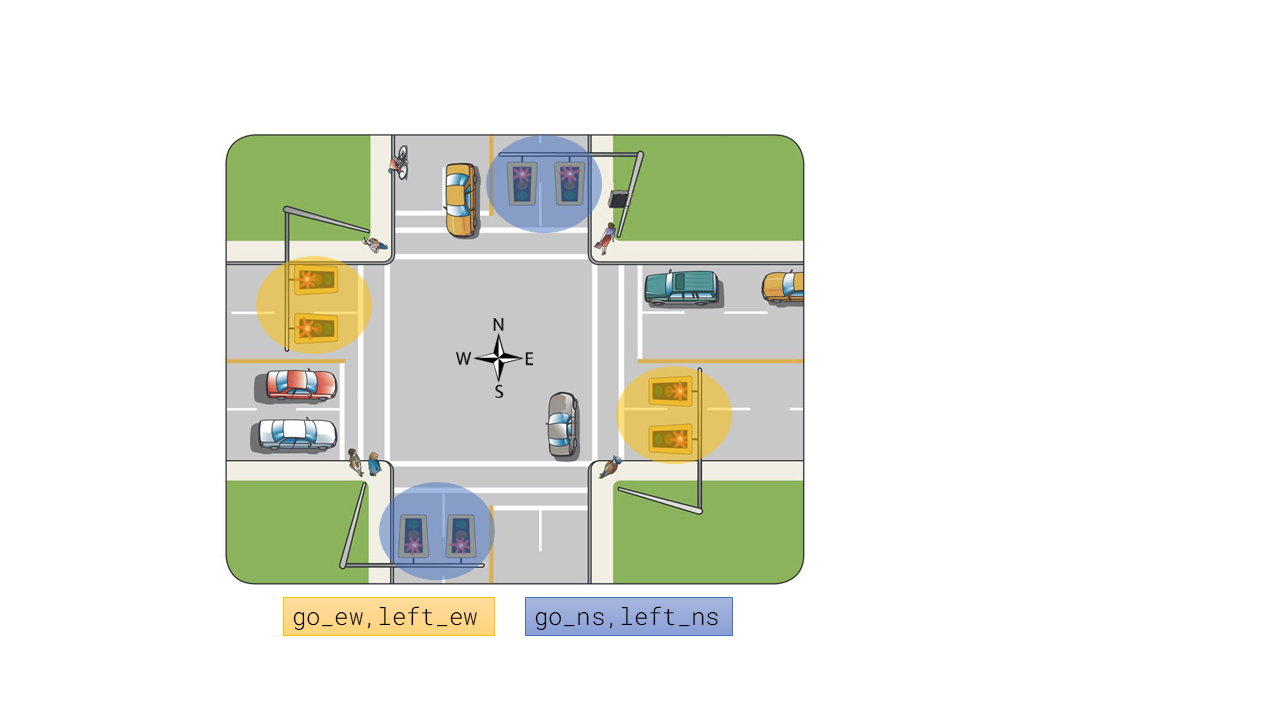
소프트웨어테스팅이론 ITEC414001

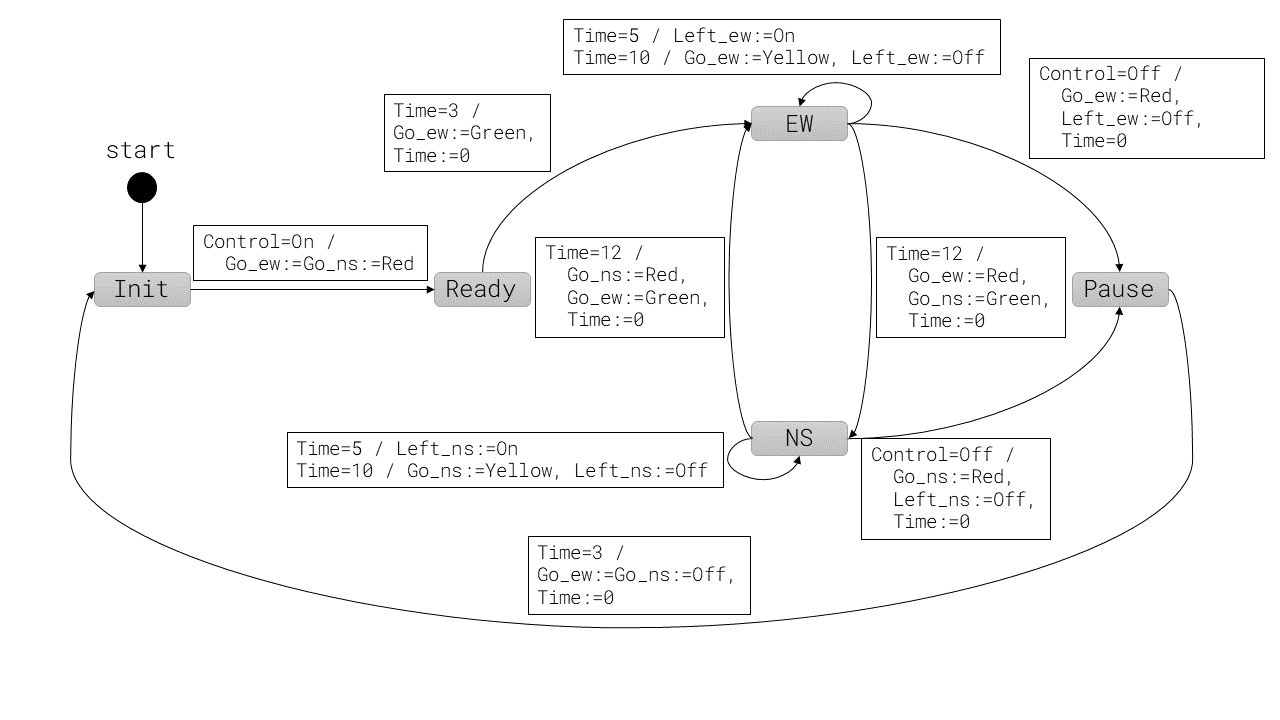
Model Based Test Generation

2016112905 김민섭

# **1. Finite State Machine Model**



* North-South 방향의 signal board를 go\_ew(직진), left\_ew(좌회전)로 두었습니다.
* East-West 방향의 signal board도 마찬가지로 go\_ns, left\_ns로 두었습니다.



* 초기 state는 Init 입니다. 여기서 On 신호를 받으면 Ready로 이동합니다.
* EW, NS는 각각 East-West, North-West 방향의 signal board가 초록불인 상태입니다.
* EW, NS는 Time 값에 따라 같은 상태 내에서 go\_xx를 초록불에서 노란불, 빨간불로 변경합니다. left\_xx 또한 마찬가지입니다. go와 left는 다른 이벤트에 영향을 받지 않으므로 상태를 세분화하지 않았습니다.
* EW, NS에서 Off 신호를 받으면 Pause로 이동합니다. 다른 상태에서는 Pause에 도달할 수 없습니다.
* Pause에서 Time=3이 되면 Init으로 이동합니다. 신호 시스템이 완전히 꺼진 상태와 Init은 같습니다. 따라서 제가 설계한 FSM 모델은 종료가 필요하지 않습니다.

# **2. Test Cases and Coverage Analysis Report**

|  |  |
| --- | --- |
| Specification 코드 | 콘솔 결과 |
| SPEC EF (state = Init)  SPEC EF (state = Ready)  SPEC EF (state = EW)  SPEC EF (state = NS)  SPEC EF (state = Pause) | NuSMV > check\_ctlspec  -- specification EF state = Init is true  -- specification EF state = Ready is true  -- specification EF state = EW is true  -- specification EF state = NS is true  -- specification EF state = Pause is true |

* CTL specification EF를 사용하여 모든 state에 도달 가능함을 확인해 보았습니다.

