CS580
Homework 07:
Elevator system.
Yie Sheng Chen 1303

Introduction

A formal specification of an elevator system with a single cabin in a building with N floors is to be developed. There are upCall and downCall buttons on each floor except the first and Nth floors where there is only downCall button at the Nth floor and upCall button at the first floor.

Abstract State Model

```
cMo : cabin motion ;
      integer enumerated type with 3 possible values:
      0:STOPPED,
      1:UP(going up),
      2:DOWN (going down)
cLoc : cabin location
      integer ;
      0 means the cabin is between any 2 floors
      1~N means the cabin is at exactly one of the floors

cDoorOpen : cabin door is open
      boolean

fDoorOpen : door to the shaft is open
      boolean array of length N
      where the k-th element represents
      whether door at k-th floor is open

downCall : down call button at each floor
      boolean array of length N
      where the k-th element represents the down call button
      at k-th floor is pressed

upCall : up call button at each floor
      boolean array of length N
      where the k-th element represents the up call button
      at k-th floor is pressed

cReq : cabin request
      boolean array of length N
      where the k-th element represents a stop
      at k-th floor is requested

fireAlarm : whether the fire alarm is on
      boolean

cDir : the direction of cabin and is used for spec no. 3
      integer enumerated type with 3 possible values:
      0:NONE (no need to go anywhere),
      1:UP(going up),
      2:DOWN (going down)

cStop2s : whether the cabin stop for 2 seconds and is used for spec no. 3
      boolean
```

notations used:

```
|-> : leads to
=> : logical implication
<=> : if and only if
inv. q : q is invariant
{comments} : comments are enclosed in curly brackets
```

specification no. 1

a highly abstract and minimal specification of the requirements the system must meet.

```
{safety req:}
inv. ¬(cMo=STOPPED) <=> ¬cDoorOpen ^ ¬fDoorOpen[k]
inv. cDoorOpen <=> cMo=STOPPED ^ fDoorOpen[cLoc] ^ ¬(cLoc=0)

{service req:}
upCall[k]   until ¬upCall[k] ^ cMo=STOPPED ^ cLoc=k ^ cDoorOpen
downCall[k] until ¬downCall[k] ^ cMo=STOPPED ^ cLoc=k ^ cDoorOpen
cReq[k]     until ¬cReq[k] ^ cMo=STOPPED ^ cLoc=k ^ cDoorOpen

{additional formuals for call buttons at 1st and Nth floors}
inv. upCall[N]=False
inv. downCall[1]=False
```

specification no. 2

In this spec., presence of the fire alarm and any requirements related to handling the fire emergency are factored in

```
{safety req:}
inv. ¬(cMo=STOPPED) <=> ¬cDoorOpen ^ ¬fDoorOpen[k]
inv. cDoorOpen <=> cMo=STOPPED ^ fDoorOpen[cLoc] ^ ¬(cLoc=0)

{service req:}
upCall[k] ^ ¬fireAlarm   until ¬upCall[k] ^ cMo=STOPPED ^ cLoc=k
downCall[k] ^ ¬fireAlarm until ¬downCall[k] ^ cMo=STOPPED ^ cLoc=k
cReq[k] ^ ¬fireAlarm     until ¬cReq[k] ^ cMo=STOPPED ^ cLoc=k
fireAlarm   until fireAlarm ^ cLoc=1 ^ cMo=STOPPED ^ cDoorOpen ^ fDoorOpen[1]

{additional formuals for call buttons at 1st and Nth floors}
inv. upCall[N]=False
inv. downCall[1]=False
```

specification no. 3

a more detailed and prescriptive refinement of spec. no. 2

```
{function definition:}
isUpCalled(k) : <∃ i : k < i <= N :: upCall[i] V cReq[i]>
isDownCalled(k) : <∃ i : 0 < i < k :: downCall[i] V cReq[i]>

{safety req:}
inv. ¬(cMo=STOPPED) <=> ¬cDoorOpen ^ ¬fDoorOpen[k]
inv. cDoorOpen <=> cMo=STOPPED ^ fDoorOpen[cLoc] ^ ¬(cLoc=0)

{service req:}
{no.1}
p1 until q1
where
p1 = cLoc=k ^ isUpCalled(k) ^ cDir=UP
            ^ (upCall[k+1] V cReq[k+1]) ^ ¬fireAlarm
```

```
            ^ ¬(cLoc=N)
q1 = cLoc=k+1 ^ upCall[k+1]=False ^ cReq[k+1]=False
            ^ cDir=UP ^ cStop2s=True

{no.2}
p2 until q2
where
p2 = cLoc=k ^ isUpCalled(k) ^ cDir=UP
            ^ ¬(upCall[k+1] V cReq[k+1]) ^ ¬fireAlarm
            ^ ¬(cLoc=N)
q2 = cLoc=k+1 ^ cDir=UP ^ cStop2s=False

{no.3}
cStop2s until cStop2s=False

{no.4}
p3 until q3
where
p3 = cLoc=k ^ ¬isUpCalled(k) ^ ¬isDownCalled(k)
            ^ (cDir=UP V cDir=DOWN) ^ ¬fireAlarm
q3 = cDir=NONE


{no.5}
p4 until q4
where
p4 = cLoc=k ^ ¬isUpCalled(k) ^ isDownCalled(k)
            ^ cDir=UP ^ ¬fireAlarm
q4 = cDir=DOWN

{no.6}
p5 until q5
where
p5 = cLoc=k ^ isDownCalled(k) ^ cDir=DOWN
            ^ (downCall[k-1] V cReq[k-1]) ^ ¬fireAlarm
            ^ ¬(cLoc=1)
q5 = cLoc=k-1 ^ upCall[k-1]=False ^ cReq[k-1]=False
            ^ cDir=DOWN ^ cStop2s=True

{no.7}
p6 until q6
where
p6 = cLoc=k ^ isDownCalled(k) ^ cDir=DOWN
            ^ ¬(downCall[k-1] V cReq[k-1]) ^ ¬fireAlarm
            ^ ¬(cLoc=1)
q6 = cLoc=k-1 ^ cDir=DOWN ^ cStop2s=False

{no.8}
p7 until q7
where
p7 = cLoc=k ^ isUpCalled(k) ^ ¬isDownCalled(k)
            ^ cDir=DOWN ^ ¬fireAlarm
q7 = cDir=UP

{no.9}
fireAlarm ^ cLoc=k ^ ¬(cLoc=1) until cLoc=k-1

{no.10}
fireAlarm ^ cLoc=1 until cDir=NONE ^ cDoorOpen

{additional formuals for call buttons at 1st and Nth floors}
inv. upCall[N]=False
inv. downCall[1]=False
```