Coursework 1

6CCS3VER

Rui Han Ji Chen
K200271

Question 1-

- The result used is based on the bit manipulation, the rotation consists of 0 to 4, so we can stand 0 as 000, 1 as 001, 2 as 010, 3 as 011 and 4 as 100. With only three digits we can process a counter. The Boolean property is already a binary variable, therefore, using only three digits is already enough.

```
NAME counter
VAR
    s1 : boolean;
    s2 : boolean;
    s3 : boolean;


INIT
    !s1 & !s2 & !s3;

RULES

!s1 & !s2 & !s3 :
    s1 := false ; s2 := false ; s3:= true
!s1 & !s2 & s3 :
    s1 := false ; s2 := true ; s3:= false
!s1 & s2 & !s3 :
    s1 := false ; s2 := true ; s3:= true
!s1 & s2 & s3 :
    s1 := true ; s2 := false ; s3:= false
s1 & !s2 & !s3 :
    s1 := false ; s2 := false ; s3:= false
```

Question 2-

- The CTL properties used are:
  - AF (s1=F ∧ s2 = F ∧ s3 = F) → True
    - From every path, the counter will return to the initial state.
  - EF (s1=F ∧ s2 = F ∧ s3 = F) → True
    - If it works for AF, it works for EF.
  - AX (s1=F ∧ s2 = F ∧ s3 = T) → True
    - (The next of initial state is 1)
  - AX (s1=F ∧ s2 = T ∧ s3 = T) → False
    - (The next of initial state is not 2)
  - AG (Any state) → False
    - (Is a counter, can't be always one single state)

Question 3-

- We only need to add an “or” in our init for s3.

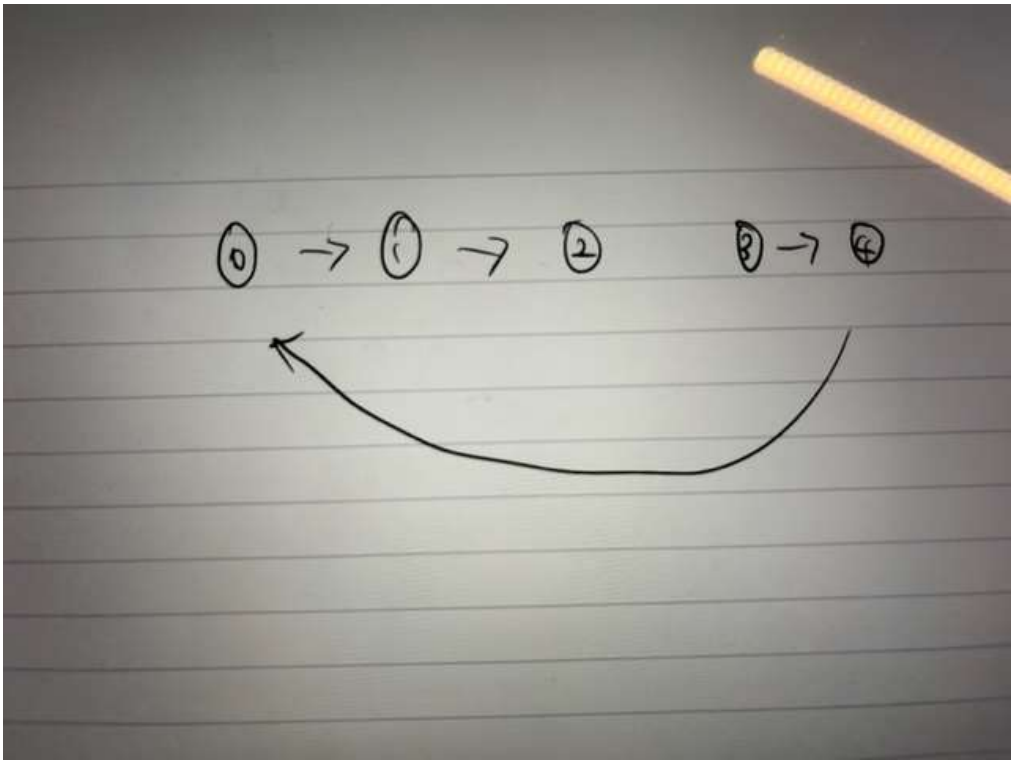
```
NAME counter
VAR
    s1 : boolean;
    s2 : boolean;
    s3 : boolean;


INIT
    !s1 & !s2 & (!s3 | s3);

RULES

!s1 & !s2 & !s3 :
    s1 := false ; s2 := false ; s3:= true
!s1 & !s2 & s3 :
    s1 := false ; s2 := true ; s3:= false
!s1 & s2 & !s3 :
    s1 := false ; s2 := true ; s3:= true
!s1 & s2 & s3 :
    s1 := true ; s2 := false ; s3:= false
s1 & !s2 & !s3 :
    s1 := false ; s2 := false ; s3:= false
```

Question 4-

- The properties used are:
  - (s1=F ∧ s2 = F ∧ s3 = F) | (s1=F ∧ s2 = F ∧ s3 = T) → T
    - Check if both initial states exist at the same time.
  - EX ((s1=F ∧ s2 = F ∧ s3 = F) | (s1=F ∧ s2 = F ∧ s3 = T)) → T
    - Check if the next stage of both initial states is correct.
  - EX ((s1=F ∧ s2 = F ∧ s3 = F) | (s1=F ∧ s2 = T ∧ s3 = T)) → F
    - Check to be sure It should be only the correct initial states.

Question 5 and 6-

- The bug produced is by extracting one of the rules (in this case the transition between 2-3), which means that our counter now should be like this:



- However, if we try using our CTL Properties from question 2, it seems that it still works fine! Which is something that shouldn't happen, since there's no way you get from 2 to 3, and so AF (s1 = False & s2 = True & s3 = True) shouldn't be true, neither should AF (s1 = False & s2 = False & s3 = False) since we cannot come back to state 0 or 1 since there's no transition between 2 to 3.

Model Checking: AF(!s1 & s2 & s3)
AF ¬s1 ∧ s2 ∧ s3
Done in: *0.0s*
Result is: *T*

- After many tries, I conclude that this is due to if I don't write the specific transition between 2 to 3, when we arrive at 2, the next stage can be anything, including 3, 4 or 5, or getting back to the initial states.

- To fix this, we should get back the original transition rule back!

KRIPKE STRUCTURE