Elevator(UPPAAL)

정혁준, 이명재

Contents

- **Elevator System Specifications**
- **Modeling**
- **Model Checking**

- ***Elevator System Specifications***

• Elevator System

요구사항 정의 - 기본

• Elevator System

요구사항 정의

- 메인 모듈과 엘리베이터 내부 모듈은 정확한 위치를 동기화 한다.
- 엘리베이터 문은 특정한 위치에서만 열려야 한다.
- 엘리베이터 내부에서는 한 방향으로만 움직여야 한다.
- 엘리베이터 내부에서 누른 층으로 반드시 언젠가는 이동해야 한다.
- 특정 층에 서있을 때 엘리베이터 내부에서 열기 버튼을 누르면 엘리베이터 내부와 외부 문이 반드시 열려야 한다. 각각의 엘리베이터에는 엘리베이터를 구동하는 모듈이 있다.
- 전체 층과 전체 엘리베이터를 관리하는 메인 모듈이 있다.
- 엘리베이터는 4대로 구성되어 있다.
- 10층 건물이다.
- 이 외의 상세사항은 일반적인 엘리베이터의 특성을 따른다.

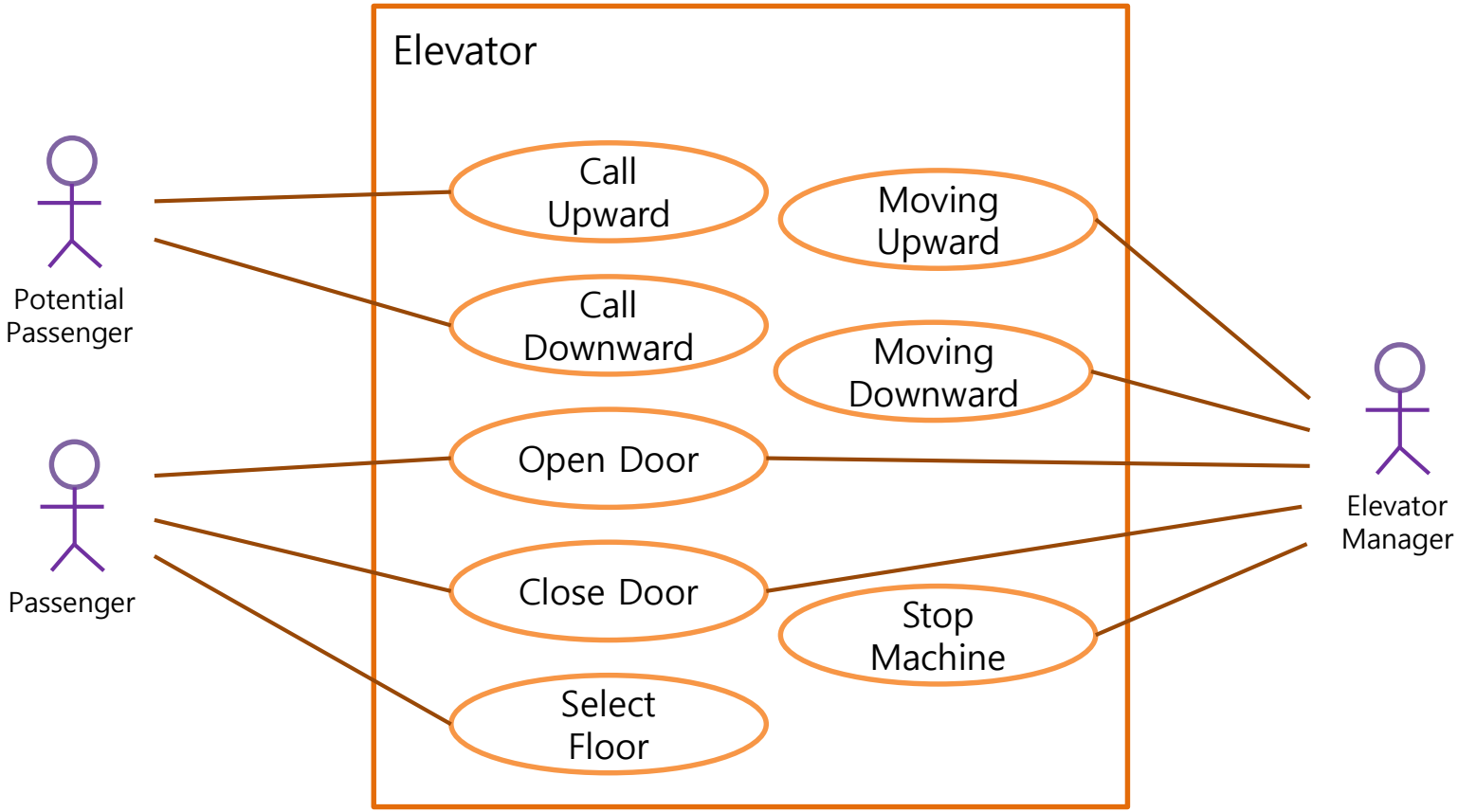
• Elevator System

요구사항 정의 – 층별 요구사항

- 특정 층에서 위 혹은 아래 버튼을 누르면 이동 방향을 고려해서 가장 가까운 엘리베이터가 와야 한다.
- 특정 층에서 위 혹은 아래 버튼을 누르면 언젠가는 반드시 엘리베이터가 와야 한다.
- 한 층에서 위 혹은 아래 버튼은 4대의 엘리베이터와 동기화 되어 버튼을 누르면 한 대의 엘리베이터만 와야 한다.
- 엘리베이터가 도착하지 않은 상태에서 각 층의 문은 절대 열리면 안 된다.
- 엘리베이터의 문과 각 층의 문은 항상 동시에 열려야 한다.

