

Correctness by Construction

Third Event-B Exercise: A Doctor's Office

Deadline: Tuesday, April 30th 2024, 23:59

Manuel Carro

manuel.carro@upm.es

April 8th, 2024

1 Problem Description

We have to design a system to control the access to a doctor's office, where only one patient can be at a time. The office has an external status light that can be red (meaning that the office cannot be accessed) or green (meaning the opposite). A sensor at the entrance door can detect when someone is waiting. A sensor at the exit detects when a patient heading out of the office walks past it. The doctor has a button to signal the s/he is ready to accept a new patient.

The inside of the office cannot be seen from the outside, and the other way around. All communication takes place through the external status light. Patients to be attended should wait until the status light is green to enter the office. When the light is red, patients waiting stand on the sensor. The persons inside the office can leave at any time through an exit door different from the entrance door.¹ Figure 1 shows a schematic representation of the office.

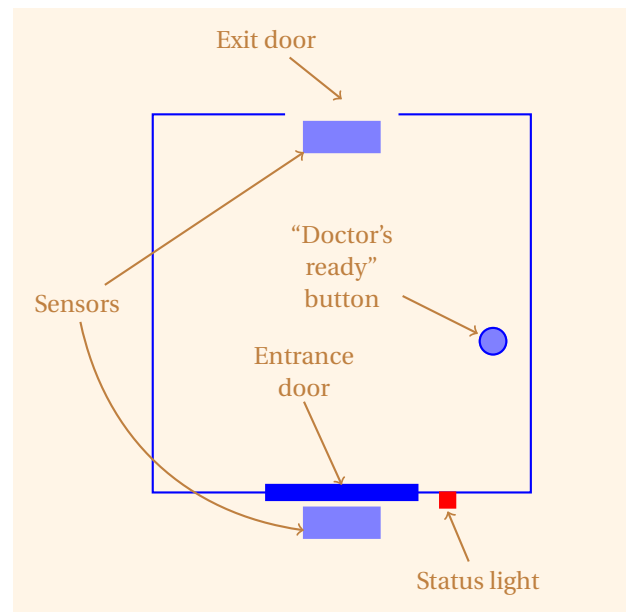


Figure 1: Doctor's Office.

¹In real life, patients would wait for the doctor to examine them. To simplify, patients can leave whenever they wish to do so.

Patients are expected to follow some common-sense rules, among which we can list:

- The status light is respected.
- Patients stand in front of the door waiting for the light to turn green.
- Only one patient stands in front of the door.
- Only one patient enters the office at a time, and only when the light is green.
- Anyone waiting at the entrance door does not walk away, but eventually enters the office when the status light indicates that this is possible.

The office has a button / web interface / ... that doctors press to let the controller know that they are ready to accept another patient. This may happen when the patient is still in the office and getting ready to leave, or any arbitrary amount of time after the patient has left (the doctor may have to take care of administrative tasks). In any case, a new patient should not be able to enter the office until the previous patient has left the office **and** the doctor has signaled that they are ready to attend a new patient.

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.

2 Additional Information

2.1 The Sensors

The sensors work similarly to what we presented in the [One-Way Bridge](#) example. When a person steps on the sensor, its state (which we access through a variable in the model) goes from *off* to *on*. When someone standing on the sensor walks away, the state returns to *off* (see Figure 2).

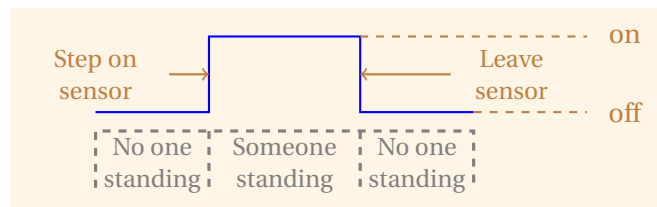


Figure 2: Behavior of the sensor when someone stands on it. Time goes from left to right.

2.2 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. Please state which events and variables pertain to the environment simulation and to the controller. It is advisable to adopt a naming scheme that reflects this. Also, every variable needs to have a clear *meaning* in the model: what it represents and what it is used for. Respect this meaning idea throughout the development of the model and it will help avoid mistakes.

The events that model the behavior of patients can:

- Change the sensor state, because that change is triggered by a patient standing on or leaving the sensor.
- Read the sensor state, because that corresponds to physically seeing a patient standing on it. You may also have a specific (environment) variable to denote a patient standing on a sensor if you think that this is cleaner.
- See and react to the color of the status light, but not change it.

The events that model the controller can:

- Read the sensor status, because it would be (physically) connected to the controller.
- Read and set the light status.
- Read and write any additional variables necessary to keep an internal state.

