Late

Report System Validation Project
Modeling a Bridge's Control System
Creating a safety layer with mCRL using Labelled Transition Systems

Aimee FEROUGE 4014030
Wieger IJNTEMA 4034570
Arian STOLWIJK 4001079

Group ...

December 10, 2013

Date Performed: December, 2013
Instructor: J. A. Keiren

1 First deliverable

1.1 Introduction

As has been pointed out by the incident at the Ketelbrug between Emmeloord and Lelystad, bridges require a very precise control system. Many examples exist where bridges are not opened or closed properly, where bridge components break down or the bridge is being stuck in a particular state. The goal of this project is to design a satefy layer that can be added to the main control interface, in order to capture all possible situations and failures. Throughout the report, we make use of the following assumptions:

- If three or four sensors detect an open/closed bridge, the bridge can be considered open/closed
- If only the four sensor have a 50/50 detection on the bridge deck, the bridge should remain in its current position (open/closed)
- A barrier can only be considered to be down when the majority of the sensors detect a lowered barrier

In this section, the desired behaviour of the bridge is being determinated and defined. Section 1.2 contains the global requirements that will be met in our system. Specific interactions that can be performed by the bridge are discussed in Section 1.3, whereas Section 1.4 translates these interactions into the requirements stated earlier.

1.2 Global Requirements

Using the Project Guide[1] as a source for the desired behaviour, the following requirements are defined:

Opening the bridge

1. Switching on pre signs should be the first action when opening the bridge
2. Stop signs cannot be lit as long as the pre lights have not been lit
3. Barriers cannot be lowered if the stop signs have not been lit
4. Bridge can only be unlocked when all barriers are down
5. The deck can only lifted when both locks are unlocked

Closing the bridge

6. Bridge can only be locked when the deck is down
7. Barriers can only be up when the bridge is locked by at least

Functional requirements

1. The bridge should be able to be opened when a ship approaches
2. The bridge should be able to close in order to let cars pass
3. The first barrier to be encountered by the cars is lowered earlier than the second in order to enable cars to leave the bridge

Failure

4. The stop signs can only be lit if at least one pre signs is being lit at each side of the bridge
5. The barriers can only be lowered if both stop signs are being lit at each side of the bridge
6. If the motor is in the 'broken' status, the bridge may not be opened

[1] J. A. Keiren, System Validation (IN4387) Project Guide, VU University Amterdam, november 2013

1.3 Interactions

Interaction	Descripton	Parameters
`open`	Opens the bridge	
`close`	Closes the bridge	
`getSign`	Checks te status of a specific sign	Sign
`setSign`	Switches a sign on or off	Sign, status
`getBarrier`	Checks the status of a specific barrier	Barrier
`setBarrier`	Lowers of lifts a specific barrier	Barrier, status
`getLock`	Checks the status of a specific locking pin	Lock
`setLock`	Locks or unlocks a specific locking pin	Lock, status
`getDeck`	Checks the status of the bridge deck	
`setDeck`	Lowers or liftst the bridge deck	Status
`motorStatus`	Checks the status of the motor	

Table 1: All interactions to be performed by the safety layer of the control system.

1.4 Architecture

Three parallel processes exist being *signs*, *barriers* and *bridge*. *Signs* handles the control of the lights, whereas *barriers* handles the barriers and *bridge* the locks and the deck.
Signs is the first one to be executed when an `open` or `close` command is given. After first lighting the pre signs and then the stop signs, control is given to the *barriers* process.
Barriers makes sure all barriers are lowered if the stop signs are lit. When this is the case, *bridge* takes over.

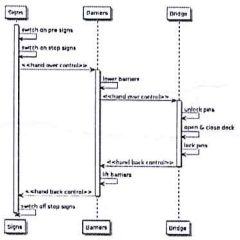

Figure 1: Sequence diagram of control flow between parallel processes *signs*, *barriers* and *bridge*.