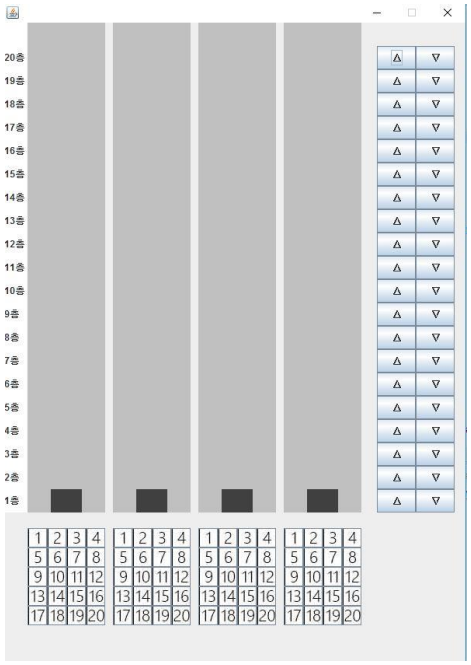
• Elevator System

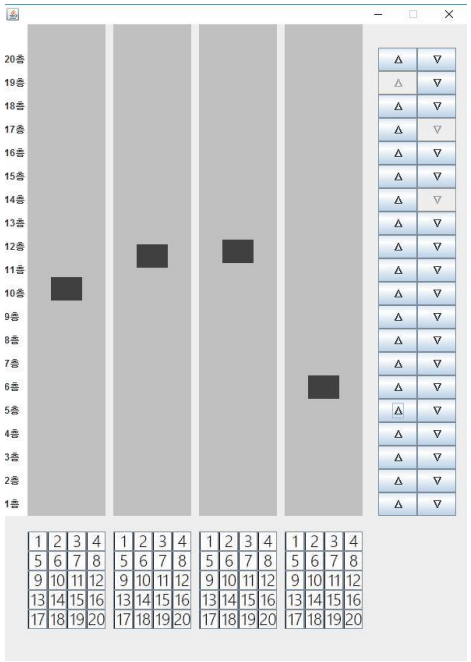
Use Case



• Elevator System

Elevator Simulator 구현

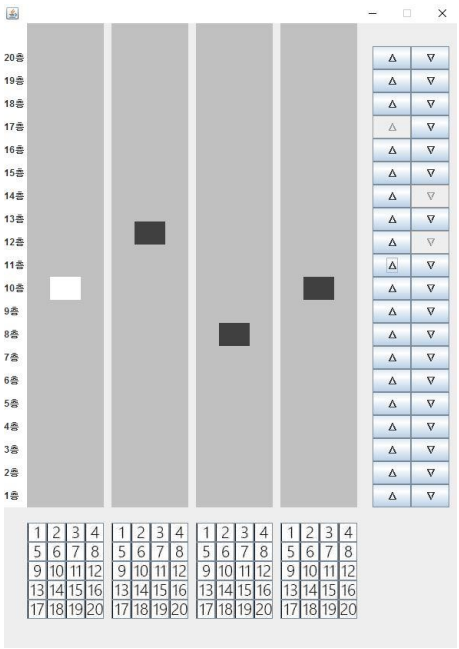


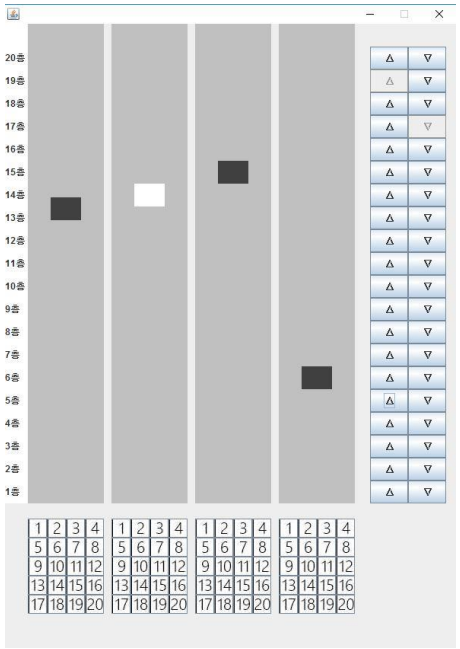


- Simulator 시작 시 4대의 엘리베이터는 1층에서 대기한다.
- 각 층에서 위 혹은 아래 버튼이 눌리면 각 엘리베이터에 목적 층이 정해지고 엘리베이터는 해당 층으로 이동한다.
- 각 층에서 위 혹은 아래 버튼이 눌리면 버튼이 비활성화 된다.

• Elevator System

Elevator Simulator 구현



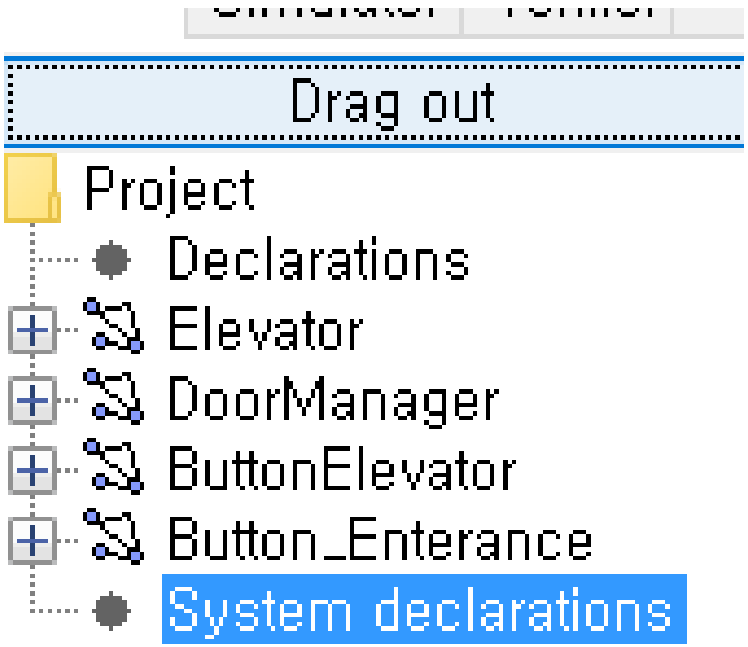


- 해당 층에 도착하면 엘리베이터가 멈추고 문이 잠시 열린 후 닫힌다.
- 해당 층에 도착하면 위 혹은 아래 버튼이 활성화된다.

