DD2459 Software Reliability

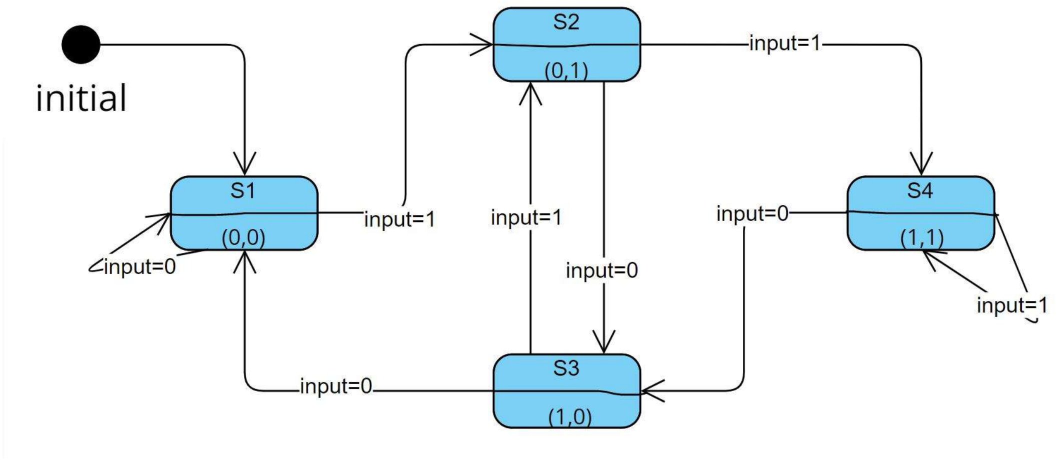
# Lab 3

# Model-based Testing and Automated Test-case Generation

Yuze Cui
Yuzec@kth.se

Rui Shi
Srui@kth.se

1. UML statecharts for the 2-bit shift register FSM:



2. For the bitshift.smv, According to statechart, complete the definitions:
   We can find that the output is defined as (Bit2,Bit1)

```
next( Bit2 ) := Bit1;

next( Bit1 ) := input;
```

   The output of the NuSMV:

```
-- no counterexample found with bound 0
-- no counterexample found with bound 1
-- no counterexample found with bound 2
-- no counterexample found with bound 3
-- no counterexample found with bound 4
-- no counterexample found with bound 5
-- no counterexample found with bound 6
-- no counterexample found with bound 7
-- no counterexample found with bound 8
-- no counterexample found with bound 9
-- no counterexample found with bound 10
-- no counterexample found with bound 0
-- specification  G (Bit1 <->  X Bit1)   is false
-- as demonstrated by the following execution sequence
Trace Description: BMC Counterexample
Trace Type: Counterexample
  -> State: 1.1 <-
    Bit1 = FALSE
    Bit2 = FALSE
    state = s1
  -> Input: 1.2 <-
    input = TRUE
  -> State: 1.2 <-
```

```
Bit1 = TRUE
state = s2
```

We can conclude from the test:

a) For Bit2-> X Bit1
   The test traverse all bounds but doesn't find a counterexample of the first trap property, which agrees with the definition.
b) For Bit1-> X Bit1
   The test can generate such output as counterexamples:

| test | State | input | output | Correctness of the test |
|---|---|---|---|---|
| 1 | S1 (0,0) | 1 | S2 (0,1) | False |

   For the trap property, the output Bit1=1 doesn't equal the Bit1 in previous, so the second definition of the trap property is wrong.

3. For the car-controller
   a) From the report from NuSMV tool:

```
  -- specification  G !(state = stop)    is false
  -- as demonstrated by the following execution sequence
  Trace Description: BMC Counterexample
  Trace Type: Counterexample
    -> State: 1.1 <-
      state = stop
  -- no counterexample found with bound 0
  -- specification  G ((state = stop & accelerate) ->  X !(state
= slow))    is false
  -- as demonstrated by the following execution sequence
  Trace Description: BMC Counterexample
  Trace Type: Counterexample
    -> State: 2.1 <-
      state = stop
    -> Input: 2.2 <-
      accelerate = TRUE
      brake = FALSE
    -> State: 2.2 <-
  state = slow
```

