Report System Validation Project
Modeling a Bridge's Control System
Creating a safety layer with mCRL2 using Labelled Transition Systems

Aimee Ferouge 4014030
Wieger IJntema 4034570
Arian Stolwijk 4001079
Group 7

December 18, 2013

Date Performed: Januari, 2013
Instructor: J. 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 safety 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 the four sensors 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 determined 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:

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

Opening the bridge

1. Stop signs cannot be lit unless the pre lights are lit
2. Barriers cannot be lowered unless the stop signs are on
3. Bridge can only be unlocked when all barriers are down
4. The deck can only lifted when both locks are disengaged

Closing the bridge

5. Bridge can only be locked when the deck is down
6. Barriers can only be raised when the bridge is locked by at least one lock

Functional requirements

7. The bridge should be able to be opened when a ship approaches
8. The bridge should be able to close and barriers should in order to let cars pass
9. The second barrier encountered by the cars is lowered after the first in order to enable cars to leave the bridge

Failure

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

1.3 Interactions

The interactions can be divided into two categories. The first category contains the global interactions, which are the direct translations of the requirements from Section 1.2. Switching of the signs, moving the barriers or engaging the locking pins are all action to be performed by the controller and can be found in Table 1 The internal interactions are used for communication purposes, combining the state perceptions of different processes. These can be found in Table 2

Interaction	Descripton	Parameters
`openBridge`	Initiates the system to open the bridge	
`closeBridge`	Initiates the system to close the bridge	
`setPre`	Switches on/off the pre sign	Sign, signStatus
`setStop`	Switches on/off the stop sign	Sign, signStatus
`setBarrier`	Lifts/lowers the barriers	Barrier, barrierStatus
`setLock`	Engages/disengages the locks	Lock, lockStatus
`setDeck`	Lifts/lowers the bridge deck	Deck, deckStatus

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

For the values of the interactions, datatypes are introduced which group certain values. Table **??** lists these datatypes and their possible values.

Interaction	Description	Parameters
`sendPre`	Sends the status of a particular pre signs	Sign, signStatus
`receivePre`	Receives status of a particular pre signs	Sign, signStatus
`commPreSign`	Synchronization action for sendPre | receivePre	
`sendStop`	Sends the status of a particular stop sign	Sign, signStatus
`receiveStop`	Receives status of a particular stop signs	Sign, signStatus
`commStopSign`	Synchronization action for sendStop | receiveStop	
`sendBarrier`	Sends the status of a particular barrier	Barrier, barrierStatus
`receiveBarrier`	Receives status of a particular barrier	Barrier, barrierStatus
`commBarrier`	Synchronization action for sendBarrier | receiveBarrier	
`sendLock`	Sends the status of a locking pin	Lock, lockStatus
`receiveLock`	Receives status of a particular locking pin	Lock, lockStatus
`commLock`	Synchronization action for sendLock | receiveLock	
`sendDeck`	Sends the status of the bridge deck	deckStatus
`receiveDeck`	Receives status of the bridge deck	deckStatus
`commDeck`	Synchronization action for sendDeck | receiveDeck	
`motorStatus`	Receives the status of the motor	Status

Table 2: All internal interactions of the safety layer of the control system.

Datatype	Values
Sign	P1, P2, P3, P4, S1, S2, S3, S4
signStatus	on, off
barrierStatus	up, down
lockStatus	engage, disengage
deckStatus	up, down

Table 3: Datatypes used as parameters for the interactions.

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.
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.

2 Translation of Requirements in Terms of Interactions

In order to create a model for the bridge in mCRL2, 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. **Stop signs cannot be lit unless the pre lights are on**
   Always after a `setSign(Si,off)`, a `setSign(Pi,on)` cannot occur unless a `setSign(Pi,on)` has

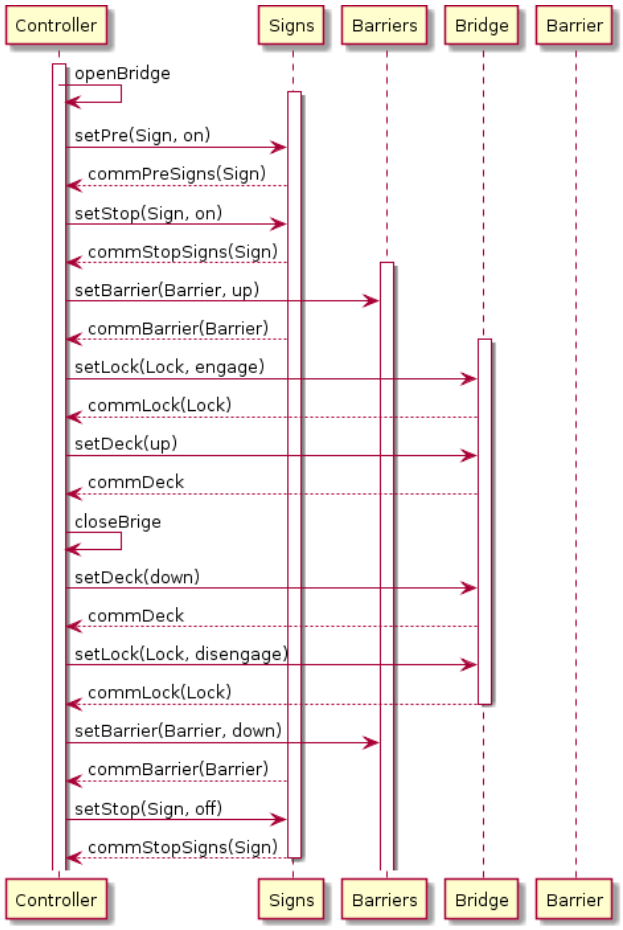

Figure 1: Sequence diagram of control flow between parallel processes *signs*, *barriers* and *bridge*.

occured intermediately.
`[true*.`

2. **Barriers cannot be lowered unless the stop signs are on** Always after a `setSign(Si,off)`, a `setBarrier(Bi,down)` cannot occur unless a `setSign(Si,on)` has occured intermediately.

3. **Bridge can only be unlocked when all barriers are down** Always after a `setBarrier(Bi,up)`, a `setLock(Li,disengage)` cannot occur unless a `setBarrier(Bi,down)` has occured intermediately.

4. **The deck can only be lifted when both locks are disengaged** Always after a `setBarrier(Li,engage)`, a `setDeck(up)` cannot occur unless a `setLock(Li,disengage)` has occured intermediately.

5. **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.

6. **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.

7. **Stop sign can be shut off only when the barriers are up** Always when setSign(S_i, off) occurs, setBarrier(B_i, up) has to be occured before for all barriers. No setBarrier(B_i, down) may have occured intermediately.

8. **The bridge should be able to be opened when a ship approaches** When open occurs, setDeck(OPEN) should eventually occur.

9. **The bridge should be able to close in order to let cars pass** When close occurs, setDeck(CLOSE) should eventually occur.

10. **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(B_2, down), setBarrier(B_1, up) has to be occured before. No setBarrier(B_1, up) may have occured intermediately. Always when setBarrier(B3, down), setBarrier(B_4, up) has to be occured before. No setBarrier(B_4, up) may have occured intermediately.

11. **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(S_1, 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.

12. **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.

13. **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.

3 Second Deliverable