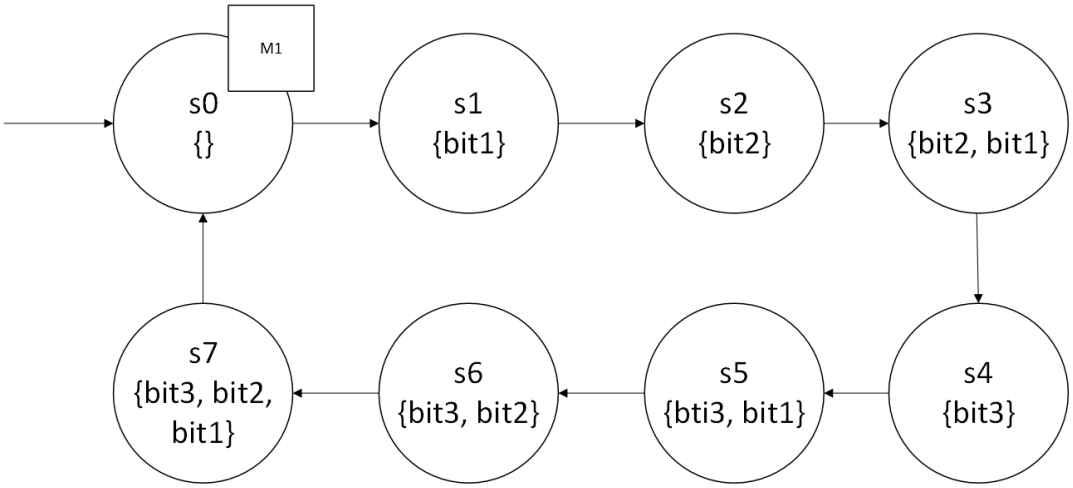
6CCS3VER 2024-25

Name: Yangan Yagol Xu Chen

K number: 24086575

- Problem 1

To minimize the number of variables, we can use 3 bool bits. Numbers from 0 to 7 can be represented with these 3 bits. The model starts from 0 going to 7 and then wrapping around to 0 again.



The table below shows the bits for each node. Each node s0 to s7 has 3 binary bits. When converted to the decimal equivalent, the “Decimal Value” column shows its decimal value.

Node	Bit 3	Bit 2	Bit 1	Decimal Value
s0	0	0	0	0
s1	0	0	1	1
s2	0	1	0	2
s3	0	1	1	3
s4	1	0	0	4
s5	1	0	1	5
s6	1	1	0	6
s7	1	1	1	7

- Problem 2

1) AG ((bit1 ->AX(!bit1)) | (bit2 -> AX(!bit2)) | (bit3 -> AX(!bit3)))

Safety property. Assure that when we have 111 the next state will be 000.

2) AG((bit1 & bit2 & bit3) -> AX(!bit1 & !bit2 & !bit3))

Safety property. This property shows that after 7, the next state is 0. This enforces the modulo operation on 8 making the count go back to 0 ensuring that the count does not exceed the maximum value.

3) AG((!bit1 & !bit2 & !bit3) -> AF(bit1 & bit2 & bit3))

Safety property. When having 000 eventually, it will go up to 111.

