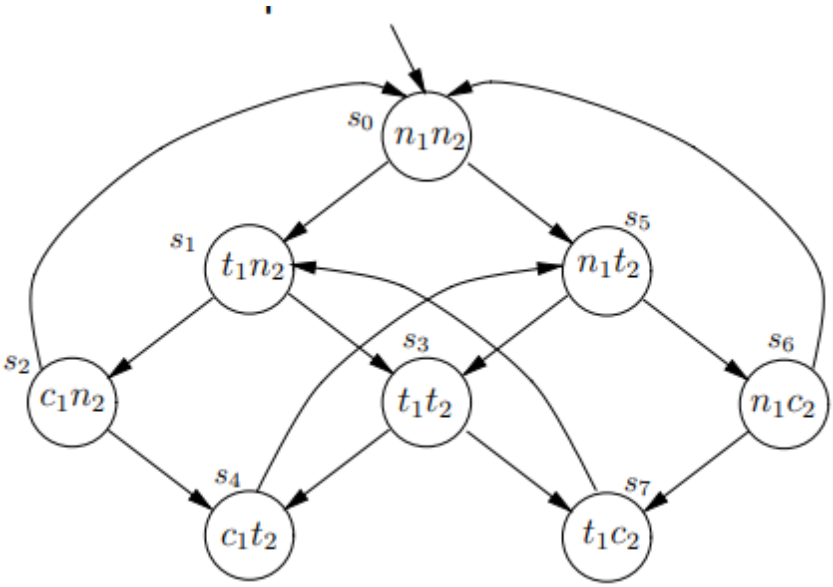
1 实验题目

```
使用 NuSMV 实现 PPT 中 first-attempt model，要求：
- 用 CTL 设计 Non-blocking，No strict sequencing，并验证所有四个性质
- 给出源码、实验报告
```

2 实验过程

2.1 使用NuSMV描述first-attempt model

2.1.1 模型及约束



Specify the properties of the protocol using NuSMV

- *Safety*: Only *one process* is in its *critical section* at any time.
- *Liveness*: Whenever any process *requests* to enter its critical section, it will *eventually be permitted* to do so.
- *Non-blocking*: A process can *always* request to *enter* its critical section.
- *No strict sequencing*: Processes need not enter their critical section in strict sequence.

2.1.2 使用NuSMV描述模型中的状态转移以及待检查的约束

```
MODULE main
    VAR
        st: array 0..1 of {c, n, t};
    ASSIGN
        init(st[0]) := n;
        init(st[1]) := n;
    TRANS
        st[0] = n & st[1] = n -> (next(st[0]) = t & next(st[1]) = n) |
                                 (next(st[0]) = n & next(st[1]) = t);
    TRANS
        st[0] = t & st[1] = n -> (next(st[0]) = c & next(st[1]) = n) |
                                 (next(st[0]) = t & next(st[1]) = t);
    TRANS
        st[0] = n & st[1] = t -> (next(st[0]) = t & next(st[1]) = t) |
                                 (next(st[0]) = n & next(st[1]) = c);
    TRANS
        st[0] = c & st[1] = n -> (next(st[0]) = n & next(st[1]) = n) |
                                 (next(st[0]) = c & next(st[1]) = t);
    TRANS
        st[0] = t & st[1] = t -> (next(st[0]) = c & next(st[1]) = t) |
                                 (next(st[0]) = t & next(st[1]) = c);
    TRANS
        st[0] = n & st[1] = c -> (next(st[0]) = t & next(st[1]) = c) |
                                 (next(st[0]) = n & next(st[1]) = n);
    TRANS
        st[0] = c & st[1] = t -> (next(st[0]) = n & next(st[1]) = t);

    TRANS
        st[0] = t & st[1] = c -> (next(st[0]) = t & next(st[1]) = n);

    -- safety
    CTLSPEC
        AG!(st[0] = c & st[1] = c);

    -- liveness
    CTLSPEC
        AG(st[0] = t -> AF(st[0] = c));
    CTLSPEC
        AG(st[1] = t -> AF(st[1] = c));

    -- non-blocking
    CTLSPEC
        AG(st[0] = n -> EF(st[0] = t));
    CTLSPEC
        AG(st[1] = n -> EF(st[1] = t));

    -- no strict sequencing
    -- LTLSPEC
    --     G(st[0] = c -> (G st[0] = c | (st[0] = c  U (st[0] != c & G st[0] != c |
((st[0] != c) U st[1] = c))));
```

```
50      CTLSPEC
51          -- EF((st[0]=c) -> E[(st[0]=c) U E[(st[0]!=c & st[1]!=c)U(st[0]=c)]]);
52          AG(st[0] = c -> (AG(st[0] = c) | A[st[0] = c U (st[0] != c & AG(st[0] != c) |
    A[st[0] != c U st[1] =c])]));
53
```