- ***System Design***

• System Design

Template 구성



- Declarations
  - 변수 정의와 Elevator movement 관련 함수
- Elevator
  - Elevator 오토마타 정의
- DoorManager
  - 각 라인별 복도 문 관련 template
- ButtonElevator
  - 엘리베이터 내부 버튼 이벤트 처리
- ButtonEnterance
  - 복도측 엘리베이터 버튼 이벤트 처리

• System Design

int [a, b] num => int 변수를 a와 b 사이만 사용

declarations

```
const int numberEle=1;
const int maxFloor=10;
const int minFloor=1;
const int delay=15;
int[1,1] UPWARD=1;
int[2,2] DOWNWARD=2;
int[0,0] STANDBY=0;
int[-1,maxFloor+1] eleFloor[numberEle];
int[0,1] eleDestFloor[numberEle][maxFloor+2]
int[0,2] direction[numberEle];
int[0,2] directionSave[numberEle];
int[0,15] openDoorFloor[numberEle];
```

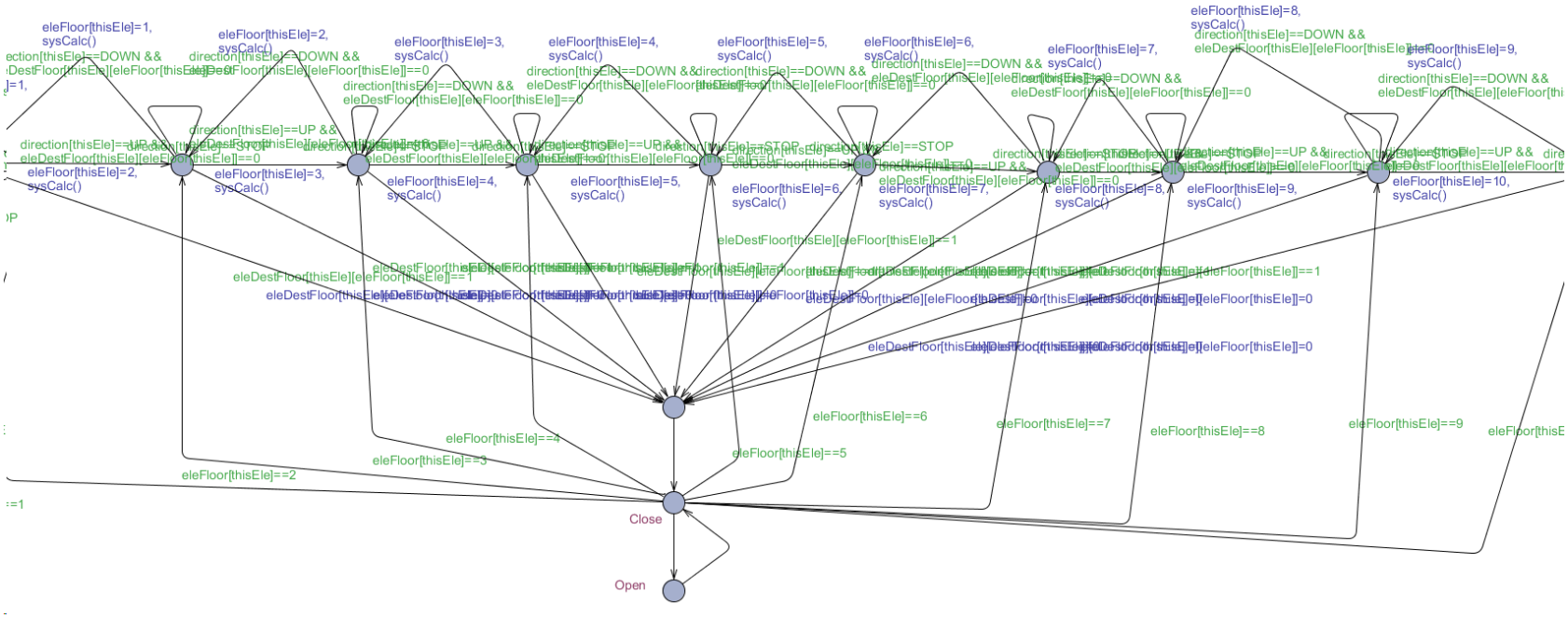
```
void calcEle(int[0,numberEle] num){
        int[-1,maxFloor+1] i=0;
        int[0,10] count1=0;
        int[0,10] count2=0;
        if(eleDestFloor[num][eleFloor[num]]==1){
                direction[num]=STANDBY;
                return;
        }
        for(i=eleFloor[num];i<=maxFloor;i++){
                if(eleDestFloor[num][i]==1){
                        count1++;
                }
        }
```

