

# 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

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. N

### 1.2 Global Requirements

Using the Project Guide<sup>1</sup> 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

---

<sup>1</sup>J. J. A. Keiren, System Validation (IN4387) Project Guide, VU University Amsterdam, november 2013

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 be 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 sign 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

### 1.3 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 ???. Switching of the signs, moving the barriers or engaging the locking pins are all actions to be performed by the controller. The internal interactions are used for communication purposes, combining the state perceptions of different processes (e.g. `accept(50ct)` `insert(50ct)` `-! true*(50ct)`).

Interaction	Description	Parameters
<code>openBridge</code>	Initiates the system to open the bridge	
<code>closeBridge</code>	Initiates the system to close the bridge	
<code>setStatusPreSigns</code>	Switches on/off the pre sign	Sign, status
<code>setStatusStopSigns</code>	Switches on/off the stop sign	Sign, status
<code>setStatusBarriers</code>	Lifts/lowers the barriers	Barrier, status
<code>setStatusLocks</code>	Engages/disengages the locks	Lock, status
<code>setStatusDeck</code>	Lifts/lowers the bridge deck	Deck, status

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

Interaction	Description	Parameters
sendPre	Sends the status of a particular pre signs	Sign, status
receivePre	Receives status of a particular pre signs	Sign, status
commPreSign	Synchronization action for sendStatusPreSign   receiveStatusPreSign	
sendStop	Sends the status of a particular stop sign	Sign, status
receiveStop	Receives status of a particular stop signs	Sign, status
commStopSign	Synchronization action for sendStatusStopSign   receiveStatusStopSign	
sendBarrier	Sends the status of a particular barrier	Barrier, status
receiveBarrier	Receives status of a particular barrier	Barriers, status
commBarrier	Synchronization action for sendStatusBarrier   receiveStatusBarrier	
sendLock	Sends the status of a locking pin	Lock, status
receiveLock	Receives status of a particular locking pin	Lock, status
commLock	Synchronization action for sendStatusLock   receiveStatusLock	
sendDeck	Sends the status of the bridge deck	Status
receiveDeck	Receives status of the bridge deck	Status
commDeck	Synchronization action for sendStatusDeck   receiveStatusDeck	
motorStatus	Receives the status of the motor	Status

Table 2: All internal interactions of 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. *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.

n

## 1.5 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 be 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 as long as the pre lights have not been lit** When setSign(Si,On), before setSign(Pi,On) should have occurred and setSign(Pi,Off) may not occur intermediate.
2. **Barriers cannot be closed if the stop signs have not been lit** When setBarrier(Bi,Up), before setSign(Si,On) should have occurred and setSign(Si,Off) may not occur intermediate.
3. **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.
4. **The deck can only be 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.

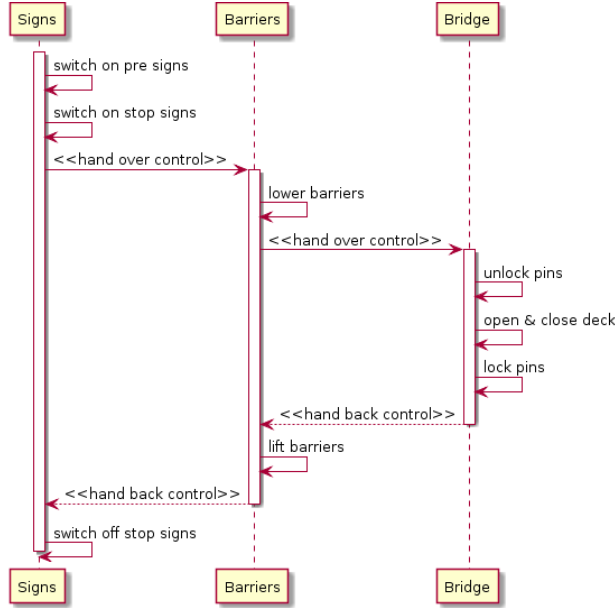


Figure 1: Sequence diagram of control flow between parallel processes *signs*, *barriers* and *bridge*.

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<sub>i</sub>, off) occurs, setBarrier(B<sub>i</sub>, up) has to be occurred before for all barriers. No setBarrier(B<sub>i</sub>, down) may have occurred 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<sub>2</sub>, down), setBarrier(B<sub>1</sub>, up) has to be occurred before. No setBarrier(B<sub>1</sub>, up) may have occurred intermediately. Always when setBarrier(B<sub>3</sub>, down), setBarrier(B<sub>4</sub>, up) has to be occurred before. No setBarrier(B<sub>4</sub>, up) may have occurred 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<sub>1</sub>, on) or setSign(S<sub>2</sub>, on), at least one status of getSign(P1) or getSign(P2) must be on. If not, the process is stopped. Always when setSign(S<sub>1</sub>, on) or setSign(S<sub>2</sub>, 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.

## **2 Second Deliverable**