SPIN

Simple Promela Interpreter

고급 소프트웨어공학

이종원, 이상진

2017.04.03

목차

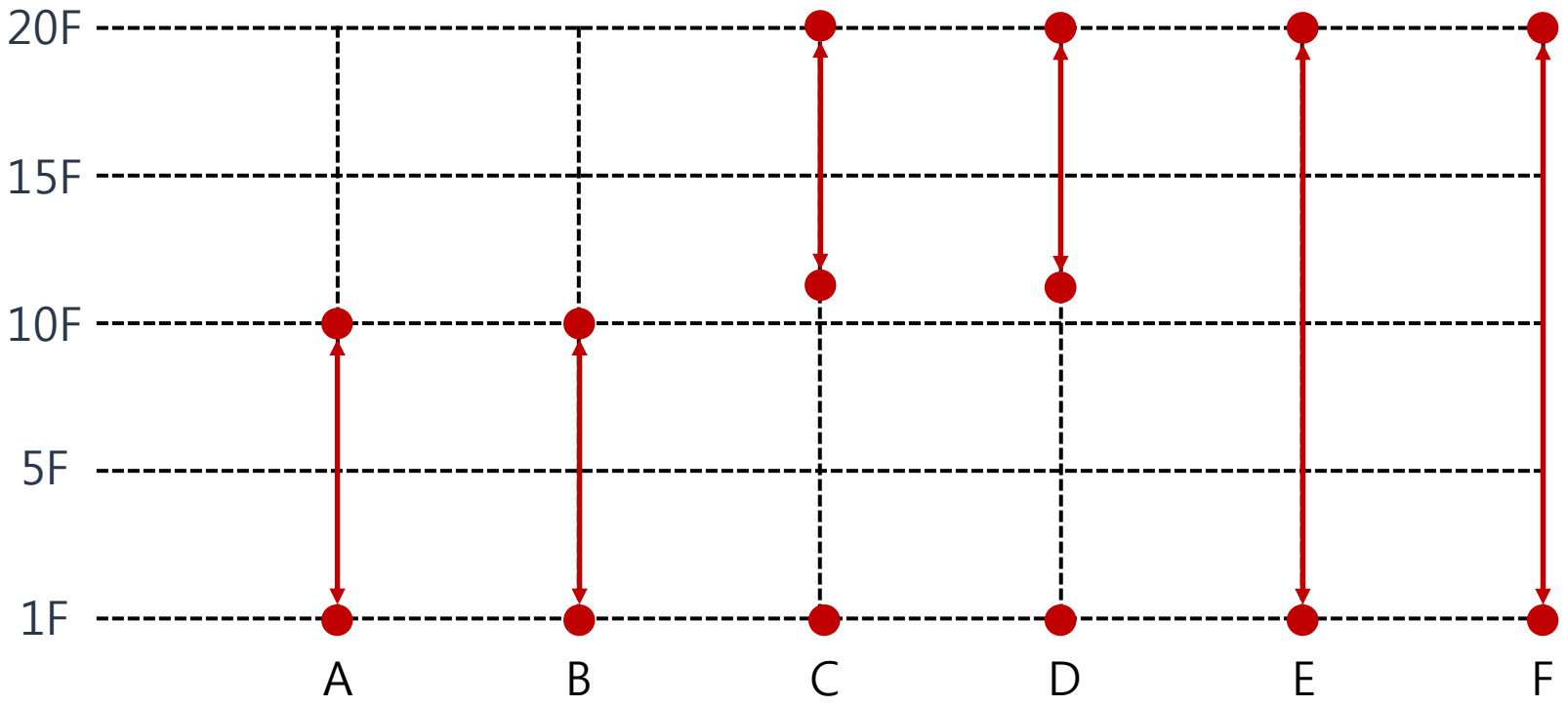
01

엘리베이터 설계

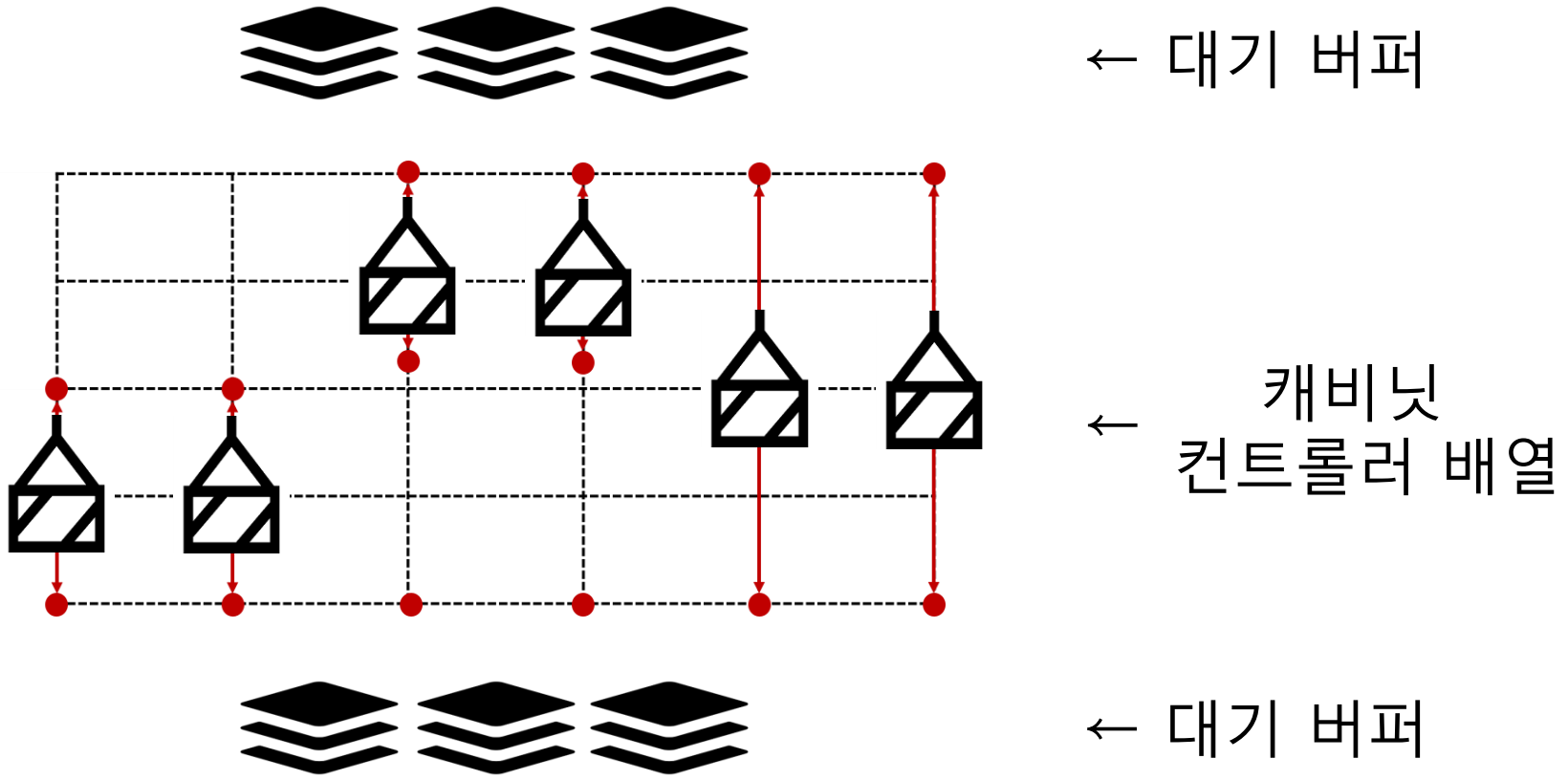
01. 엘리베이터 설계

- 엘리베이터의 층
- 엘리베이터의 캐비닛
- 엘리베이터의 컨트롤러
- 엘리베이터의 설계 고려사항

- **엘리베이터의 캐비닛:**
  - ✓ 저층용(1F~7F) × 2
  - ✓ 고층용(1F, 8F~14F) × 2
  - ✓ 전층용(1F~14F) × 2