However, they cannot:

- Set variables that respond to actions by patients (e.g., the sensor state).
- Read or set variables that are only necessary to model the behavior of the persons.

An exception to this division of concerns comes from the need to model that the hardware / software is fast enough to register and process signals from the sensors. See Section 2.3.

2.3 Processing Speed of Hardware Equipment

Real hardware / software takes some time to process signals and actuate on external devices. But this is often so fast that it is safe to assume it is immediate from a human point of view. In our case, we will assume the software and hardware is so fast that the signals generated by the sensor are received and treated by the controller before they change, and they are not lost (because, e.g., someone walks very fast past a sensor).

Since we are not representing (real) time in our models, a simple specification can allow interleavings that would not take place when humans are involved, due to the huge speed difference between persons and machines.² To prevent these unreal interleavings to be considered by the theorem provers, you can add additional variables to block / delay (environment) events that would not immediately take place in real life and disallow schedulings that would not happen due to the aforementioned speed difference.

In particular, it is admissible to include the variables and logic necessary to reflect that events corresponding to a patient leaving the sensor can only be enabled after the *on* signal has been registered by the controller. In the real world we cannot enforce this, but if we assume that it is what happens in reality, enforcing it in the model does not violate reality.

However, **this logic must not unnecessarily serialize the design**, which has to be concurrent by nature. For example, a person may be standing at the entrance of the office while another person is inside and/or exiting and the doctor is pressing the “next patient” button. These events may take place in any order, and none of these orders should be removed.

²This 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.

2.4 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. However, the behavior of the office is simpler (there is only one area to control and a maximum of one person) and considerable simplifications can be made.
- Use invariants / guards to capture the problem requirements. In the model, please add comments stating which requirement in Section 3 each guard / invariant is supposed to address, when they address requirements directly.³
- Since at most one patient can be in the office at any time, it is tempting to use a BOOL to model this. That may not be a good choice, because controlling a violation of the maximum occupancy requirement will not be straightforward then. A number with suitable bounds will be more appropriate (and less error-prone) to capture these violations, and easier to modify if the requirements of the problem change.
- The proofs are likely to be rather easy. Some of them may need invoking directly the **PP** predicate prover. However, coming up with the right model may be delicate. The interplay of events makes it necessary to think carefully the role of each variable, and stick to it.

3 Requirements

FUN 1	This system deals with the access control of a doctor's office.
FUN 2	The system must not deadlock.
SAF 3	At most one patient can be in the office at any given time.
EQP 4	Doctors have a device which is used to signal the controller when they are ready to attend a new patient.
FUN 5	After a patient has entered the office, doctors can signal at any time that they are ready to see a new patient.
EQP 6	There is a status light outside the office with two colors: green and red.
FUN 7	When the status light is red, patients cannot enter. When the status light is green, patients can enter.

³You will surely need auxiliary invariants to help prove some of the invariants / refinements that are used to capture the requirements.

FUN 8	The status light can be green only when there is no patient in the office and the doctor has signaled that they are ready to see a new patient.
ENV 9	People obey the status light.
EQP 10	There is a sensor at the entrance of the office that detects when a person enters.
EQP 11	There is a sensor at the exit of the office that detects when a person leaves. The exit of the office is different from and independent of the entrance of the office.
FUN 12	The sensors produce an <i>on</i> signal when a person is standing on / in front of them and an <i>off</i> signal otherwise.
FUN 13	Anyone wishing to enter the office must step on the sensor, and wait there until the status light is green, if it is not already.
ENV 14	Anyone who stands on the sensor will wait there for the status light to turn green and enter the office.
FUN 15	A patient inside the office can leave at any moment.
ENV 16	The inside of the office cannot be seen from its outside and vice-versa.

4 Tasks

You have to:

1. Develop an Event-B model that captures the requirements in Section 3 and the general description in Section 1. Make sure that the events and variables are clearly identified as belonging to the environment or to the controller (Section 2.2).
2. If you have made any changes or additions to the behavior rules, write a **short** document stating them and the motivation to propose these changes. Make sure that your model takes care of them. If you have new requirements, give them a number/name to identify them and make it easier to refer to them in the comments of the model.

5 To Be Turned In

The documents / files mentioned below must be sent to manuel.carro@upm.es by Tuesday, April 30th 2024, 23:59:

- The Event-B model mentioned in point 1 of Section 4, developed using Rodin. Ideally, all the proof obligations should be discharged.
 - Name the model **DoctorOffice_ XYZ**, where **XYZ** are your initials. If you have two family names, please use the initials for both.
 - Export it to a zip file and send me that zip file as a mail attachment.
- If you suggested changes to the problem description, please submit the document mentioned in point 2 of Section 4 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 **DoctorOffice_ NewReqs_ XYZ.pdf**.