Übungen zur Vorlesung
Safety-Critical Systems I
Sommersemester 2007

AG BS
Jan Peleska
Christof Efkemann

**Blatt 3**

# Modelling Safety-Critical Controllers With Temporal Logic

### Facing the Inevitable for Safety Specialists: The Elevator

**Equipment Under Control.** The EUC is an **elevator** that operates the floors $\{0, 1, \ldots, max - 1\}$ of a building. Its physical state space is represented by vectors of the following component variables:

1. A sensor transmits the door state which can be open or closed or in the process of being opened or closed: $door : \{opening, open, closing, closed\}$
2. The door movement can be controlled using the interface $doorCtrl : \{doOpen, doClose\}$
3. A sensor tells whether somebody is between the doors: $doorBlocked : bool$
4. The elevator is in movement states $move : \{up, down\}$ – also if it is not moving it has a movement state *up* or *down*.
5. The movement can be controlled via interface $moveCtrl : \{goUp, goDown\}$
6. The elevator speed can be sensed via $speed : \{zeroSpeed, speeding, breaking\}$
7. The elevator speed can be controlled via $speedCtrl : \{stop, speedUp, break\}$
8. A sensor tells when the next floor is approaching: $near : bool$. This is used to start breaking the elevator in time for the next floor if it has to stop there. *near* switches back to $false$ when the next floor has been reached.
9. Another sensor at each floor transmits the floor number which has been reached: $floor : \{0, 1, \ldots, max-1\}$. This variable value remains unchanged until the next floor has been reached.
10. An interface $request : \{0, 1, \ldots, max - 1\}$ where users can request the elevator to stop at a given floor.

**User Requirements.** Users press buttons $request : \{0, 1, \ldots, max - 1\}$, in order to make the elevator stop at the requested floor. The elevator moves to the requested floors according to the following strategy:

1. Never change the movement direction until the last requested floor in that direction has been reached.
2. Every requested floor shall finally be reached.
3. If no requests have to be processed the elevator stays where it is with doors open.

**Safety Requirements.** The elevator operates safely if the following requirements are fulfilled:

1. Doors must not close if somebody is between them. If they are in the process of closing when something comes in between, they must be opened again immediately.
2. Door must be closed if elevator is moving.
3. After speeding, the elevator must first break before stopping altogether.
4. The elevator may only stop at a requested floor if it still has time for breaking. As a consequence, a request for floor $n$ is only recorded but does not lead to an immediate stop, if the elevator already approaches floor $n$ and the *near* sensor signals *true*.
5. When on top-floor $max - 1$ and not halted, the movement must not be *up*.
6. When in basement 0 and not halted, the movement must not be *down*.

**The Controller** reads the following observables:

1. Door state *door*
2. Door-blocked sensor *doorBlocked*
3. Movement state *move*
4. Speed *speed*
5. Sensor *near*
6. Sensor $floor$
7. Request interface *request*

The controller writes to the following control interfaces:

1. Door control *doorCtrl*
2. Movement control *moveCtrl*
3. Speed control *speedCtrl*

It manages an internal data structure for recording the floors to stop at, for storing the movement direction, the requested speed and the actual floor. The latter (redundant) information can be used to implement a **safety monitor** which can detect some aspects of abnormal behaviour (e. g. sensor or movement errors) which shall always lead to an immediate STOP of the elevators – the doors stay closed – the passengers first get nervous and then fall in love with each other,

## Aufgabe 1: System Specification

1. Draw a block diagram showing all the interfaces.
2. Specify the properties of the EUC using temporal logic: This involves the relationship between control interfaces and controlled state variables and sensor information and actual state.

   Examples:

   - *If a doClose command is given to an open door, it starts closing and then will finally be closed if is not blocked before.*
   - *If the actual floor is $n$, the movement is up and the speed is not zeroSpeed, then finally we will have sensor near = true and after that finally sensor $floor = n+1$.*
3. Specify the user requirements using temporal logic.
4. Specify the safety requirements using temporal logic.

Give a textual version for each logical formula.

## Aufgabe 2: Controller Code

Program the controller according to the safe I/O-TDIOHS paradigm. The controller shall ensure safety and user requirements and also include a safety monitor.

## Aufgabe 3: Safety Proof

Prove the safety of controller code with the degree of formality shown in the lecture notes.

## Aufgabe 4: User Requirements Verification

Give an informal textual argument why your controller implementation fulfils the user requirements

**Abgabe: Bis Dienstag, 19.06.2007, per Mail.**