- **엘리베이터의 설계 고려사항:**
  - ✓ 대기 버퍼: (저층, 고층, 전층) × (위,아래) = 3 × 2개
  - ✓ 각 캐비닛 컨트롤러 마다의 배열, 6개



- **엘리베이터의 설계 고려사항:**
  - ✓ 엘리베이터 동작
    1. 사용자의 방향(↑・↓)을 판단
    2. 사용자가 어떤 캐비닛(저・고・전층용)을 타야 하는지를 판단
    3. (2.)에서 선택된 캐비닛 중, 정지 되어있는 캐비닛이 있다면 동작
    4. 없다면, (2.)에서 선택된 캐비닛 들의 방향(↑・↓)을 판단
    5. 캐비닛의 방향(3.)과 사용자의 방향(2.)의 일치 여부 판단

- **엘리베이터의 설계 고려사항:**

✓ 엘리베이터 동작

1. 사용자의 방향(↑·↓)을 판단

❖ selectUser: (currentUserFloor, destinationUserFloor)

➢ 사용자의 현재 층: (int) currentUserFloor

➢ 사용자의 목적지 층: (int) destinationUserFloor

A. selectUser:

currentFloorUser – destinationUserFloor > 0 // ↓

currentFloorUser – destinationUserFloor < 0 // ↑

currentFloorUser – destinationUserFloor = 0 // 재입력 요구

- selectUser: (currentFloorUser, destinationUserFloor) // ((F1~F14), (F1~F14))

- **엘리베이터의 설계 고려사항:**

✓ 엘리베이터 동작

2. 사용자가 버튼을 누르면 컨트롤러에게 전달되며,
   어떤 캐비닛(저·고 ·전층용)을 타야 하는지를 판단

❖ selectUser: (currentUserFloor, destinationUserFloor)

➢ 사용자의 현재 층: (int) currentUserFloor

➢ 사용자의 목적지 층: (int) destinationUserFloor

A. 현재층에서 현재층 버튼을 누를 시에는 재입력 알람

a. if(currentUserFloor = destinationUserFloor)
   ~>재입력

- 저층용: cabinetA, cabinetB
- 전층용: cabinetE, cabinetF
- 고층용: cabinetC, cabinetD

- **엘리베이터의 설계 고려사항:**

✓ 엘리베이터 동작

B. (1F, (2F~7F)): 저층 & 전층용

a. if(currentUserFloor=1 &&

2<=destinationUserFloor<=7)

~> cabinetA||cabinetB||cabinetE||cabinetF

C. (1F, (8F~14F)): 고층 & 전층용

a. if(currentUserFloor=1 &&

8<=FdestinationOfUserFloor<=14)

~> cabinetC||cabinetD||cabinetE||cabinetF

- **엘리베이터의 설계 고려사항:**

✓ 엘리베이터 동작

D. ((2F~7F), (1F~7F)): 저층 & 전층용

a. if(2<=currentUserFloor<=10 &&
1<=destinationUserFloor<=10)
~> cabinetA||cabinetB||cabinetE||cabinetF

E. ((2F~7F), (8F~14F)): 전층용

a. if(2<=currentUserFloor<=10 &&
11<=destinationUserFloor<=20)
~> cabinetE||cabinetF

- **엘리베이터의 설계 고려사항:**

✓ 엘리베이터 동작

F. ((8F~14F), (F1, 8F~14F)): 고층 & 전층용

a. if(8<=currentUserFloor<=14 &&
(1=destinationUserFloor ||
8<=destinationUserFloor<=14)
~> cabinetC||cabinetD||cabinetE||cabinetF

G. ((8F~14F), (F2~8F)): 전층용

a. if(8<=currentUserFloor<=14 &&
2<=destinationUserFloor<=7)
~> cabinetE||cabinetF

- **엘리베이터의 설계 고려사항:**

✓ 엘리베이터 동작

3. (2.)에서 선택된 캐비닛 중, 정지 되어있는 캐비닛이 있다면 동작

❖ Cabinet(A~F): (currentCabinetFloor, cabinetArray[6~8])