다양한 상황에 대응하기 위해 모든 변수를 배열 형태로 선언

State Explosion 현상 발생

• System Design

Elevator

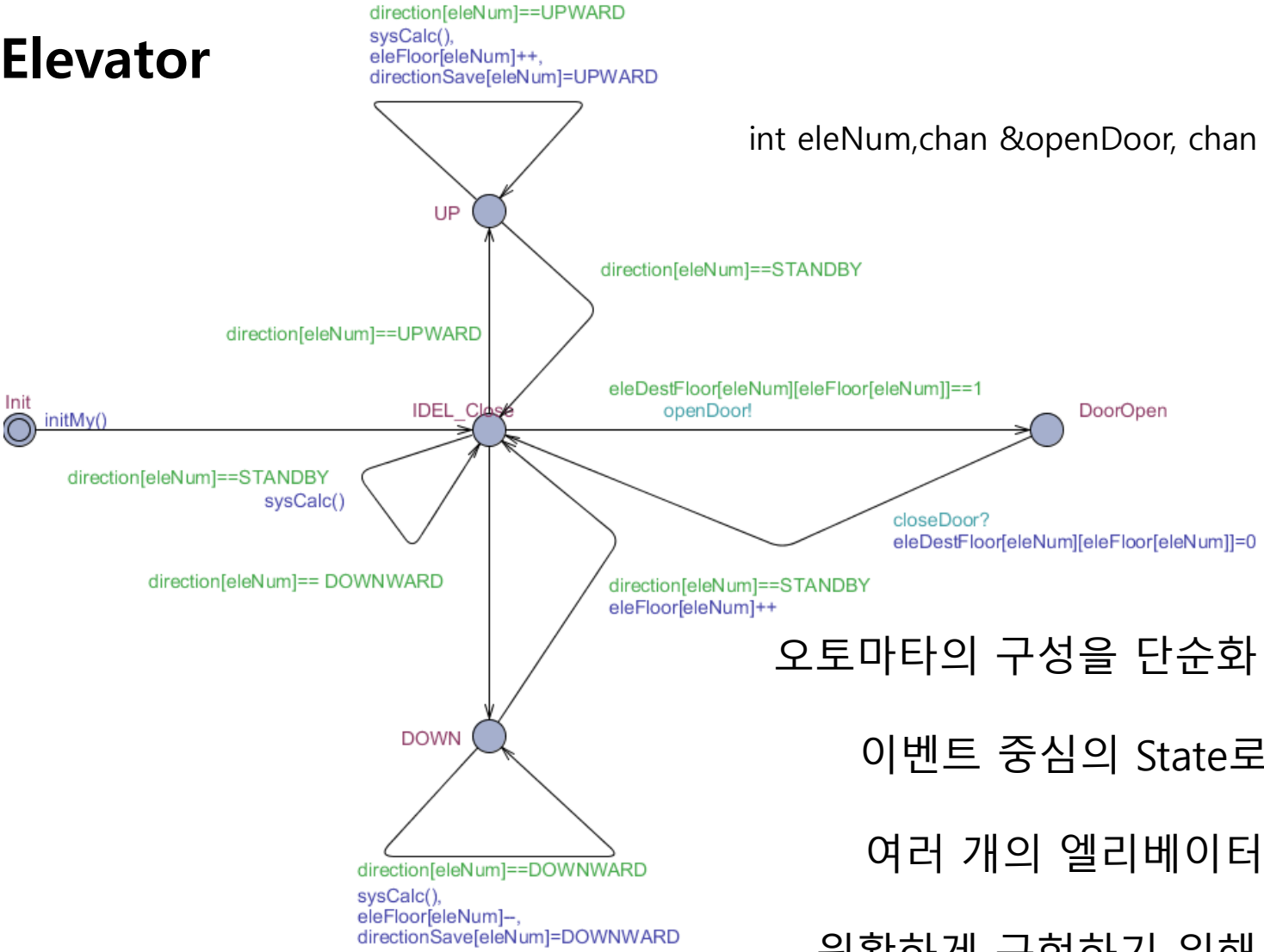


프로젝트 초반에 구성한 Elevator 오토마타

너무 복잡한 구성 때문에 다음 슬라이드와 같이 변경

• System Design

Elevator



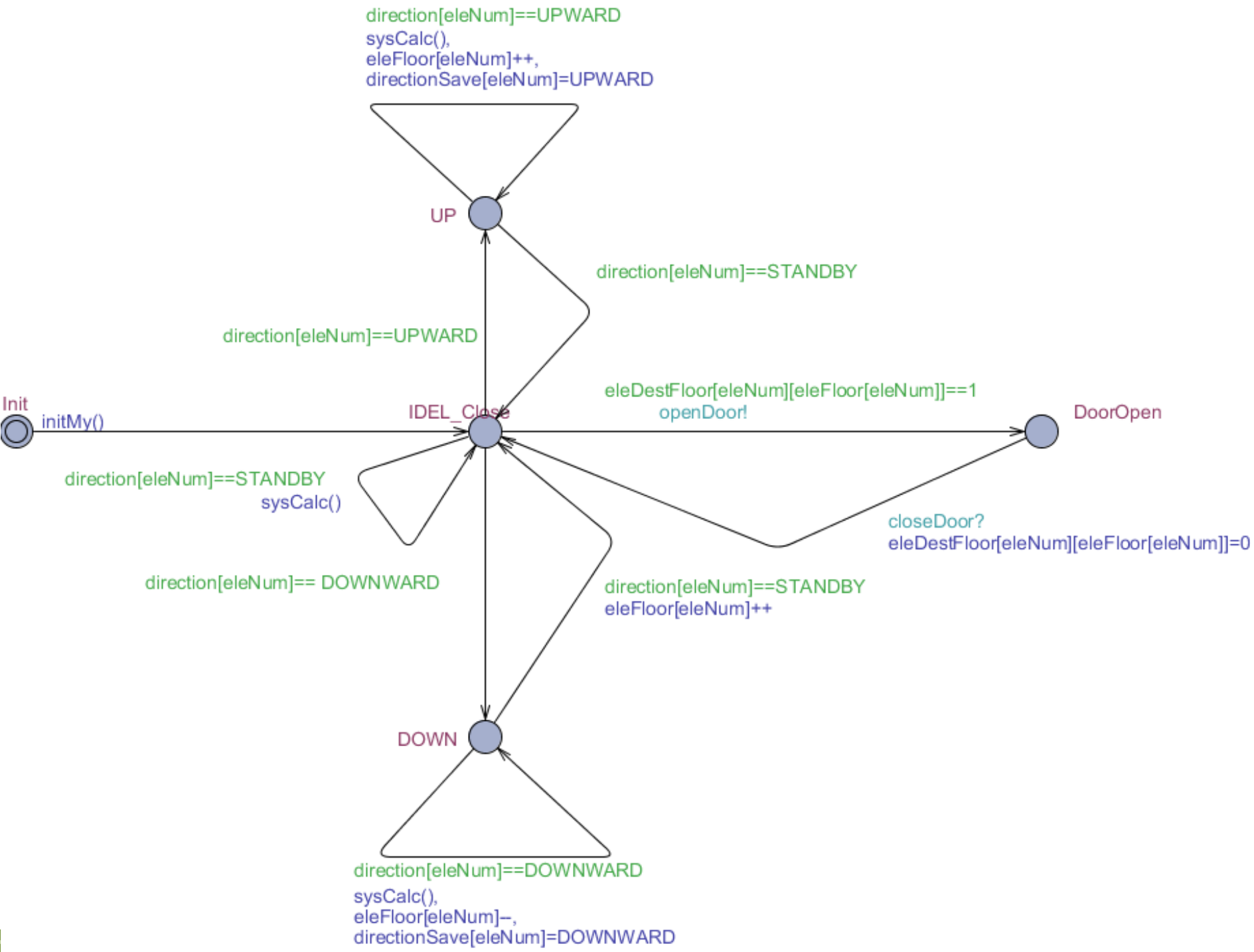
int eleNum,chan &openDoor, chan &closeDoor