Bridge handles the lifting and lowering of the bridge deck. Only when all barriers are down, the two locks of the bridge are removed and the deck can be moved up and down. When finishing this movement, control is given back first to *barriers* and finally back to *barriers*.
Figure 1 shows a sequence diagram of how the control shifts between these parallel processes.

1.5 Translation of Requirements in Terms of Interactions

In order to create a model for the bridge in mCRL, which is modeling software used for system validation, the requirements have to expressed in terms of the previous stated interactions. At the moment, we are still rewriting these translations after the feedback we received during last week's meeting. The following translations are from last week.

1. **Switching on pre signs are be the first operation when opening the bridge** Always when Open, then getDeck must be down and for all barriers getBarrier must be up and for all locks getLock must be locked then for all pre-signs setSign on. Else when Open, then getDeck must be down and for all barriers getBarrier must be up and one lock getLock must be locked then for all pre-signs setSign on. Else, give error and stop.

2. **Stop signs cannot be lit as long as the pre lights have not been lit** When setSign is set on for all stop-signs, then check for all pre-signs if getSign is on before, and not off intermediate. Else when setSign is set on for all stop-signs, then check for three pre-signs if getSign is on before, and not off intermediate. Else, give error and stop.

3. **Barriers cannot be closed if the stop signs have not been lit** When setBarrier is set down for all barriers, then check for all stop-signs if getSign is on before, and not off intermediate. Else when setBarrier is set down for all barriers, then check for three stop-signs if getSign is on before, and not off intermediate. Else, give error and stop.

4. **Bridge can only be unlocked when all barriers are down** When setLock is set unlocked for all locks, then check for all barriers if getBarrier is down before, and not up intermediate. Else, give error and stop.

5. **The deck can only lifted when it is completely unlocked** When setDeck is set up for the bridge deck, then check for all locks if getLock is unlocked before, and not locked intermediate. Else, give error and stop.

6. **Bridge can only be locked when the deck is down** Always when setLock(L2), then getDeck is down before, and not changed intermediately. Always when setLock(L1), then getDeck is down before, and not changed intermediately.

7. **Barriers can only be up when the bridge is locked by at least one lock** Always when setBarrier(B, up), getLock(L1) or getLock(L2) are enabled, and not disabled intermediately.

8. **Stop sign can be shut off only when the barriers are up** Always when setSign(S, off), for each B1, B2, B3, B4, getBarier should be up before, and not changed intermediately.

9. **The bridge should be able to be opened when a ship approaches** When giving the open command, somewhere in the process setDeck should be set to up.

10. **The bridge should be able to close in order to let cars pass** When giving the close command, somewhere in the process setDeck should be set to down.

11. **The first barrier to be encountered by the cars is lowered earlier than the second in order to enable cars to leave the bridge** Always when setBarrier(B2, down), getBarrier(B1) must be down or else the process is stopped. Always when setBarrier(B3, down), getBarrier(B4) must be down or else the process is stopped.

12. **The stop signs can only be lit if at least one pre signs is being lit at each side of the bridge** Always when setSign(S1, on) or setSign(S2, on), at least one status of getSign(P1) or getSign(P2) must be on. If not, the process is stopped. Always when setSign(S1, on) or setSign(S2, on), at least one status of getSign(P1) or getSign(P2) must be on. If not, the process is stopped.

13. **The barriers can only be lowered if both stop signs are being lit at each side of the bridge** Always when setBarrier(B1, down), getSign(S1) and getSign(S2) must be down. If not, the process is stopped. Always when setBarrier(B1, down), getSign(S1) and getSign(S2) must be down. If not, the process is stopped.

14. **If the motor is in the 'broken' status, the bridge should stay in the current position** Whenever motorStatus has been changed to broken and has not changed back to stopped, moving up or moving down, no components may be set anymore.

2 Second Deliverable

The full report, draft version.