➢ 캐비닛의 현재 층: (int) currentCabinetFloor

➢ 캐비닛의 목적지 층:

(int) cabinetArray[6~8]={$destinationUserFloor_0$, … }

❖ destinationUserFloor에 도착 시,

destinationUserFloor = currentCabinetFloor 업데이트

- **엘리베이터의 설계 고려사항:**

✓ 엘리베이터 동작

A. 사용자의 층에 캐비닛이 있는 경우,

➢ cabinet(A||B||C||D||E||F) &&
currentCabinetFloor = currentFloorUser &&
cabinetArray[6~8]={NULL})
~>cabinetArray[6~8]={destinationUserFloor},
doorOpen-> close-> move

B. 사용자의 층에 캐비닛이 없는 경우,

➢ cabinet(A||B||C||D||E||F) && cabinetArray[4]={NULL}
~>cabinetArray[8]=
{currentFloorUser, destinationUserFloor}

- **엘리베이터의 설계 고려사항:**

✓ 엘리베이터 동작

4. 정지 되어있는 캐비닛이 없다면,

(2.)에서 선택된 캐비닛 들의 방향(↑ · ↓)을 판단 후 동작

A. 캐비닛의 방향↑, 사용자의 방향↑

a. 캐비닛이 사용자보다 아래층에 있는 경우

cabinet(A||B||C||D||E||F) &&

cabinetArray[0] – currentCabinetFloor > 0&&

destinationUserFloor – currentUserFloor > 0&&

currentUserFloor – currentCabinetFloor > 2

~>cabinetArray[6~8]에

currentFloorUser, destinationUserFloor 추가

- **엘리베이터의 설계 고려사항:**

✓ 엘리베이터 동작

A. 캐비닛의 방향↑, 사용자의 방향↑

b. 캐비닛이 사용자보다 위층에 있는 경우

cabinet(A||B||C||D||E||F) &&

cabinetArray[0] – currentCabinetFloor > 0&&

destinationUserFloor – currentUserFloor > 0&&

currentUserFloor – currentCabinetFloor < 0

~>upbuffer[]에

currentFloorUser, destinationUserFloor 추가

❖ 반대의 경우 기호만 반대!

- **엘리베이터의 설계 고려사항:**

✓ 엘리베이터 동작

A. 캐비닛의 방향↑, 사용자의 방향 ↓

a. cabinet(A||B||C||D||E||F) &&
cabinetArray[0] – currentCabinetFloor > 0&&
currentUserFloor – destinationUserFloor > 0&&
~>downBuffer[]에
currentFloorUser, destinationUserFloor 추가

❖ 반대의 경우, upBuffer[]에 추가

02

엘리베이터 모델링

- **메인 컨트롤러:**

✓ 버튼 입력 처리

```
active proctype maincontroller(){
        if                          /* 버튼 입력에 대한 처리*/
        ::button_1F!_1to2 -> run low_and_all(1,2);
        ::button_1F!_1to3 -> run low_and_all(1,3);
        ::button_1F!_1to4 -> run low_and_all(1,4);
        ::button_1F!_1to5 -> run low_and_all(1,5);
        ::button_1F!_1to6 -> run low_and_all(1,6);
        ::button_1F!_1to7 -> run low_and_all(1,7);
        ::button_1F!_1to8 -> run high_and_all(1,8);
        ::button_1F!_1to9 -> run high_and_all(1,9);
        ::button_1F!_1to10 -> run high_and_all(1,10);
        ::button_1F!_1to11 -> run high_and_all(1,11);
        ::button_1F!_1to12 -> run high_and_all(1,12);
        ::button_1F!_1to13 -> run high_and_all(1,13);
        ::button_1F!_1to14 -> run high_and_all(1,14);
        ::button_2F!_2to1 -> run low_and_all(2,1);
        ::button_2F!_2to3 -> run low_and_all(2,3);
        ::button_2F!_2to4 -> run low_and_all(2,4);

         …
```

