

# Modelling a Lift Control System

Ken Robinson

January 29, 2009

CONTEXT Lift.ctx

SETS

*DIRECTION*

*STATUS*

CONSTANTS

*MAXFLOOR*

*FLOOR*

*MAXLIFT*

*LIFT*

*UP*

*DOWN*

*STOPPED*

*MOVING*

*CHANGE*

AXIOMS

*axm1* :  $MAXFLOOR \in \mathbb{N}_1$

*axm2* :  $FLOOR = 0 .. MAXFLOOR$

*axm3* :  $MAXLIFT \in \mathbb{N}_1$

*axm4* :  $LIFT = 1 .. MAXLIFT$

*axm5* :  $DIRECTION = \{UP, DOWN\}$

*axm6* :  $UP \neq DOWN$

*axm7* :  $STATUS = \{STOPPED, MOVING\}$

*axm8* :  $STOPPED \neq MOVING$

*axm9* :  $CHANGE \in DIRECTION \rightsquigarrow DIRECTION$

*axm10* :  $CHANGE = \{UP \mapsto DOWN, DOWN \mapsto UP\}$

THEOREMS

*thm1* :  $FLOOR \neq \emptyset$

END

**MACHINE** BasicLift

The machine models the basic lift movements.

The behaviour is all non-deterministic: there is no attempt to express any sort of lift control or scheduling.

A discipline of lift direction is established: level 0: direction is UP level MAXFLOOR: direction is DOWN other floors: either direction is valid

There are no doors.

There are no buttons

**SEES** Lift\_ctx**VARIABLES**

*liftposition*

*liftstatus*

*liftdirection*

**INVARIANTS**

*inv1* :  $liftposition \in LIFT \rightarrow FLOOR$

*inv2* :  $liftstatus \in LIFT \rightarrow STATUS$

*inv3* :  $liftdirection \in LIFT \rightarrow DIRECTION$

$\forall l.l \in LIFT \wedge liftposition(l) = 0$

*inv4* :  $\Rightarrow$

$liftdirection(l) = UP$

$\forall l.l \in LIFT \wedge liftposition(l) = MAXFLOOR$

*inv5* :  $\Rightarrow$

$liftdirection(l) = DOWN$

**THEOREMS**

*thm1* :  $\forall l.l \in LIFT \wedge liftdirection(l) = DOWN$   
 $\Rightarrow$   
 $liftposition(l) \neq 0$

*thm2* :  $\forall l.l \in LIFT \wedge liftdirection(l) = UP$   
 $\Rightarrow$   
 $liftposition(l) \neq MAXFLOOR$

*thm3* :  $\forall l.l \in LIFT \wedge liftdirection(l) = UP$   
 $\Rightarrow$   
 $liftposition(l) + 1 \leq MAXFLOOR$

**EVENTS****Initialisation**

**begin**

$liftposition, liftdirection, liftstatus : |$   
 $liftposition' \in LIFT \rightarrow FLOOR$   
 $\wedge liftdirection' \in LIFT \rightarrow DIRECTION$   
 $\wedge liftstatus' \in LIFT \rightarrow STATUS$   
**act1** :  $\wedge (\forall l \cdot l \in LIFT \wedge liftposition'(l) = 0$   
 $\Rightarrow$   
 $liftdirection'(l) = UP)$   
 $\wedge (\forall m \cdot m \in LIFT \wedge liftposition'(m) = MAXFLOOR$   
 $\Rightarrow$   
 $liftdirection'(m) = DOWN)$

There seems to be a bug in Rodin. The two quantifications, in  $l$  and  $m$ , are interfering. See PO Initialisation FIS!! Also, if the quantification over  $m$  is replaced by quantification over  $l$ , which of course is legal, a syntax error is diagnosed!