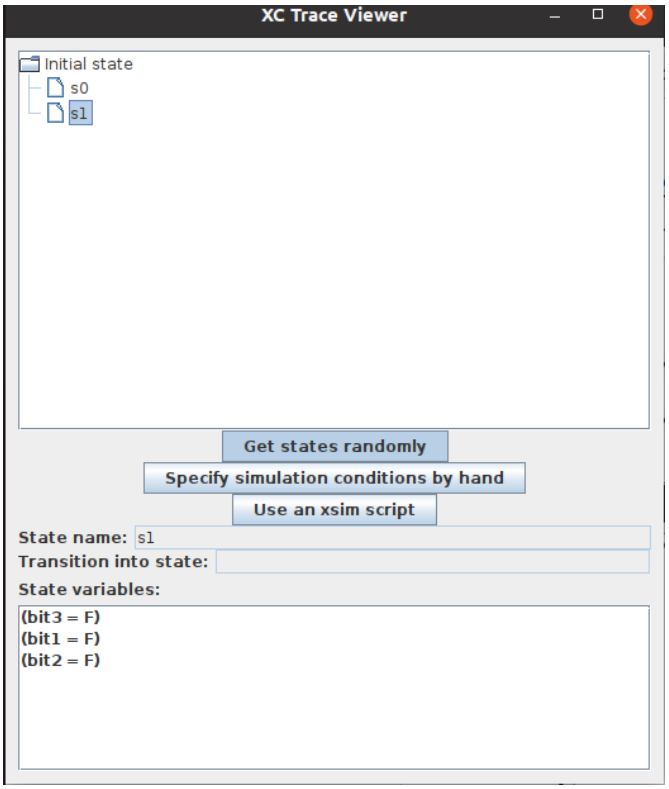
4) AG ((!bit3 & !bit2 & bit1) -> AX(!bit3 & bit2 & !bit1))

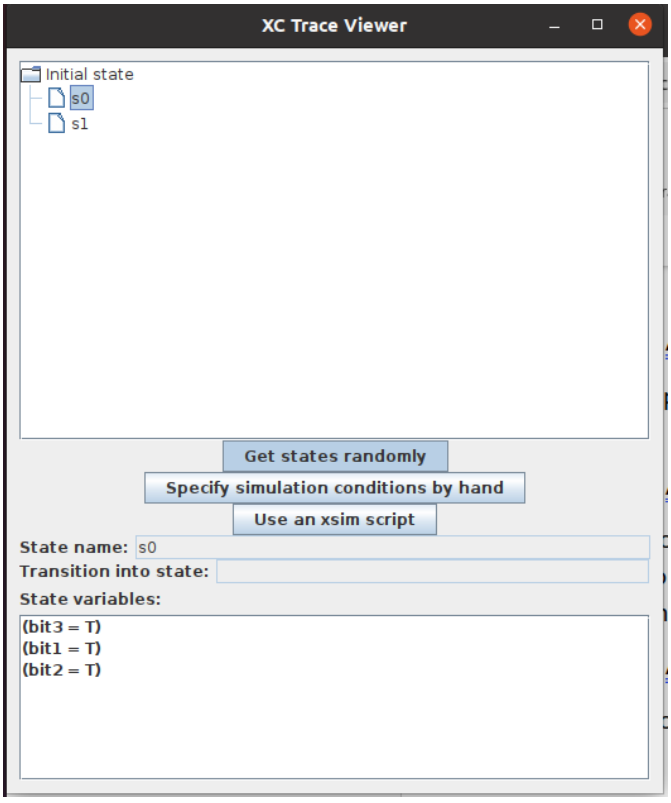
Safety property. The safety property defines that the behavior will always happen. In this case, the value goes from 1 to 2.

- Problem 3

To create a new model M2, we will create another state in the initialization block. In this case, the disjunction symbol is used to create another initial state.

```
INIT
 (!bit1 & !bit2 & !bit3) | (bit1 & bit2 & bit3);  -- Initial states (0 or 7)
```


XC Trace Viewer
Initial state
s0
s1
Get states randomly
Specify simulation conditions by hand
Use an xsim script
State name: s1
Transition into state:
State variables:
(bit3 = F)
(bit1 = F)
(bit2 = F)


XC Trace Viewer
Initial state
s0
s1
Get states randomly
Specify simulation conditions by hand
Use an xsim script
State name: s0
Transition into state:
State variables:
(bit3 = T)
(bit1 = T)
(bit2 = T)

- Problem 4

1) AF(¬bit1∧¬bit2∧¬bit3)

Model Checking: AF(¬bit1∧¬bit2∧¬bit3)
AF ¬bit1 ∧ ¬bit2 ∧ ¬bit3
Done in: *0.001s*
Result is: *T*