- **메인 컨트롤러:**

✓ 저층•고층•전층 캐비닛 선택 및 동작

```
proctype low_and_all(byte from, to){      /* 저층 • 전층 캐비닛을 타야 할 때*/
        if
        ::(from>to) ->                    /* 사용자가 내려 갈 때 */
        if
        ::(A_cabin_dest_list[0]==-1 || B_cabin_dest_list[0]==-1 || E_cabin_dest_list[0]==-1 ||
        F_cabin_dest_list[0]==-1) ->
        do                                /* 사용자가 내려 갈 때 + 정지된 캐비닛이 있을 때*/
        ::A_cabin_dest_list[0]==-1 ->
                run sorting(from,to,1,false);
                break;
        ::B_cabin_dest_list[0]==-1 ->
                run sorting(from,to,2,false);
                break;
        ::E_cabin_dest_list[0]==-1 ->
                run sorting(from,to,5,false);
                break;
        ::F_cabin_dest_list[0]==-1 ->
                run sorting(from,to,6,false);
                break;
        od;
```

- **메인 컨트롤러:**

✓ 저층•고층•전층 캐비닛 선택 및 동작

```
::else ->   /* 사용자가 내려 갈 때 + 정지된 캐비닛이 없을 때*/
if
::((A_cabin_dest_list[0] - A_floor <0)&&(A_floor - from > 2)) || ((B_cabin_dest_list[0] -
B_floor <0)&&(B_floor - from > 2)) || ((E_cabin_dest_list[0] - E_floor <0)&&(E_floor -
from > 2)) || ((F_cabin_dest_list[0] - F_floor <0)&&(F_floor - from > 2)) ->

do          /* 사용자가 내려 갈 때 + 캐비닛이 사용자보다 2층 위에서 내려 올 때*/
::((A_cabin_dest_list[0] - A_floor <0)&&(A_floor - from > 2))->
                run sorting(from,to,1,false);
                break;
::((B_cabin_dest_list[0] - B_floor <0)&&(B_floor - from > 2)) ->
                run sorting(from,to,2,false);
                break;
::((E_cabin_dest_list[0] - E_floor <0)&&(E_floor - from > 2)) ->
                run sorting(from,to,5,false);
                break;
::((F_cabin_dest_list[0] - F_floor <0)&&(F_floor - from > 2)) ->
                run sorting(from,to,6,false);
                break;
od;
```

- **메인 컨트롤러:**

✓ 저층•고층•전층 캐비닛 선택 및 동작

```
::((A_cabin_dest_list[0] - A_floor <0)&&(A_floor - from <= 2)) || ((B_cabin_dest_list[0] -
B_floor <0)&&(from - B_floor <= 2)) || ((E_cabin_dest_list[0] - E_floor <0)&&(from -
E_floor <= 2)) || ((F_cabin_dest_list[0] - F_floor <0)&&(from - F_floor <= 2)) ->
      run buffer_low_and_all_down(from,to);
      /* 사용자가 내려 갈 때 + 캐비닛이 사용자 아래에서 내려 갈 때*/
      /* 아래로 내려가는 저층•고층용 버퍼에 저장*/

::(A_cabin_dest_list[0] - A_floor >0) || (B_cabin_dest_list[0] - B_floor >0) ||
(E_cabin_dest_list[0] - E_floor >0) || (F_cabin_dest_list[0] - F_floor >0) ->
      run buffer_low_and_all_down(from,to);
       /* 사용자가 내려 갈 때 + 캐비닛이 사용자 위 또는 아래에서 올라 갈 때*/
       /* 아래로 내려가는 저층•고층용 버퍼에 저장*/
 ::else -> skip;
 fi;
fi;
```

- **메인 컨트롤러:**

✓ 저층•고층•전층 캐비닛 선택 및 동작

```
::else ->                    /* 사용자가 올라 갈 때 + 정지된 캐비닛이 있을 때*/
    if
    ::(A_cabin_dest_list[0]==-1 || B_cabin_dest_list[0]==-1 || E_cabin_dest_list[0]==-1 ||
    F_cabin_dest_list[0]==-1) ->
            do
            ::A_cabin_dest_list[0]==-1 ->
                    run sorting(from,to,1,true);
                    break;
            ::B_cabin_dest_list[0]==-1 ->
                    run sorting(from,to,2,true);
                    break;
            ::E_cabin_dest_list[0]==-1 ->
                    run sorting(from,to,5,true);
                    break;
            ::F_cabin_dest_list[0]==-1 ->
                    run sorting(from,to,6,true);
                    break;
            od;
```