We can find the counterexample:

| Test | Trap Property | counterexample |
|---|---|---|
| 1 | G!(state=stop) | State=stop |

b) TR: state=stop

| | T=0 | T=1 | T=2 | T=3 |
|---|---|---|---|---|
| Inputs | / | | | |
| outputs | stop | | | |

c) 2 additional examples are described below:

```
LTLSPEC
   G( !(state = slow))
LTLSPEC
   G( !(state = fast))
```

We can get the report below:

```
-- specification  G !(state = stop)    is false
-- as demonstrated by the following execution sequence
Trace Description: BMC Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
 state = stop
-- no counterexample found with bound 0
-- specification  G !(state = slow)    is false
-- as demonstrated by the following execution sequence
Trace Description: BMC Counterexample
Trace Type: Counterexample
-> State: 2.1 <-
state = stop
-> Input: 2.2 <-
    accelerate = TRUE
    brake = FALSE
  -> State: 2.2 <-
    state = slow
-- no counterexample found with bound 0
-- no counterexample found with bound 1
-- specification  G !(state = fast)    is false
-- as demonstrated by the following execution sequence
Trace Description: BMC Counterexample
Trace Type: Counterexample
  -> State: 3.1 <-
    state = stop
  -> Input: 3.2 <-
    accelerate = TRUE
    brake = FALSE
  -> State: 3.2 <-
    state = slow
  -> Input: 3.3 <-
  -> State: 3.3 <-
    state = fast
-- no counterexample found with bound 0
-- specification  G ((state = stop & accelerate) ->  X !(state =
slow))    is false
-- as demonstrated by the following execution sequence
Trace Description: BMC Counterexample
Trace Type: Counterexample
  -> State: 4.1 <-
    state = stop
  -> Input: 4.2 <-
    accelerate = TRUE
    brake = FALSE
  -> State: 4.2 <-
    state = slow
```

For TR: state=slow

| c | T=0 | T=1 | T=2 | T=3 |
|---|---|---|---|---|
| Inputs | Accelerate & !brake | / | | |
| outputs | stop | slow | | |

For TR: state=fast

| | T=0 | T=1 | T=2 | T=3 |
|---|---|---|---|---|
| Inputs | Accelerate & !brake | Accelerate & !brake | / | |
| outputs | stop | slow | fast | |

4. Edge coverage:
   a) Run the carcontroller.smv, we have the output:

```
    -- no counterexample found with bound 0
    -- specification  G ((state = stop & accelerate) ->  X !(state
= slow))    is false
    -- as demonstrated by the following execution sequence
    Trace Description: BMC Counterexample
    Trace Type: Counterexample
      -> State: 4.1 <-
        state = stop
      -> Input: 4.2 <-
        accelerate = TRUE
        brake = FALSE
      -> State: 4.2 <-
    state = slow
```

| Test | Trap Property | counterexample |
|---|---|---|
| 1 | (state = stop & accelerate) -> X !(state = slow) | (state = stop)->(accelerate&!brake) -> X !(state = slow) |

b) TR: (state = stop & accelerate) -> X (state = slow)

| | T=0 | T=1 | T=2 | T=3 |
|---|---|---|---|---|
| Inputs | Accelerate & !brake | \ | | |
| outputs | stop | slow | | |

c) Writing the five trap properties:

```
LTLSPEC
    G( state=slow & (accelerate & !brake)-> X( !(state=fast) ) )

LTLSPEC
    G( state=slow & !accelerate & !brake -> X( !(state=stop) ) )

LTLSPEC
    G( state=slow & brake -> X( !(state=stop) ) )

LTLSPEC
    G( state=fast & (!accelerate & !brake)-> X( !(state=slow) ) )

LTLSPEC
    G( state=fast & brake -> X( !(state=stop) ) )
```