## **2.1. Execution Sequence: state = Ready**

|  |  |
| --- | --- |
| **Specification 코드** | LTLSPEC G ! (state = Init) |
| **콘솔 결과** | -- Loop starts here  -> State: 2.1 <-  **state = Init**  control = None  go\_ew = Off  left\_ew = Off  go\_ns = Off  left\_ns = Off  time = 0  -> State: 2.2 <-  **control = On**  -> State: 2.3 <-  **state = Ready**  control = None  go\_ew = Red  go\_ns = Red  -> State: 2.4 <-  time = 1  -> State: 2.5 <-  time = 2  -> State: 2.6 <-  state = EW  control = Off  go\_ew = Green  time = 3  -> State: 2.7 <-  state = Pause  control = None  go\_ew = Red  time = 0  -> State: 2.8 <-  time = 1  -> State: 2.9 <-  time = 2  -> State: 2.10 <-  state = Init  go\_ew = Off  go\_ns = Off  time = 3  -> State: 2.11 <-  time = 0 |

* state = Init 상태에서 control = On을 받으면 state = Ready가 됨을 확인할 수 있었습니다.

## **2.2. Execution Sequence: state = EW**

|  |  |
| --- | --- |
| **Specification 코드** | LTLSPEC G ! (state = EW) |
| **콘솔 결과** | -- Loop starts here  -> State: 3.1 <-  state = Init  control = None  go\_ew = Off  left\_ew = Off  go\_ns = Off  left\_ns = Off  time = 0  -> State: 3.2 <-  control = On  -> State: 3.3 <-  **state = Ready**  control = None  go\_ew = Red  go\_ns = Red  -> State: 3.4 <-  time = 1  -> State: 3.5 <-  time = 2  -> State: 3.6 <-  **state = EW**  go\_ew = Green  **time = 3**  -> State: 3.7 <-  control = Off  time = 4  -> State: 3.8 <-  state = Pause  control = None  go\_ew = Red  time = 0  -> State: 3.9 <-  time = 1  -> State: 3.10 <-  time = 2  -> State: 3.11 <-  state = Init  go\_ew = Off  go\_ns = Off  time = 3  -> State: 3.12 <-  time = 0 |

* state = Ready 에서 time = 3일 때 state = EW가 됨을 확인할 수 있었습니다.

## **2.3. Execution Sequence: state = NS**

|  |  |
| --- | --- |
| **Specification 코드** | LTLSPEC G ! (state = NS) |
| **콘솔 결과** | -- Loop starts here  -> State: 4.1 <-  state = Init  control = None  go\_ew = Off  left\_ew = Off  go\_ns = Off  left\_ns = Off  time = 0  s\_running = EW, NS  t\_bef\_pause = 2  t\_bef\_switch = 11  t\_bef\_left = 4  t\_bef\_yellow = 9  t\_bef\_ready = 2  -> State: 4.2 <-  control = On  -> State: 4.3 <-  state = Ready  control = None  go\_ew = Red  go\_ns = Red  -> State: 4.4 <-  time = 1  -> State: 4.5 <-  time = 2  -> State: 4.6 <-  **state = EW**  control = On  go\_ew = Green  time = 3  -> State: 4.7 <-  time = 4  -> State: 4.8 <-  left\_ew = On  time = 5  -> State: 4.9 <-  time = 6  -> State: 4.10 <-  time = 7  -> State: 4.11 <-  time = 8  -> State: 4.12 <-  time = 9  -> State: 4.13 <-  go\_ew = Yellow  left\_ew = Off  time = 10  -> State: 4.14 <-  time = 11  -> State: 4.15 <-  **state = NS**  control = None  go\_ew = Red  **time = 12**  -> State: 4.16 <-  control = Off  time = 0  -> State: 4.17 <-  state = Pause  control = None  -> State: 4.18 <-  time = 1  -> State: 4.19 <-  time = 2  -> State: 4.20 <-  state = Init  go\_ew = Off  go\_ns = Off  time = 3  -> State: 4.21 <-  time = 0 |

* state = EW 에서 time = 12 일 때 state = NS 가 됨을 확인할 수 있었습니다.