**end**

**Event** *StopLift*  $\hat{=}$

Models a lift arriving at a floor and stopping

**any**

*lift*

**where**

**grd1** :  $lift \in LIFT$

**grd2** :  $liftstatus(lift) = MOVING$

**then**

**act1** :  $liftstatus(lift) := STOPPED$

**end**

**Event** *StartLift*  $\hat{=}$

Models the starting of a STOPPED lift, maintaining of previous direction

**any**

*lift*

**where**

**grd1** :  $lift \in LIFT$

**grd2** :  $liftstatus(lift) = STOPPED$

**then**

**act1** :  $liftstatus(lift) := MOVING$

**end**

**Event** *ChangeDir*  $\hat{=}$

Models the changing of direction of a STOPPED lift

**any**

*lift*

**where**

**grd1** :  $lift \in LIFT$

**grd2** :  $liftstatus(lift) = STOPPED$

```

    grd3 : liftposition(lift) ≠ 0
    grd4 : liftposition(lift) ≠ MAXFLOOR
  then
    act1 : liftdirection(lift) := CHANGE(liftdirection(lift))
  end
Event MoveUp ≐
  Models a lift moving up to the next floor
  any
    lift
  where
    grd1 : lift ∈ LIFT
    grd2 : liftstatus(lift) = MOVING
    grd3 : liftdirection(lift) = UP
  then
    act1 : liftposition(lift) := liftposition(lift) + 1
           liftdirection : |liftdirection' ∈ LIFT → DIRECTION
           ∧ (liftposition(lift) + 1 = MAXFLOOR
           ⇒
    act2 : liftdirection' = liftdirection ⇐ {lift ↦ DOWN}
           ∧ (liftposition(lift) + 1 ≠ MAXFLOOR
           ⇒
           liftdirection' = liftdirection)
  end
Event MoveDown ≐
  Models a lift moving down to the next floor
  any
    lift
  where
    grd1 : lift ∈ LIFT
    grd2 : liftstatus(lift) = MOVING
    grd3 : liftdirection(lift) = DOWN
  then
    act1 : liftposition(lift) := liftposition(lift) - 1
           liftdirection : |liftdirection' ∈ LIFT → DIRECTION
           ∧ (liftposition(lift) = 1
           ⇒
    act2 : liftdirection' = liftdirection ⇐ {lift ↦ UP}
           ∧ (liftposition(lift) ≠ 1
           ⇒
           liftdirection' = liftdirection)
  end
END

```

**CONTEXT** Doors.ctx

**SETS**

*DOORS*

**CONSTANTS**

*OPEN*

*CLOSED*

**AXIOMS**

*axm1* :  $DOORS = \{OPEN, CLOSED\}$

*axm2* :  $OPEN \neq CLOSED$

**END**

**MACHINE** LiftPlusDoors  
 This refinement adds lift doors  
 Lift doors must always be closed when the lift is moving

**REFINES** BasicLift

**SEES** Lift\_ctx, Doors\_ctx

**VARIABLES**

*liftposition*  
*liftstatus*  
*liftdirection*  
*liftdoorstatus*

**INVARIANTS**

*inv1* :  $liftdoorstatus \in LIFT \rightarrow DOORS$   
 $\forall l \cdot l \in LIFT \wedge liftstatus(l) = MOVING$   
*inv2* :  $\Rightarrow$   
 $liftdoorstatus(l) = CLOSED$

**EVENTS**

**Initialisation**  
*extended*

**begin**  
 $act2$  :  $liftdoorstatus := LIFT \times \{CLOSED\}$   
**end**

**Event** *OpenLiftDoor*  $\hat{=}$

**any**  
*lift*  
**where**  
 $grd1$  :  $lift \in LIFT$   
 $grd2$  :  $liftstatus(lift) = STOPPED$   
 $grd3$  :  $liftdoorstatus(lift) = CLOSED$   
**then**  
 $act1$  :  $liftdoorstatus(lift) := OPEN$   
**end**