2.2 模型检测结果

```
*** This version of NuSMV is linked to the CUDD library version 2.4.1
*** Copyright (c) 1995-2004, Regents of the University of Colorado

*** This version of NuSMV is linked to the MiniSat SAT solver.
*** See http://minisat.se/MiniSat.html
*** Copyright (c) 2003-2006, Niklas Een, Niklas Sorensson
*** Copyright (c) 2007-2010, Niklas Sorensson

-- specification AG !(st[0] = c & st[1] = c)  is true
-- specification AG (st[0] = t -> AF st[0] = c)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    st[0] = n
    st[1] = n
  -- Loop starts here
  -> State: 1.2 <-
    st[0] = t
  -> State: 1.3 <-
    st[1] = t
  -> State: 1.4 <-
    st[1] = c
  -> State: 1.5 <-
    st[1] = n
-- specification AG (st[1] = t -> AF st[1] = c)  is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -> State: 2.1 <-
    st[0] = n
    st[1] = n
  -- Loop starts here
  -> State: 2.2 <-
    st[1] = t
  -> State: 2.3 <-
    st[0] = t
  -> State: 2.4 <-
    st[0] = c
  -> State: 2.5 <-
    st[0] = n
-- specification AG (st[0] = n -> EF st[0] = t)  is true
-- specification AG (st[1] = n -> EF st[1] = t)  is true
-- specification EF (st[0] = c -> E [ st[0] = c U E [ (st[0] != c & st[1] != c) U st[0] = c ]  ] )  is true
```

2.2.1 Safety

保证两个进程不会同时进入c:

```
AG!(st[0] = c & st[1] = c);
```

对于起始点出发的所有路径上的所有时间点都不存在两个进程都在关键区。由结果可知，该条件满足

2.2.2 Liveness

无论哪个进程请求进入其临界区，它最终都会被允许进入:

```
AG(st[0] = t -> AF(st[0] = c));
AG(st[1] = t -> AF(st[1] = c));
```

由结果可知，当另一个进程在n->t->c->n->t->c...循环时，该进程不满足Liveness

2.2.3 Non-blocking

对于每一个满足n1的状态，都存在一个满足t1的后继状态:

```
AG(st[0] = n -> EF(st[0] = t));
AG(st[1] = n -> EF(st[1] = t));
```

由结果可知，该条件满足

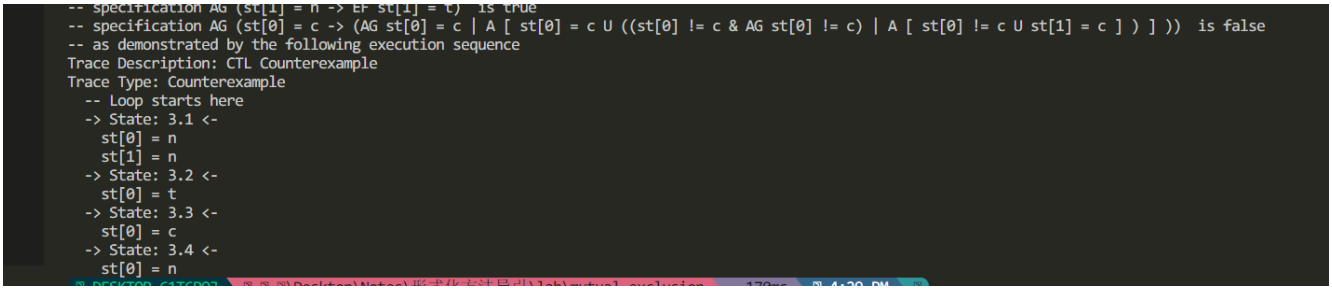
2.2.4 No strict sequencing

两个进程试图进入临界区时，不要求进程之间的访问按照固定的顺序进行，允许进程之间的访问顺序是随机的，
尝试使用两种方式描述:

直接`EF((st[0]=c) -> E[(st[0]=c) U E[(st[0]!=c & st[1]!=c)U(st[0]=c)]])`或者`AG(st[0] = c -> (AG(st[0] = c) | A[st[0] = c U (st[0] != c & AG(st[0] != c) | A[st[0] != c U st[1] =c])]))`的补集

采用直接表达的结果由上图可知,该条件满足

采用补集的表达结果如下:



由结果可知不满足，因此原条件满足

3 实验总结与反思

通过此次实验，了解了NuSMV在模型检测方面的应用，在实验过程中也遇到一些麻烦，如NuSMV中文参考资料过少，在寻找同时表达两个状态的转移时，
查找半天才找到使用数组表达并用Trans刻画状态转移;以及CTL语法中各个符号的优先级问题