## **2.4. Execution Sequence: state = Pause**

|  |  |
| --- | --- |
| **Specification 코드** | LTLSPEC G ! (state = Pause) |
| **콘솔 결과** | -- Loop starts here  -> State: 5.1 <-  state = Init  control = None  go\_ew = Off  left\_ew = Off  go\_ns = Off  left\_ns = Off  time = 0  s\_running = EW, NS  t\_bef\_pause = 2  t\_bef\_switch = 11  t\_bef\_left = 4  t\_bef\_yellow = 9  t\_bef\_ready = 2  -> State: 5.2 <-  control = On  -> State: 5.3 <-  state = Ready  control = None  go\_ew = Red  go\_ns = Red  -> State: 5.4 <-  time = 1  -> State: 5.5 <-  time = 2  -> State: 5.6 <-  **state = EW**  **control = Off**  go\_ew = Green  time = 3  -> State: 5.7 <-  **state = Pause**  control = None  go\_ew = Red  time = 0  -> State: 5.8 <-  time = 1  -> State: 5.9 <-  time = 2  -> State: 5.10 <-  state = Init  go\_ew = Off  go\_ns = Off  time = 3  -> State: 5.11 <-  time = 0 |

* state = EW 에서 control = Off 일 때 state = Pause 가 됨을 확인할 수 있었습니다.

## **2.5. Execution Sequence with loop: state = Init**

|  |  |
| --- | --- |
| **Specification 코드** | LTLSPEC G ! (state != Init & X (state = Init)) |
| **콘솔 결과** | -> State: 1.1 <-  state = Init  control = None  go\_ew = Off  left\_ew = Off  go\_ns = Off  left\_ns = Off  time = 0  s\_running = EW, NS  t\_bef\_pause = 2  t\_bef\_switch = 11  t\_bef\_left = 4  t\_bef\_yellow = 9  t\_bef\_ready = 2  -> State: 1.2 <-  control = On  -> State: 1.3 <-  state = Ready  control = None  go\_ew = Red  go\_ns = Red  -> State: 1.4 <-  time = 1  -> State: 1.5 <-  time = 2  -> State: 1.6 <-  state = EW  control = Off  go\_ew = Green  time = 3  -> State: 1.7 <-  **state = Pause**  control = None  go\_ew = Red  time = 0  -> State: 1.8 <-  time = 1  -> State: 1.9 <-  time = 2  -> State: 1.10 <-  **state = Init**  go\_ew = Off  go\_ns = Off  **time = 3**  -- Loop starts here  -> State: 1.11 <-  time = 0  -> State: 1.12 <- |

* 초기 상태가 Init이기 때문에 초기상태가 아닌 Init에 도달하는 경로를 찾아보았습니다.
* state = Pause 에서 time = 3 일 때 state = Init 이 됨을 확인할 수 있었습니다.

# **3. Optional Specification**

|  |  |
| --- | --- |
| **Specification 코드** | LTLSPEC G ! (go\_ew = Green & go\_ns = Green) |
| **콘솔 결과** | \*\*\*\* PROPERTY LIST [ Type, Status, Counter-example Number, Name ] \*\*\*\*  -------------------------- PROPERTY LIST -------------------------  000 : G !(go\_ew = Green & go\_ns = Green)  [LTL Unchecked N/A N/A]  NuSMV > check\_ltlspec  -- specification G !(go\_ew = Green & go\_ns = Green) is **true** |

* 모든 Execution Path에서 NS green과 EW green 이 동시에 켜진 상태가 없음을 확인했습니다.

Traffic signal 이미지 출처: <https://www.tranbc.ca/2016/03/17/traffic-signal-power-outage-what-do-you-do/>