**Event** *CloseLiftdoor*  $\hat{=}$

**any**  
*lift*  
**where**  
 $grd1$  :  $lift \in LIFT$

```

        grd2 : liftstatus(lift) = STOPPED
    then
        act1 : liftdoorstatus(lift) := CLOSED
    end

Event StopLift  $\hat{=}$ 
    Models a lift arriving at a floor and stopping
extends StopLift

    begin
        skip
    end

Event StartLift  $\hat{=}$ 
    Models the starting of a STOPPED lift, maintaining of previous direction
extends StartLift

    when
        grd3 : liftdoorstatus(lift) = CLOSED
    then
        skip
    end

Event ChangeDir  $\hat{=}$ 
    Models the changing of direction of a STOPPED lift
extends ChangeDir

    begin
        skip
    end

Event MoveUp  $\hat{=}$ 
    Models a lift moving up to the next floor
extends MoveUp

    begin
        skip
    end

Event MoveDown  $\hat{=}$ 
    Models a lift moving down to the next floor
extends MoveDown

    begin
        skip
    end

END

```

**MACHINE** LiftPlusFloorDoors

This refinement adds floor doors

Floor doors may be OPEN only on the floor where a lift is stopped

The floor door opens AFTER the lift door opens

The floor door closes BEFORE the lift door closes

This ensures that floor door OPEN implies lift door OPEN

And if a lift is MOVING then the floor door for that lift is CLOSED on all floors

**REFINES** LiftPlusDoors

**SEES** Lift\_ctx, Doors\_ctx

**VARIABLES**

*liftposition*

*liftstatus*

*liftdirection*

*liftdoorstatus*

*floordoorstatus*

**INVARIANTS**

*inv1* :  $floordoorstatus \in LIFT \rightarrow (FLOOR \rightarrow DOORS)$

$\forall l \cdot l \in LIFT \wedge liftdoorstatus(l) = CLOSED$

*inv2* :  $\Rightarrow$

$floordoorstatus(l)(liftposition(l)) = CLOSED$

The floor door opens AFTER the lift door opens

$\forall l, f \cdot l \in LIFT \wedge f \in FLOOR \setminus \{liftposition(l)\}$

*inv3* :  $\Rightarrow$

$floordoorstatus(l)(f) = CLOSED$

Floor doors may be OPEN only on the floor where a lift is stopped

**THEOREMS**

$\forall l, f \cdot l \in LIFT \wedge f \in FLOOR \wedge liftstatus(l) = MOVING$

*thm1* :  $\Rightarrow$

$floordoorstatus(l)(f) = CLOSED$

If a lift is MOVING then the floor door for that lift is CLOSED on all floors

$\forall l \cdot l \in LIFT \wedge floordoorstatus(l)(liftposition(l)) = OPEN$

*thm2* :  $\Rightarrow$

$liftdoorstatus(l) = OPEN$

Floor door OPEN implies lift door OPEN

**EVENTS****Initialisation**

**begin**

$liftposition, liftdirection, liftstatus : |$   
 $liftposition' \in LIFT \rightarrow FLOOR$   
 $\wedge liftdirection' \in LIFT \rightarrow DIRECTION$   
 $\wedge liftstatus' \in LIFT \rightarrow STATUS$   
**act1** :  $\wedge (\forall l \cdot l \in LIFT \wedge liftposition'(l) = 0$   
 $\Rightarrow$   
 $liftdirection'(l) = UP)$   
 $\wedge (\forall m \cdot m \in LIFT \wedge liftposition'(m) = MAXFLOOR$   
 $\Rightarrow$   
 $liftdirection'(m) = DOWN)$

There seems to be a bug in Rodin. The two quantifications, in  $l$  and  $m$ , are interfering. See PO Initialisation FIS!! Also, if the quantification over  $m$  is replaced by quantification over  $l$ , which of course is legal, a syntax error is diagnosed!