- **메인 컨트롤러:**

✓ 저층•고층•전층 캐비닛 선택 및 동작

```
::else ->        /* 사용자가 올라 갈 때 + 정지된 캐비닛이 없을 때*/
    if
    ::((A_cabin_dest_list[0] - A_floor >0)&&(from - A_floor > 2)) || ((B_cabin_dest_list[0] -
    B_floor >0)&&(from - B_floor > 2)) || ((E_cabin_dest_list[0] - E_floor >0)&&(from -
    E_floor > 2)) || ((F_cabin_dest_list[0] - F_floor >0)&&(from - F_floor > 2)) ->

    do          /* 사용자가 올라 갈 때 + 캐비닛이 사용자보다 2층 아래에서 올라 올 때*/
    ::((A_cabin_dest_list[0] - A_floor >0)&&(from - A_floor > 2)) ->
                run sorting(from,to,1,true);
                break;
    ::((B_cabin_dest_list[0] - B_floor >0)&&(from - B_floor > 2)) ->
                run sorting(from,to,2,true);
                break;
    ::((E_cabin_dest_list[0] - E_floor >0)&&(from - E_floor > 2)) ->
                run sorting(from,to,5,true);
                break;
    ::((F_cabin_dest_list[0] - F_floor >0)&&(from - F_floor > 2)) ->
                run sorting(from,to,6,true);
                break;
    od;
```

- **메인 컨트롤러:**

✓ 저층•고층•전층 캐비닛 선택 및 동작

```
::((A_cabin_dest_list[0] - A_floor >0)&&(from - A_floor <= 2)) || ((B_cabin_dest_list[0]
- B_floor >0)&&(from - B_floor <= 2)) || ((E_cabin_dest_list[0] - E_floor >0)&&(from
- E_floor <= 2)) || ((F_cabin_dest_list[0] - F_floor >0)&&(from - F_floor <= 2)) ->
        run buffer_low_and_all_up(from,to);
        /* 사용자가 올라 갈 때 + 캐비닛이 사용자 위에서 올라 갈 때*/
        /* 위로 올라가는 저층•고층용 버퍼에 저장*/

::(A_cabin_dest_list[0] - A_floor <0) || (B_cabin_dest_list[0] - B_floor <0) ||
(E_cabin_dest_list[0] - E_floor <0) || (F_cabin_dest_list[0] - F_floor <0) ->
        run buffer_low_and_all_up(from,to);
        /* 사용자가 올라 갈 때 + 캐비닛이 사용자 위 또는 아래에서 내려 올 때*/
        /* 위로 올라가는 저층•고층용 버퍼에 저장*/
::else -> skip;
fi;
    fi;
fi;
}
```

❖ 고층•전층 또는 전층 또한 같은 개념 적용!

- **캐비닛 컨트롤러:**

✓ 채널을 통해 메시지 전송

```
active proctype cabin_controller_E(){
        E_on1:                          /* 캐비닛의 층별 메시지 전송에 관한 사항*/
            E_cabin_state=false;
            door_stateE!open;
            door_stateE!close;
            if
            ::{sender_channel_E?_2F} ->
                        E_cabin_state=true;
                        upDown_E!UP;
                        E_floor = E_floor+1;
                        goto E_on2;
            ::{sender_channel_E?_3F} ->
                        E_cabin_state=true;
                        upDown_E!UP;
                        E_floor = E_floor+1;
                        upDown_E!UP;
                        E_floor = E_floor+1;
                        goto E_on3;

                        …
```