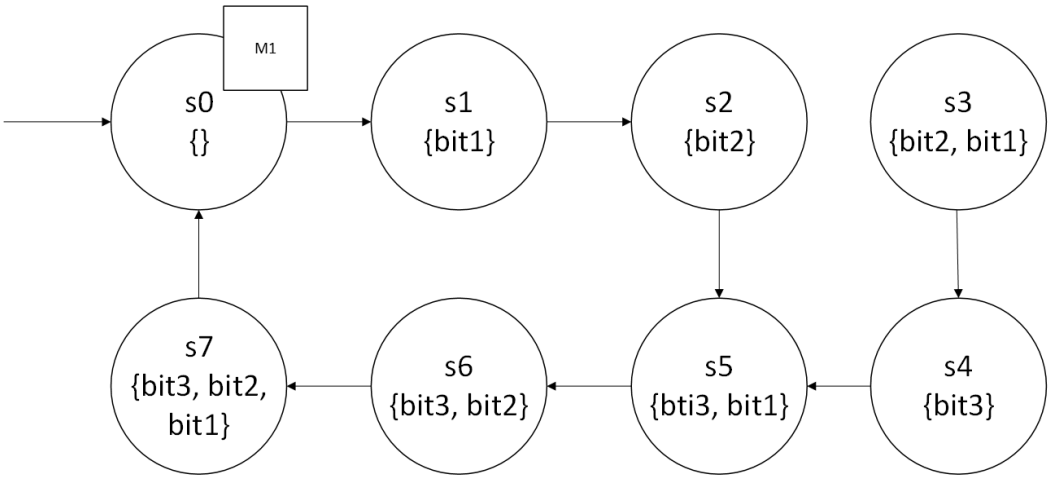
Prove that there exists a path from the initial state 7 to 0.

2) AF(bit1∧bit2∧bit3)

Model Checking: AF(bit1∧bit2∧bit3)
AF bit1 ∧ bit2 ∧ bit3
Done in: *0.001s*
Result is: *T*

Check that exists a path from the initial state 0 to 7.

- Problem 5



To introduce a bug not found in any of the properties, the s2 will point to s5. This makes the M1 model not capture the complete count of values. On the initial state the new M3 model goes from 0 to 2, then jumps to 5. If this bug occurs, the model does not jump in every state skipping the s3 and s4.

We can prove this with the following property: (returns false)

AG ((!bit3 & bit2 & !bit3) -> AX(!bit3 & bit2 & bit3))

- Problem 6

Our current properties do not check if all the bits have been seen. We can check if from the initial state we can go to all the other states. For all the paths eventually it will go through all the bits. With this we ensure that all the bits will be seen. If any of the pointers are skipped, the following CTL will return false:

AG ((!bit1 & !bit2 & !bit3) ->

(

AF (!bit1 & !bit2 & bit3) &

AF (!bit1 & bit2 & !bit3) &

AF (!bit1 & bit2 & bit3) &

AF (bit1 & !bit2 & !bit3) &

AF (bit1 & !bit2 & bit3) &

AF (bit1 & bit2 & !bit3) &

AF (bit1 & bit2 & bit3)

)

)

An alternative could also be to explicitly write the commands to go in order from 0 to 7 and then wrap around as shown in the exercise 2, 4th CTL.

1. AG ((!bit3 & !bit2 & !bit3) -> AX(!bit3 & !bit2 & bit3))

2. AG ((!bit3 & !bit2 & bit3) -> AX(!bit3 & bit2 & !bit3))

3. AG ((!bit3 & bit2 & !bit3) -> AX(!bit3 & bit2 & bit3))

4. AG ((!bit3 & bit2 & bit3) -> AX(bit3 & !bit2 & !bit3))

5. AG ((bit3 & !bit2 & !bit3) -> AX(bit3 & !bit2 & bit3))

6. AG ((bit3 & !bit2 & bit3) -> AX(bit3 & bit2 & !bit3))

7. AG ((bit3 & bit2 & !bit3) -> AX(bit3 & bit2 & bit3))

8. AG ((bit3 & bit2 & bit3) ->AX(!bit3 & !bit2 & !bit3))