**act2** :  $liftdoorstatus := LIFT \times \{CLOSED\}$   
**act3** :  $floordoorstatus := LIFT \times \{FLOOR \times \{CLOSED\}\}$

end

**Event**  $OpenFloorDoor \hat{=}$

any

$lift$   
 $floor$

where

**grd1** :  $lift \in LIFT$   
**grd2** :  $floor = liftposition(lift)$   
**grd3** :  $liftdoorstatus(lift) = OPEN$

then

**act1** :  $floordoorstatus(lift) := floordoorstatus(lift) \Leftarrow \{floor \mapsto OPEN\}$

end

**Event**  $CloseFloorDoor \hat{=}$

any

$lift$   
 $floor$

where

**grd1** :  $lift \in LIFT$   
**grd2** :  $floor = liftposition(lift)$   
**grd3** :  $floordoorstatus(lift)(floor) = OPEN$

then

**act1** :  $floordoorstatus(lift) := floordoorstatus(lift) \Leftarrow \{floor \mapsto CLOSED\}$

end

**Event**  $OpenLiftDoor \hat{=}$

**extends**  $OpenLiftDoor$

```

    begin
        skip
    end
Event CloseLiftdoor  $\hat{=}$ 
extends CloseLiftdoor
    when
        grd3 : floordoorsstatus(lift)(liftposition(lift)) = CLOSED
    then
        skip
    end
Event StopLift  $\hat{=}$ 
    Models a lift arriving at a floor and stopping
extends StopLift
    begin
        skip
    end
Event StartLift  $\hat{=}$ 
    Models the starting of a STOPPED lift, maintaining of previous direction
extends StartLift
    begin
        skip
    end
Event ChangeDir  $\hat{=}$ 
    Models the changing of direction of a STOPPED lift
extends ChangeDir
    begin
        skip
    end
Event MoveUp  $\hat{=}$ 
    Models a lift moving up to the next floor
extends MoveUp
    begin
        skip
    end

```

```
Event MoveDown  $\hat{=}$   
    Models a lift moving down to the next floor  
extends MoveDown  
    begin  
        skip  
    end  
END
```

**MACHINE** LiftWithButtons

**REFINES** LiftPlusFloorDoors

**SEES** Lift\_ctx, Doors\_ctx

**VARIABLES**

*liftposition*

*liftstatus*

*liftdirection*

*liftdoorstatus*

*floordoorsstatus*

*extrequests*

*intrequests*

**INVARIANTS**

*inv1* :  $extrequests \in FLOOR \rightarrow \mathbb{P}(DIRECTION)$

*inv2* :  $intrequests \in LIFT \rightarrow \mathbb{P}(FLOOR)$

**EVENTS**

**Initialisation**

*extended*

**begin**

*act4* :  $extrequests := FLOOR \times \{\emptyset\}$

*act5* :  $intrequests := LIFT \times \{\emptyset\}$

**end**

**Event** *ChooseDirection*  $\hat{=}$

**any**

*floor*

*dir*

**where**

*grd2* :  $floor \in FLOOR$

*grd1* :  $dir \in DIRECTION$

**then**

*act1* :  $extrequests(floor) := extrequests(floor) \cup \{dir\}$

**end**

**Event** *ChooseFloor*  $\hat{=}$

Models internal choice of floor in lift

**any**

*lift*

*floor*

```

where
  grd2 : lift ∈ LIFT
  grd1 : floor ∈ FLOOR
then
  act1 : intrequests(lift) := intrequests(lift) ∪ {floor}
end

Event StopLift ≐
  Models a lift arriving at a floor and stopping
extends StopLift
  begin
    skip
  end

Event StartLift ≐
  Models the starting of a STOPPED lift, maintaining of previous direction
extends StartLift
  begin
    skip
  end

Event ChangeDir ≐
  Models the changing of direction of a STOPPED lift
extends ChangeDir
  begin
    skip
  end

Event OpenFloorDoor ≐
extends OpenFloorDoor
  begin
    skip
  end

Event CloseFloorDoor ≐
extends CloseFloorDoor
  begin
    skip
  end

```

```

Event MoveUp  $\hat{=}$ 
    Models a lift moving up to the next floor
extends MoveUp
    begin
        skip
    end

Event MoveDown  $\hat{=}$ 
    Models a lift moving down to the next floor
extends MoveDown
    begin
        skip
    end

Event OpenLiftDoor  $\hat{=}$ 
extends OpenLiftDoor
    begin
        skip
    end

Event CloseLiftdoor  $\hat{=}$ 
extends CloseLiftdoor
    begin
        skip
    end

END

```