Student information

*Ivo Willemsen*

*851926289*

**System Verification and Testing / Report Assignment 1 + 2**

|  |  |
| --- | --- |
|  | Task 2.1 I have provided different UPPAAL xml definition files for several properties to be tested. Reason for this being, that some properties asked for more rigorous testing than just three floors and two elevators. It was also pointed out during the lecture that the scenario of 11 floors and 3 elevators would be tested to check whether the floor-to-elevator assignment algorithm works correctly. Also, the property which tests some time constrains requires more elaborate settings of constants in order to test the cases thoroughly.  Instead of describing or commenting on the formalization in the comment section of UPPAAL, I prefer to do it here as it is easier to read for the examiner. Property 1 File: elevator\_1236.xml  The elevator control unit is deadlock-free  No comments Property 2 File: elevator\_1236.xml  The elevator never travels with its door open  A[]  (  not  (  (engine(0).moving\_up or engine(0).moving\_down)  and  (cabin\_door(0).opened or protecting\_door(0).opened)  )  )  This is clearly a safety property which should hold always globally. It could have been written by using an implication:  A[]  (  (engine(0).moving\_up or engine(0).moving\_down)  imply  (cabin\_door(0).closed and protecting\_door(0).closed)  )  But the first formalization matches the specification better. Property 3 File: elevator\_1236.xml  The amount of requests can reach the threshold value (in the description of Task 1 the value is 5) when the elevator is on the ground floor This is clearly an “EF” formula that needs to be applied, as it should hold possibly (can) eventually. The “when” could be misleading, but I interpreted the “when” as a “while”, i.e. a temporal indicator and not a causal indicator: *While* the elevator is on the floor, the amount of requests can reach the threshold value E<>  (  (dgc\_request\_handler(0).len == (MAX\_SIZE)  and  (main\_control(0).current\_floor == 0))  ) Property 4 File: elevator\_4.xml  Each floor (except the ground floor) can be reached, and only by the correct elevator The default floor number generator is based on non-deterministic behavior by mean of a “select” UPPAAL statement. So, there can never be complete certainty regarding the visit of an elevator to a certain floor. Therefore, the default behavior of the button was changed and replaced by a deterministic rotating schema, where every floor number is generated at least once. In that case, the following formula suffices:  A[]  (  (main\_control(0).dest\_reached imply  (main\_control(0).current\_floor == 0 or  (main\_control(0).current\_floor >= 1 and  main\_control(0).current\_floor <= 3 and  button.generated[main\_control(0).current\_floor] == true))) and  (main\_control(1).dest\_reached imply  (main\_control(1).current\_floor == 0 or  (main\_control(1).current\_floor >= 4 and  main\_control(1).current\_floor <= 7 and  button.generated[main\_control(1).current\_floor] == true))) and  (main\_control(2).dest\_reached imply  (main\_control(2).current\_floor == 0 or  (main\_control(2).current\_floor >= 8 and  main\_control(2).current\_floor <= 10 and  button.generated[main\_control(2).current\_floor] == true)))  )  The red-highlighted lines are not necessary, because the deterministic rotating schema ensures monotonically increasing floor number, without skipping. But for completeness reasons, an array of Booleans was created, with the index being the floor number. The respective index is made true every time the floor number is generated and checked in the formula. Also note that trips to floor 0 are ignored.  Although an “AG” formula is used, the semantics are the same. The above stated, formally checks that always and globally, **IF** the destination is reached, the correct floor is involved. Property 5 File: elevator\_5.xml  All floor requests are eventually served, i.e., the elevator reaches its destination.  This formula on the next page checks the liveness property in question: When a request is made, then it needs to be served. Basically, it checks that no requests are lost in the process.  In the setup, there are 4 floors and 2 elevators. Floors 0 and 1 are served by elevator 1 and floors 2 and 3 by elevator 2. First, just as with property 4, the non-deterministic floor number generator (button template) was replaced by a monotonically increasing, deterministic floor number generating. But for another reason. The reason here is that two subsequent requests are detected by the main controller and the latter is skipped. Another option would have been changing the main controller template.  (  ( dgc\_request\_handler(0).request\_received and  (  dgc\_request\_handler(0).req\_floor == 0 or  dgc\_request\_handler(0).req\_floor == 1  )  ) or  (  dgc\_request\_handler(1).request\_received and  (  dgc\_request\_handler(1).req\_floor == 2 or  dgc\_request\_handler(1).req\_floor == 3  )  )  )  -->  (  main\_control(0).dest\_reached and  (main\_control(0).current\_floor == 0 or  main\_control(0).current\_floor == 1)  or  main\_control(1).dest\_reached and  (main\_control(1).current\_floor == 2 or  main\_control(1).current\_floor == 3)  ) |
|  |  |