The output:

```
    -- no counterexample found with bound 0
    -- specification  G ((state = stop & accelerate) ->  X !(state
= slow))    is false
    -- as demonstrated by the following execution sequence
    Trace Description: BMC Counterexample
    Trace Type: Counterexample
      -> State: 4.1 <-
        state = stop
      -> Input: 4.2 <-
        accelerate = TRUE
        brake = FALSE
      -> State: 4.2 <-
        state = slow
    -- no counterexample found with bound 0
    -- no counterexample found with bound 1
    -- specification  G ((state = slow & (accelerate & !brake)) ->
X !(state = fast))    is false
```

```
    -- as demonstrated by the following execution sequence
    Trace Description: BMC Counterexample
    Trace Type: Counterexample
      -> State: 5.1 <-
        state = stop
      -> Input: 5.2 <-
        accelerate = TRUE
        brake = FALSE
      -> State: 5.2 <-
        state = slow
      -> Input: 5.3 <-
      -> State: 5.3 <-
        state = fast
    -- no counterexample found with bound 0
    -- no counterexample found with bound 1
    -- specification  G (((state = slow & !accelerate) & !brake) ->
X !(state = stop))    is false
    -- as demonstrated by the following execution sequence
    Trace Description: BMC Counterexample
    Trace Type: Counterexample
      -- Loop starts here
      -> State: 6.1 <-
        state = stop
      -> Input: 6.2 <-
        accelerate = TRUE
        brake = FALSE
      -> State: 6.2 <-
        state = slow
      -> Input: 6.3 <-
        accelerate = FALSE
      -> State: 6.3 <-
        state = stop
    -- no counterexample found with bound 0
    -- no counterexample found with bound 1
    -- specification  G ((state = slow & brake) ->  X !(state =
stop))    is false
    -- as demonstrated by the following execution sequence
    Trace Description: BMC Counterexample
    Trace Type: Counterexample
      -- Loop starts here
      -> State: 7.1 <-
        state = stop
      -> Input: 7.2 <-
        accelerate = TRUE
        brake = FALSE
      -> State: 7.2 <-
        state = slow
      -> Input: 7.3 <-
        accelerate = FALSE
        brake = TRUE
      -> State: 7.3 <-
        state = stop
    -- no counterexample found with bound 0
    -- no counterexample found with bound 1
    -- no counterexample found with bound 2
    -- specification  G ((state = fast & (!accelerate & !brake)) ->
X !(state = slow))    is false
    -- as demonstrated by the following execution sequence
    Trace Description: BMC Counterexample
    Trace Type: Counterexample
      -> State: 8.1 <-
        state = stop
      -> Input: 8.2 <-
        accelerate = TRUE
        brake = FALSE
      -- Loop starts here
      -> State: 8.2 <-
        state = slow
      -> Input: 8.3 <-
      -> State: 8.3 <-
        state = fast
      -> Input: 8.4 <-
        accelerate = FALSE
      -> State: 8.4 <-
        state = slow
    -- no counterexample found with bound 0
    -- no counterexample found with bound 1
    -- no counterexample found with bound 2
    -- specification  G ((state = fast & brake) ->  X !(state =
stop))    is false
    -- as demonstrated by the following execution sequence
    Trace Description: BMC Counterexample
    Trace Type: Counterexample
      -- Loop starts here
      -> State: 9.1 <-
        state = stop
      -> Input: 9.2 <-
        accelerate = TRUE
        brake = FALSE
      -> State: 9.2 <-
        state = slow
      -> Input: 9.3 <-
      -> State: 9.3 <-
        state = fast
      -> Input: 9.4 <-
        accelerate = FALSE
        brake = TRUE
      -> State: 9.4 <-
        state = stop
```

The five additional trap properties and the test cases:

1) G( state=slow & (accelerate & !brake)-> X( !(state=fast) ) )

| | T=0 | T=1 | T=2 | T=3 |
|---|---|---|---|---|
| Inputs | accelerate & !brake | accelerate & !brake | / | |
| outputs | stop | slow | fast | |

2) G( state=slow & !accelerate & !brake -> X( !(state=stop) ) )

| | T=0 | T=1 | T=2 | T=3 |
|---|---|---|---|---|
| Inputs | accelerate & !brake | !accelerate & !brake | / | |
| outputs | stop | slow | stop | |

3) G( state=slow & brake -> X( !(state=stop) ) )

| | T=0 | T=1 | T=2 | T=3 |
|---|---|---|---|---|
| Inputs | accelerate & !brake | !accelerate & brake | / | |
| outputs | stop | slow | stop | |

4) G( state=fast & (!accelerate & !brake)-> X( !(state=slow) ) )

| | T=0 | T=1 | T=2 | T=3 |
|---|---|---|---|---|
| Inputs | accelerate & !brake | accelerate & !brake | !accelerate & !brake | |
| outputs | stop | slow | fast | slow |

5) G( state=fast & brake -> X( !(state=stop) ) )

| | T=0 | T=1 | T=2 | T=3 |
|---|---|---|---|---|
| Inputs | accelerate & !brake | accelerate & !brake | !accelerate & brake | / |
| outputs | stop | slow | fast | stop |