Correctness by Construction

Third Event-B Exercise: Room for One

Deadline: Wednesday, May 4th 2022, 15:59

Manuel Carro
manuel.carro@upm.es

April 19th, 2022

1 Problem Description

We have to design a system to control the access to a room where only one person can be at a time. The room has an external status light that can be red (meaning that the room cannot be accessed) or green (meaning the opposite). A sensor at the entrance door can detect when someone is waiting. A schematic representation appears in Figure 1.

Those entering the room are expected to follow some common-sense rules. A non-comprehensive list would be: the status light is respected; people stand in front of the door to wait for the light to turn green; only one person stands in front of the door; only one person enters the room at a time; anyone waiting at the entrance door does not walk away, but eventually enters the room when the status light indicates that this is possible.

If necessary, you may introduce behavior that can be reasonable in a realistic environment. If so, you should explain and motivate it. If the rules you introduce simplify the problem too much, you may not get full marks, even if your model completely implements them.

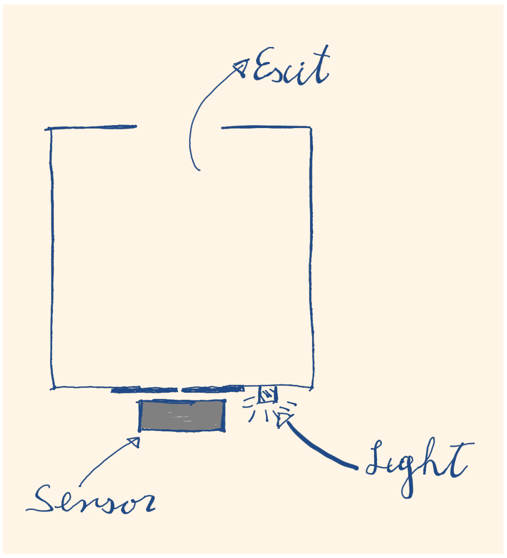

Figure 1: Room.

The inside of the room cannot be seen from the outside. All communication takes place through the external status light. The person inside the room can exit at any time through a door different from the entrance door. There is no exit sensor: when the person in the room leaves it, the corresponding event(s) adjust(s) the model variables appropriately.[1]

1.1 The Sensor

The sensor works in a way similar to the one that we presented in the *One-Way Bridge* example. When a person steps on the sensor, its signal goes from the level *off* to the level *on*. When someone standing on the sensor walks away, the level goes back to *off* again. See Figure 2 for a depiction.

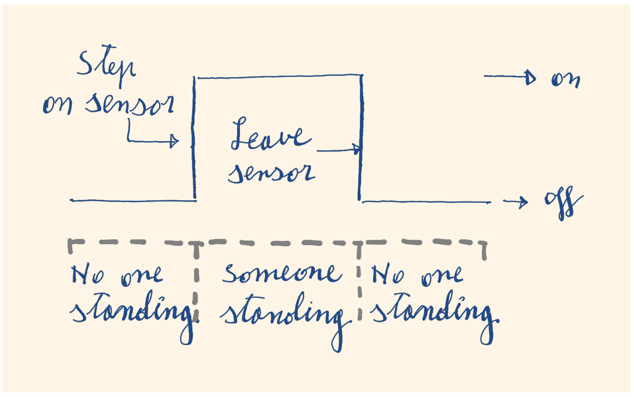

Figure 2: Behavior of the sensor when someone stands on it. Time goes from left to right.

1.2 Separation of Concerns

In order to have a model that can be implemented, the controller cannot manipulate data that models the environment. Likewise, the events that model the environment cannot (directly) read / change data that belongs to the controller. The final design should have a clear separation between variables and events that model the behavior of the environment and those that model the controller.

The events that model the behavior of persons can change the sensor state, because that change is triggered by a person standing on or leaving the sensor. They can also read that state, because that corresponds to physically seeing a person standing on it, although you may also have a specific variable to denote a person standing on a sensor if you think that this is cleaner. They can react to the color of the status light, because people can see it — but they cannot change it.

The events that model the controller can read the sensor status, read and set the status light, and read/write any additional variables necessary to keep an internal state. They cannot set variables that respond to action by the persons (e.g., the sensor state). They cannot read or set variables that are only necessary to model the behavior of the persons.

An exception to the last constraint refers to the event(s) that model a person leaving the room. As mentioned before, you do not need to fully model the communication environment / controller in that case. It would be unfeasible, since there is no sensor that would be necessary for that communication. Therefore, the variables that model the behavior / state of a person inside the room can be shared by the controller event(s) and the event(s) that model a person leaving the room, and read/set by both.