오토마타의 구성을 단순화 하기 위하여

이벤트 중심의 State로 구성함

여러 개의 엘리베이터를 보다

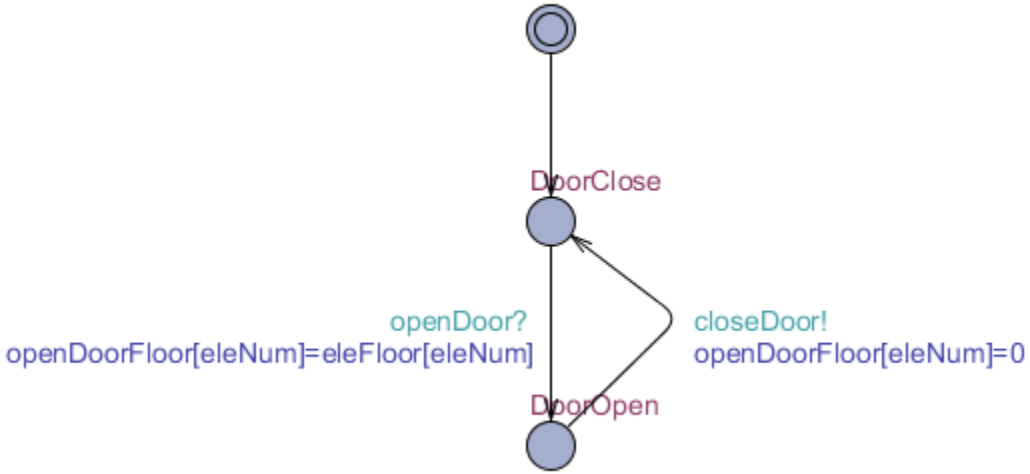
원활하게 구현하기 위해 배열 도입


direction[eleNum]==UPWARD
sysCalc(),
eleFloor[eleNum]++,
directionSave[eleNum]=UPWARD
UP
direction[eleNum]==STANDBY
direction[eleNum]==UPWARD
Init
initMy()
IDEL_Close
eleDestFloor[eleNum][eleFloor[eleNum]]==1
openDoor!
DoorOpen
direction[eleNum]==STANDBY
sysCalc()
closeDoor?
eleDestFloor[eleNum][eleFloor[eleNum]]=0
direction[eleNum]== DOWNWARD
direction[eleNum]==STANDBY
eleFloor[eleNum]++
DOWN
direction[eleNum]==DOWNWARD
sysCalc(),
eleFloor[eleNum]–,
directionSave[eleNum]=DOWNWARD