03

엘리베이터 검증

- **엘리베이터 검증 구문:**
  - ✓ 언젠간(<>) 엘리베이터는 도착하여야 한다
    - ltl p1 {<> (!A_cabin_state)}

```
smartelevator.pml:10602, state 29, "A_floor = 1"
smartelevator.pml:10603, state 30, "B_floor = 1"
smartelevator.pml:10604, state 31, "C_floor = 1"
smartelevator.pml:10605, state 32, "D_floor = 1"
smartelevator.pml:10606, state 33, "E_floor = 1"
smartelevator.pml:10607, state 34, "F_floor = 1"
smartelevator.pml:10609, state 35, "A_door_state = 0"
smartelevator.pml:10610, state 36, "B_door_state = 0"
smartelevator.pml:10611, state 37, "C_door_state = 0"
smartelevator.pml:10612, state 38, "D_door_state = 0"
smartelevator.pml:10613, state 39, "E_door_state = 0"
smartelevator.pml:10614, state 40, "F_door_state = 0"
smartelevator.pml:10616, state 41, "A_cabin_state = 0"
smartelevator.pml:10617, state 42, "B_cabin_state = 0"
smartelevator.pml:10618, state 43, "C_cabin_state = 0"
smartelevator.pml:10619, state 44, "D_cabin_state = 0"
smartelevator.pml:10620, state 45, "E_cabin_state = 0"
smartelevator.pml:10621, state 46, "F_cabin_state = 0"
smartelevator.pml:10623, state 47, "count_opened_door_A = 0"
smartelevator.pml:10625, state 48, "i = 0"
smartelevator.pml:10626, state 49, "j = 0"
smartelevator.pml:10629, state 51, "(run maincontroller())"
smartelevator.pml:10631, state 54, "-end-"
(38 of 54 states)
unreached in claim p1
        _spin_nvr.tmp:6, state 6, "-end-"
        (1 of 6 states)

pan: elapsed time 0.01 seconds
No errors found -- did you verify all claims?
```

03. 엘리베이터 검증

- **엘리베이터 검증 구문:**
  - ✓ 언제나([]) 하나의 라인 당, 하나의 문이 열려야 한다
    - ➢ ltl p2{[] count_opened_door_A==1 || count_opened_door_A==0}

```
            smartelevator.pml:10608, state 35, "A_door_state = 0"
            smartelevator.pml:10609, state 36, "B_door_state = 0"
            smartelevator.pml:10610, state 37, "C_door_state = 0"
            smartelevator.pml:10611, state 38, "D_door_state = 0"
            smartelevator.pml:10612, state 39, "E_door_state = 0"
            smartelevator.pml:10613, state 40, "F_door_state = 0"
            smartelevator.pml:10615, state 41, "A_cabin_state = 0"
            smartelevator.pml:10616, state 42, "B_cabin_state = 0"
            smartelevator.pml:10617, state 43, "C_cabin_state = 0"
            smartelevator.pml:10618, state 44, "D_cabin_state = 0"
            smartelevator.pml:10619, state 45, "E_cabin_state = 0"
            smartelevator.pml:10620, state 46, "F_cabin_state = 0"
            smartelevator.pml:10622, state 47, "count_opened_door_A = 0"
            smartelevator.pml:10624, state 48, "i = 0"
            smartelevator.pml:10625, state 49, "j = 0"
            smartelevator.pml:10628, state 51, "(run maincontroller())"
            smartelevator.pml:10630, state 54, "-end-"
            (38 of 54 states)
unreached in claim p2
            _spin_nvr.tmp:15, state 8, "-end-"
            (1 of 8 states)

pan: elapsed time 0.01 seconds
No errors found -- did you verify all claims?
```

03. 엘리베이터 검증

- **엘리베이터 검증 구문:**
  - ✓ 언젠간(<>) 엘리베이터가 모두 대기 상태일 수 있다
    - ➢ ltl p3{<> maincontroller:A_cabin_dest_list[0]==-1 && B_cabin_dest_list[0]==-1 && C_cabin_dest_list[0]==-1 && D_cabin_dest_list[0]==-1 && E_cabin_dest_list[0]==-1 && F_cabin_dest_list[0]==-1}

```
State-vector 1372 byte, depth reached 0, errors: 1
        1 states, stored
        0 states, matched
        1 transitions (= stored+matched)
        0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
    0.001   equivalent memory usage for states (stored*(State-vector + overhead))
    1.771   actual memory usage for states
   64.000   memory used for hash table (-w24)
   34.332   memory used for DFS stack (-m1000000)
   99.895   total actual memory usage



pan: elapsed time 0.004 seconds
To replay the error-trail, goto Simulate/Replay and select "Run"
```