[1]If there were an exit sensor, the logic to handle it would be similar to the one for the entrance. For learning purposes, there is no point in repeating it. However, the absence of that sensor would make that system not controllable in a real situation, since there would be no way to detect that a person leaves the room.

1.3 Classification of Variables

In order to make it easier to follow the guidelines in the previous section, it is advisable to establish clearly the category of every variable in the model: it either models the environment, models the controller, or acts as communication channel (which is then written by one and read by the other). Every variable needs to have a clear *meaning* in the model: what it represents and what it is used for. Respect this idea throughout the development of the model and it will help enormously to avoid mistakes.

1.4 Processing Speed of Hardware Equipment

Real apparatuses and software take some time to process signals and actuate on external devices. From the point of view of human reaction time, that is often so fast that it may be safely assumed to be immediate. For example, from the moment one person sets foot on a sensor to the moment the controller detects it, that person does not have time to do anything.

We are decomposing a system in separate, concurrent events that do not have a notion of (absolute) time. Interleavings that would not happen when humans are involved may be possible in the specification.[2] If necessary, you can add additional variables to block (environment) events that would not be seen in real life and disallow schedulings that would not take place due to the speed difference between humans and hardware devices.

However, these variables must not unnecessarily serialize the design, which has to be concurrent by nature. For example, a person may walk to the door and stand at the entrance of the room while another person is inside or exiting. This (and others) is a sequence of events that has to be allowed.

1.5 Some Recommendations

- The system can be designed using a single Event-B model. However, progressive refinements may help separate concerns and consider requirements incrementally.

- You can follow an approach similar to that of the One-Way Bridge. But, since the behavior of the room is simpler (there is only one area to control and a maximum of one person), large simplifications can be made.

- Remember to add invariants / guards to capture the problem requirements (Section 2). You will surely need additional invariants to help prove *other* invariants or refinements.

- The model can be small. Depending on how you design it, only a few events / variables / invariants are necessary. The proofs (again, depending on your particular model) are easy: one or two may need invoking directly the predicate prover (PP) in the theorem prover. However, coming with the model may be delicate. The interplay of events makes it necessary to think carefully the role of every variable, and stick to it.

- Since only one person can be in the room, it is tempting to use a BOOL to model room occupancy. That may not be a good choice, because controlling a violation of the maximum occupancy requirement will not be straightforward then. A number will be more appropriate to capture these violations.

[2] This is not bad, as if we were modeling a system purely based on software, these interleavings might very well happen and, if they are problematic, they would need to be detected.

2 Requirements

FUN 1	This system deals with the access control of a room.

SAF 2	No more than one person can be in the room.

FUN 3	The system must not deadlock.

EQP 4	There is a status light outside the room with two colors: green and red.

FUN 5	When the status light is red, the room cannot be accessed. When the status light is green, the room can be accessed.

EQP 6	There is a presence sensor at the entrance of the room.

FUN 7	The presence sensor produces an *on* signal when a person is standing on it and an *off* signal otherwise.

FUN 8	A person inside the room can leave at any moment (using a door different from the one used to enter the room).

ENV 9	The inside of the room cannot be seen from its outside and vice-versa.

ENV 10	People obey the status light.

FUN 11	Anyone wishing to enter the room must step on the sensor, and wait there until the status light is green, if it is not already.

ENV 12	Anyone who stands on the sensor will wait there for the status light to turn green and enter the room.

3 Tasks

You have to:

1. Develop an Event-B model that captures the requirements in Section 2 and the general description in Section 1. Some events and variables in the model must be designed so that they can be used to derive the implementation of a controller.

2. Write a **short** document that classifies the variables in the model as those that capture the environment, those that belong to the controller, and those that act as communication channels (specifying the direction). Describe briefly the meaning of each of these variables. Explain how the requirements have been taken care of. If you added additional behavior rules, state them in this document, as well as the reasons why they are necessary and how your model takes care of them.

4 To Be Turned In

All the components mentioned below must be sent to manuel.carro@upm.es by the deadline (see the first page of this document):

- The Event-B model mentioned in point 1 of Section 3, developed using Rodin. Export it to a zip file and send that zip file as a mail attachment. Name the model OnePersonRoom_XYZ, where XYZ are your initials. If you have two family names, please use the initials for both.

- Please submit the document mentioned in point 2 of Section 3 in PDF format. You can hand-write it **very clearly** and scan it or take a **good** picture and convert it to a PDF document. Please do not store it as part of the zip file where you pack the Event-B model; send it as a separate attachment instead. Similarly to the model, please name it OnePersonRoom_XYZ.pdf.