• System Design

DoorManager

int eleNum, chan &openDoor, chan &closeDoor

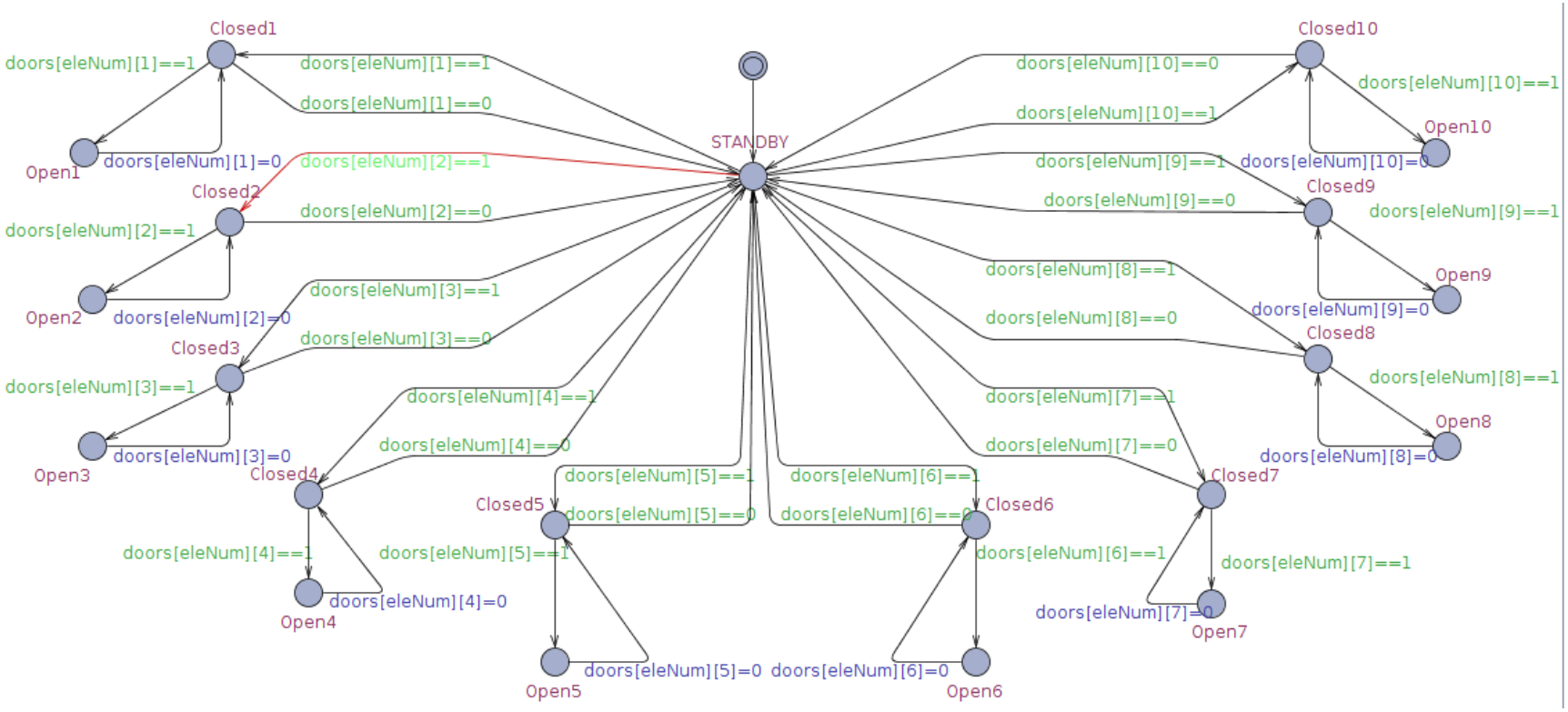


각 층의 복도 측 문을 여닫는 기능 담당함

채널로 이벤트를 전달 받고 해당 층의 문을 여는 제어기

• System Design

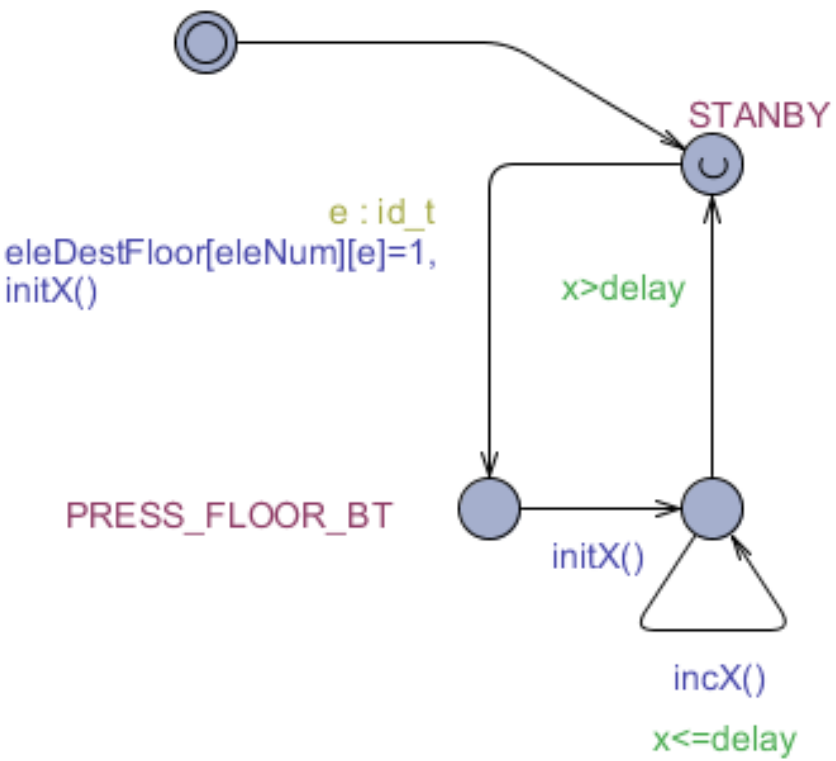
DoorManager



현재 버전 적용 이전의 오토마타

• System Design

ElevatorButton



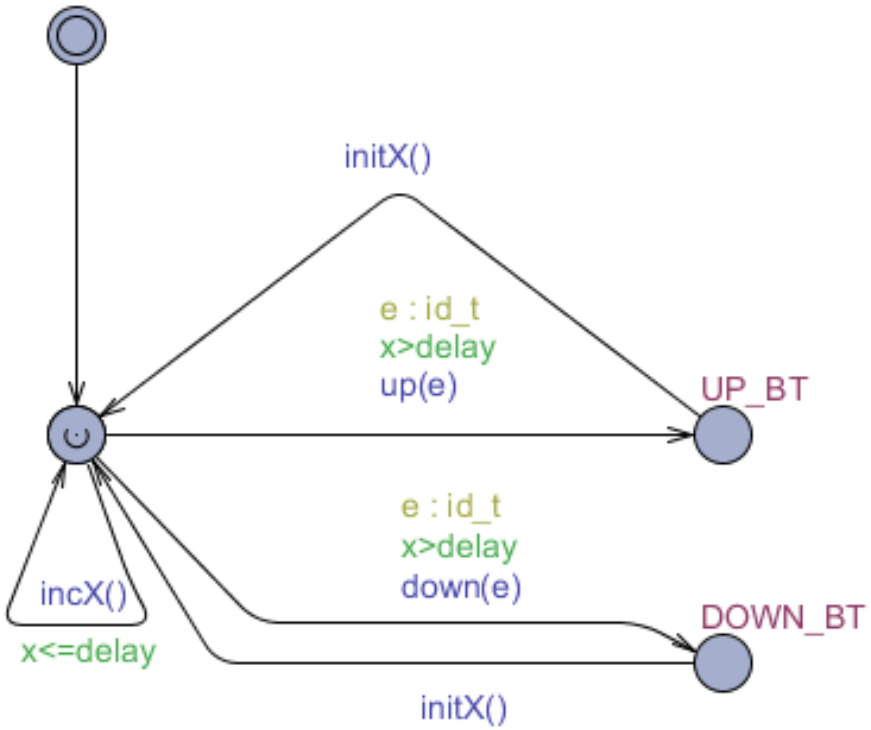
각 엘리베이터 내의 버튼 기능을 수행함

임의의 하나의 버튼을 눌러 엘리베이터를 작동 시킬 수 있도록 설계함

⇒ 시뮬레이션 측면에서는 좋으나 복잡도를 증가시키는 큰 요인

• System Design

ButtonEnterance



복도 측 엘리베이터 제어 버튼(위, 아래)

임의의 하나의 버튼을 눌러 엘리베이터 중에서 한대를 부를 수 있도록 설계

⇒ 시뮬레이션 측면에서는 좋으나
복잡도를 증가시키는 큰 요인

• System Design

System Declaration

```
chan openDoor[4];
chan closeDoor[4];
elevator1 = Elevator(0,openDoor[0],closeDoor[0]);
elevator2 = Elevator(1,openDoor[1],closeDoor[1]);
elevator3 = Elevator(2,openDoor[2],closeDoor[2]);
elevator4 = Elevator(3,openDoor[3],closeDoor[3]);
doorLine1 = DoorManager(0,openDoor[0],closeDoor[0]);
doorLine2 = DoorManager(1,openDoor[1],closeDoor[1]);
doorLine3 = DoorManager(2,openDoor[2],closeDoor[2]);
doorLine4 = DoorManager(3,openDoor[3],closeDoor[3]);
button_enterance=ButtonEnterance();
bt1= ButtonElevator(0);
bt2= ButtonElevator(1);
bt3= ButtonElevator(2);
bt4= ButtonElevator(3);
// List one or more processes to be composed into a system.
system button_enterance
        ,elevator4,doorLine4,bt4
        ,elevator3,doorLine3,bt3
        ,elevator2,doorLine2,bt2
        ,elevator1,doorLine1,bt1
```