03. 엘리베이터 검증

- **엘리베이터 검증 구문:**
  - ✓ 언제나(<>) 저층용 캐비닛은 1~7층(각층)에서 멈춘다
    - ltl p4{<> (A_floor<=7&&A_floor>=1) && !A_cabin_state}

```
Full statespace search for:
        never claim         + (p3)
        assertion violations + (if within scope of claim)
        acceptance   cycles  + (fairness disabled)
        invalid end states - (disabled by -E flag)

State-vector 1372 byte, depth reached 0, errors: 1
       1 states, stored
       0 states, matched
       1 transitions (= stored+matched)
       0 atomic steps
hash conflicts:         0 (resolved)

Stats on memory usage (in Megabytes):
   0.001   equivalent memory usage for states (stored*(State-vector + overhead))
   1.771   actual memory usage for states
  64.000   memory used for hash table (-w24)
  34.332   memory used for DFS stack (-m1000000)
  99.895   total actual memory usage



pan: elapsed time 0.004 seconds
To replay the error-trail, goto Simulate/Replay and select "Run"
```

03. 엘리베이터 검증

- **엘리베이터 검증 구문:**
  - ✓ 언젠가(<>) 고층용 캐비닛은 1,8~14층(각층)에서 멈춘다
    - ltl p5 {<> ((C_floor<=14&&C_floor>=8) || C_floor == 1) && !C_cabin_state }

```
(Spin Version 6.4.6 -- 2 December 2016)
Warning: Search not completed
        + Partial Order Reduction

Full statespace search for:
        never claim         + (p5)
        assertion violations + (if within scope of claim)
        acceptance   cycles  + (fairness disabled)
        invalid end states - (disabled by -E flag)

State-vector 1372 byte, depth reached 1170, errors: 0
   771315 states, stored (1.54185e+06 visited)
  1949350 states, matched
  3491197 transitions (= visited+matched)
        0 atomic steps
hash conflicts:     32479 (resolved)

Stats on memory usage (in Megabytes):
 1020.990 equivalent memory usage for states (stored*(State-vector + overhead))
  925.924 actual memory usage for states (compression: 90.69%)
        state-vector as stored = 1243 byte + 16 byte overhead
   64.000  memory used for hash table (-w24)
   34.332  memory used for DFS stack (-m1000000)
 1023.723 total actual memory usage



pan: elapsed time 25 seconds
No errors found -- did you verify all claims?
```

03. 엘리베이터 검증

- **엘리베이터 검증 구문:**
  - ✓ 언젠가(<>) 전층용 캐비닛은 1~14층(각층)에서 멈춘다
    - ➢ ltl p6 {<> (E_floor<=14&&E_floor>=1)&& !E_cabin_state }

```
Full statespace search for:
        never claim         + (p6)
        assertion violations + (if within scope of claim)
        acceptance   cycles  + (fairness disabled)
        invalid end states - (disabled by -E flag)

State-vector 1372 byte, depth reached 1490, errors: 0
   771315 states, stored (1.54155e+06 visited)
  1948463 states, matched
  3490011 transitions (= visited+matched)
        0 atomic steps
hash conflicts:     33579 (resolved)

Stats on memory usage (in Megabytes):
 1020.990 equivalent memory usage for states (stored*(State-vector + overhead))
  925.924 actual memory usage for states (compression: 90.69%)
        state-vector as stored = 1243 byte + 16 byte overhead
   64.000  memory used for hash table (-w24)
   34.332  memory used for DFS stack (-m1000000)
 1023.723 total actual memory usage



pan: elapsed time 25 seconds
No errors found -- did you verify all claims?
```

감사합니다