CENG424 - Logic for Computer Science
Fall 2018

Middle East Technical University
Uluç Saranlı

Homework 3 - Design and Verification of a Controller for a Mail Delivery Robot

Assigned - Nov 19, 2018, **Due - Nov 28, 2018 23:55**

1. Description of the Problem

In this homework, you will specify, implement and check a controller for a simple mail delivery robot operating on a floor with 3 offices using the NuSMV model checker.

Each office on this floor has a button that calls the robot. On the robot itself, there are three buttons to request delivery to each one of the three offices. Your implementation should include an office module with three instances (office1, office2, office3) and a robot module with only a single instance (robot). These instances should be created in the main module and appropriately connected together.

The robot module should define the following signals:

- **location** - This signal indicates the current location of the robot and can take one of the following values: { at1, btw12, at2, btw23, at3 }, describing situations where the robot is either in an office or between offices.
- **lid** - describes the status of the lid on the robot's compartment for storing mail.
- **moving** - a Boolean variable that is true when the robot is in motion.
- **direction** - indicates whether the robot is moving forward or backward in a corridor. This signal should take values from the domain {forward, backward}.
- **sensor** - Boolean signal describing whether somebody is in the process of putting mail in the robot's compartment. This will be necessary to prevent the robot from closing the lid when somebody's hand is in the compartment.

In contrast, the office module should define the following signals:

- **office_button** - Shows the status of the button within an office. It takes on one of the following values {on, blink, off }.
- **robot_button** - This is a boolean signal that shows the status of the button that requested this office on the robot. For example, when a person requests delivery to a office2 by pressing a button, office2.robot_button becomes true.

2. Specification

Each of the specifications below describes a property of the robot controller. Translate each of these into an equivalent CTL formula or LTL formula (indicating which one you used). These specifications should be implementation independent in that they should only refer to the signals defined above.

1. When the `office_button` signal in any office is on or blinking, it will stay so until the robot reaches that office and its lid opens up.
2. When the lid on the robot opens within any office, it stays open for at least three time units.
3. If `office_button` is not off, it will be blinking when the robot is moving and it will be on when the robot is stopped.
4. If the lid is open on the robot, it must stay open until the sensor has been false for at least two time units. The lid can close after one time unit following this condition.

5. When the lid is open, the sensor will eventually be false for two consecutive time units.
6. If the robot is requested from within an office, it will eventually reach that office.
7. When the robot lid is open, the robot must not move and both buttons for that office must be off.
8. The robot must not move at the first time unit after the lid is closed.
9. It takes the robot two or three time units to move between offices (i.e. after one time unit of moving it will be between offices and within one or two extra time units, it will be in front of the next office.)
10. People can put mail in the robot compartment (making the value of the sensor true) only when the lid is open.
11. If there are no requests from any other office, the robot should not move.
12. The robot cannot change direction between offices.
13. If the robot is in front of the second office and there are requests from both the first and third offices, the robot will continue moving in its current direction.
14. If the robot is in front of a requested office, it will not leave this office before the lid is open.
15. It is possible that there are requests for the first and third offices at the same time and the robot chooses to go to the third office first.
16. It is possible that there are requests for the first and third offices at the same time and the robot chooses to go to the first office first.

You should include these specifications in your SMV implementation.

3. **Additional Specifications**

Come up with 3 more specifications that you think make sense for this system. They should not be just minor derivatives of the ones above, but original and checking different aspects of the robot controller. Include both English descriptions and CTL or LTL encodings of these specifications in your SMV file as comments and corresponding specification directives.

4. **Implementation**

Implement the robot system and controller described above in the NuSMV language. Your implementation should satisfy all of the specifications above, including those that are in the description section. If the above descriptions leave certain behaviors unspecified, you may choose the alternative that makes sense to you. Make sure to thoroughly document your code through comments.

You should also embed the specifications listed above as CTL or LTL specifications in your SMV source and check that your implementation indeed satisfies all of them. You should use fairness statements to ensure that the robot does not get stuck in front of a single office forever as well as avoiding other problematic simulation scenarios.

Submission

Your submission should include both a PDF report `hw3_e1234567.pdf` with your textual answers to questions explaining relevant details of your design and additional LTL or CTL specifications, as well as the SMV source file `hw3_e1234567_soln.smv` for your solution. Name You should submit a single ZIP file `hw3_e1234567.zip` including your report and the SMV solution. You should make sure your SMV source works properly with NuSMV, we will not debug your solution for you. Your SMV file should include your implementation of the robot, the controller, all of the specifications listed above and possibly more that you have come up with as well as plenty of comments to convince us that you understood what you were doing. If your implementation does not work with NuSMV when we test it, it will not be graded. Late submissions will be penalized with $10n^2$ points where n is the number of late days, rounded up.

Note: You can discuss your discoveries and knowledge with your classmates but you must write your own answers and code for all questions above. If any significant similarities are found between your answers and other homeworks, you will be audited on your understanding of your own solutions.