템플릿을 실제로 사용하기 위해서 시스템을 정의함

엘리베이터 4대를 생성 후 채널 배정

4라인의 복도측 문 제어기를 생성

엘리베이터 버튼 4개 생성

➔ 시스템이 너무 크게 설계됨

• System Design

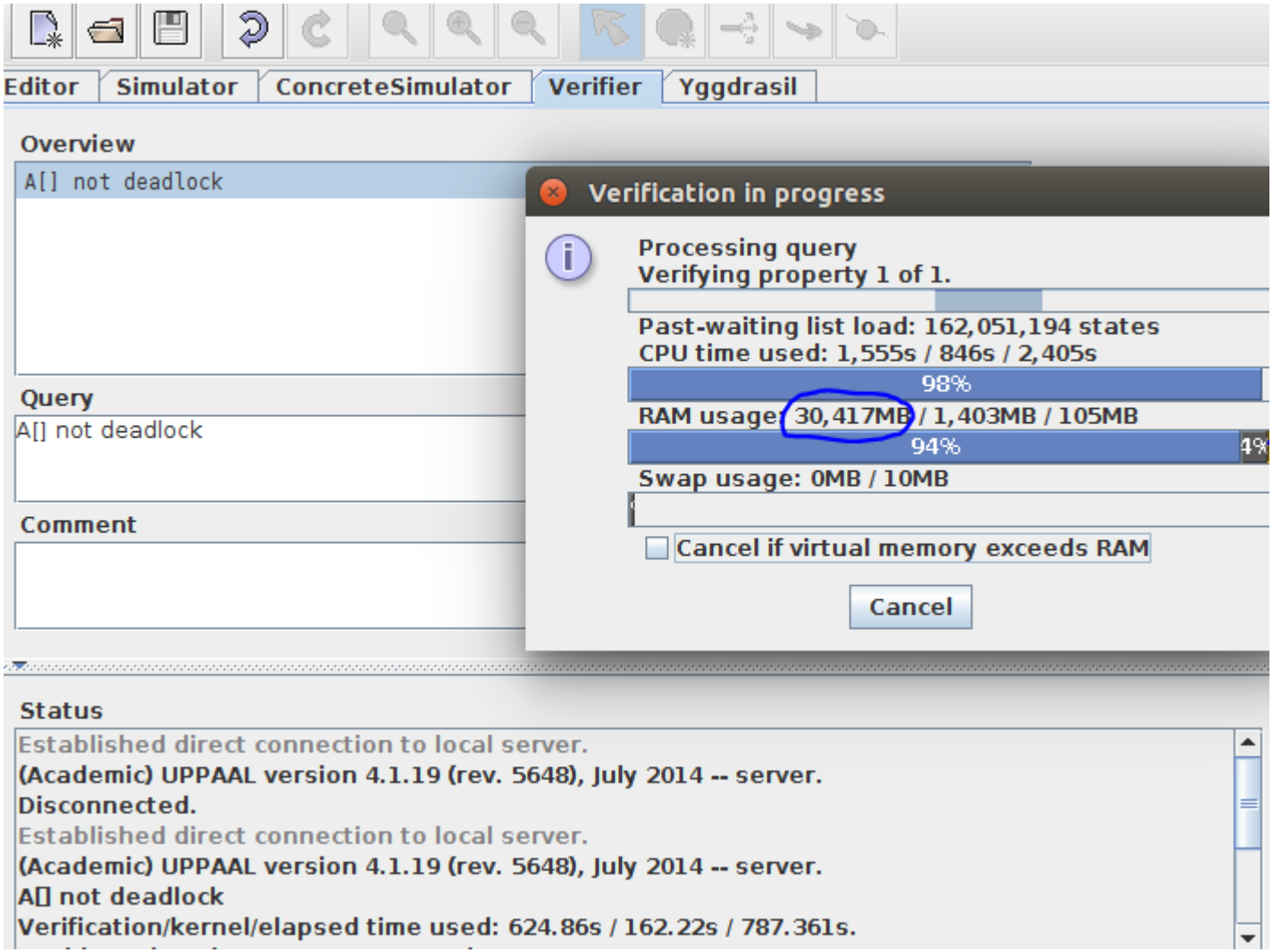
Design Discussion

1. 확장성을 고려한 시스템 설계
2. 너무 많은 기능을 고려한 시스템 설계
3. 오토마타의 지나친 단순화
4. 함수 및 배열의 사용(코드 재 사용성 고려)

프로그래밍 스타일의
시스템 설계

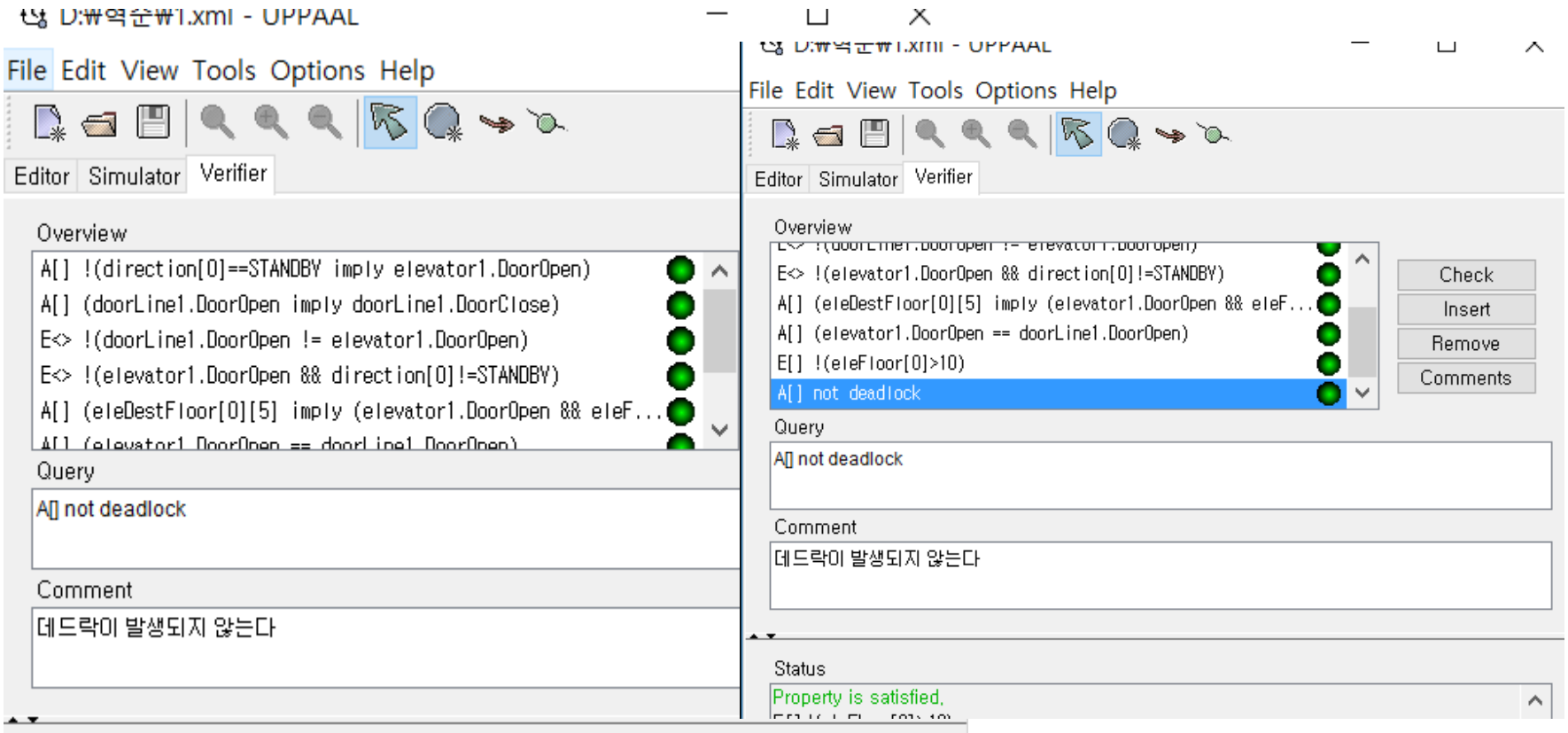
→ **State Explosion**

• System Verification



- ***System Verification***

• System Verification



- **System Verification**

A[] not deadlock

데드락이 발생되지 않는다

True

- **System Verification**

E[] !(eleFloor[0]>10)

10층을 초과해서 운행하지 않는다

True

- **System Verification**

A[] (elevator1.DoorOpen == doorLine1.DoorOpen)

엘레베이터 문이 열릴때는 복도 측 도어도 함께 열린다

True

- **System Verification**

A[] (eleDestFloor[0][5] imply (elevator1.DoorOpen && eleFloor[0]==5))

목적지가 5층이라면 엘레베이터가 5층에 도착해서 문이 열리는 경우가 반드시 있다

True

- **System Verification**

E<> !(elevator1.DoorOpen && direction[0]!=STANDBY)

엘레베이터 문이 열리는 경우엔 엘레베이터가 항상 정지해 있는다.
(엘레베이터 문이 열리는데 엘레베이터가 정지하지 않은 경우는 없다)

True

- **System Verification**

A[] (doorLine1.DoorOpen imply doorLine1.DoorClose)

문이 열리면 반드시 문이 다시 닫힌다

True

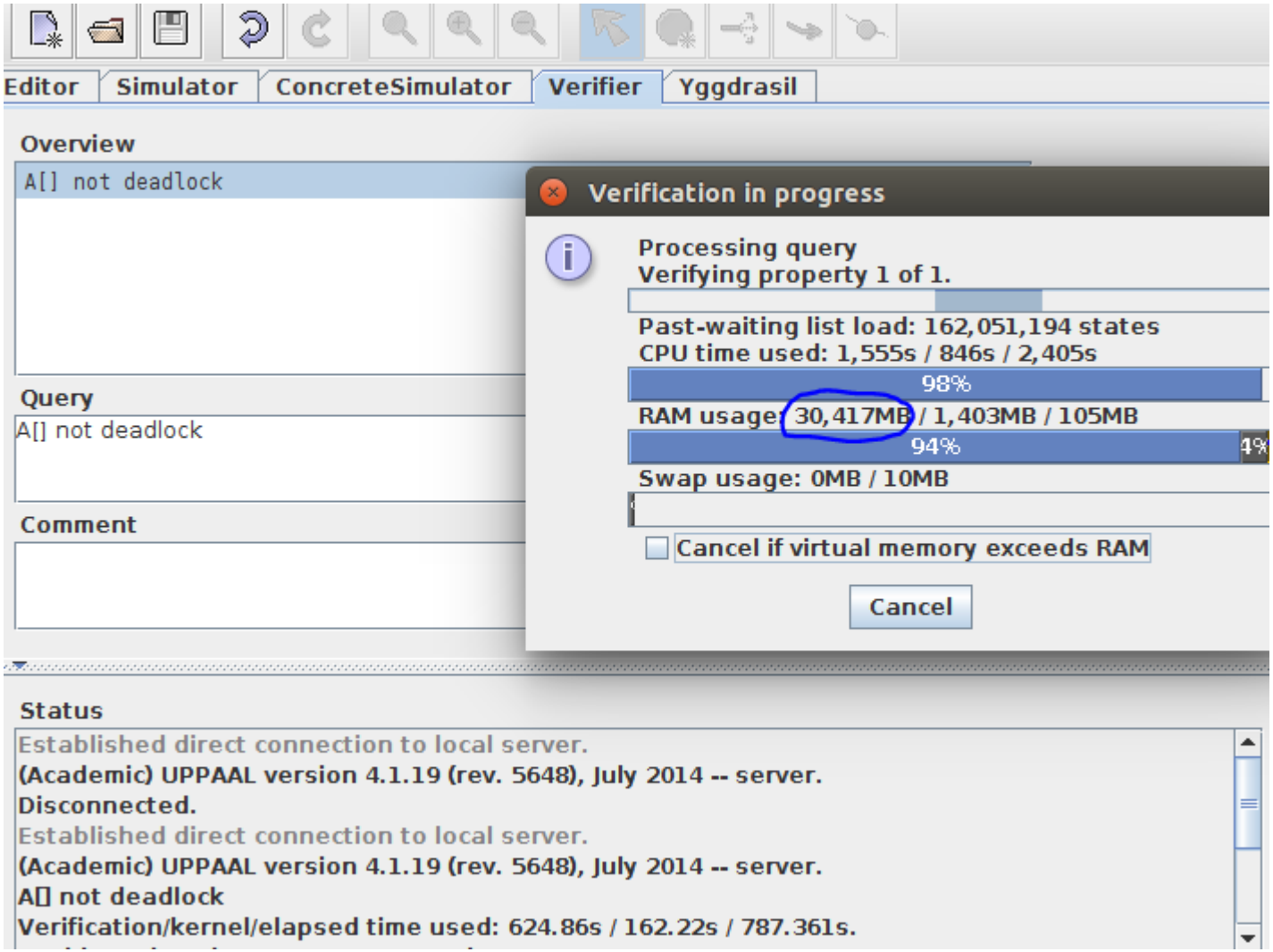
- **System Verification**

A[] !(direction[0]==STANDBY imply elevator1.DoorOpen)

엘레베이터가 정지해 있는 상태라고 해서 반드시 문이 열려있는 것은 아니다.
(닫힌 상태 대기)

True

• System Verification



Thank you

Question & Answer