System Validation Final Report

Pablo Martin Guijo (0976197)
Sri Muthu Narayanan Balasubramanian (0978367)
Nitesh Agarwal (0981729)
Prathyusha Adiraju (0977176)

October 23th 2015
Version 2.0

Contents

1 Revision History

After the first review, the following changes have been made in accordance with the comments from Prof.J.F.Groote.

1. Figure 1 was modified to include the Controller block.
2. Sections 2.1 and 3 - Explanation for the status of assumption and requirements was added respectively.
3. Section 5.2, Requirement #6, footnote added for clarity.
4. Section 8.1, Requirements #4 and #5, $\cap$ was replaced with $\wedge$
5. Section 8.2, Requirement #8 was rewritten.
6. Section 9, Details of verification setup added.
7. Punctuation errors rectified throughout the report.

2 Introduction

The purpose of this document is to design and validate an Automated Railroad Crossing System (ARCS). The goal of the system is to provide a fully automated and failsafe method for monitoring a railroad intersection to avoid accidents.
The primary functionality of the system is to monitor the intersection for the arrival and departure of trains using an array of train sensors on either side of the intersection and actuate the traffic signals, barriers, warning lights and alarms based on the data from the train sensors. The system also takes into account the failure of a train monitoring sensor, which is treated as a critical failure causing the system to enter the fail-safe mode, thereby closing the barriers and switching the signals to red until the sensors are replaced and the system is reset manually.
The system determines the speed at which the trains approach the intersection and computes the reaction time after which the signals switch and the barriers close. When multiple trains approach the intersection, the response time is determined by the earliest timer that expires. When two or more trains approach the intersection in a sequence, the behavior is designed such that only after all the trains that approach the intersection leave, the barriers open and the signals are switched to green. The rest of the document gives a formal description of the system, the system model and the verification of the requirements using the mCRL2 toolset. The complete model of the system is also presented.

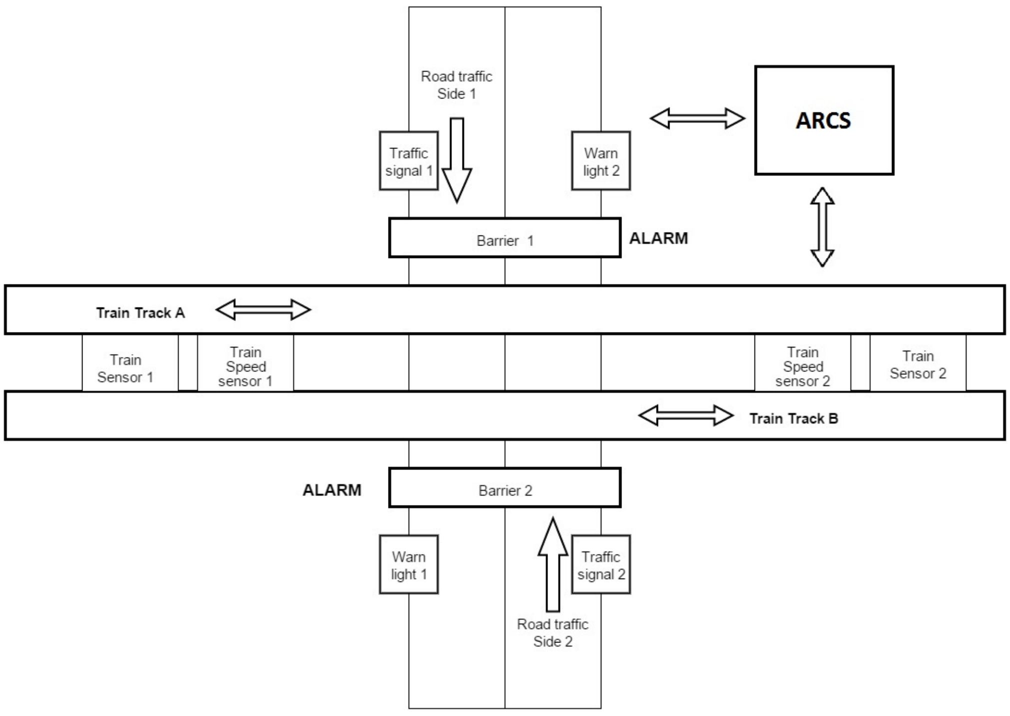


Figure 1: System Schematic Diagram

2.1 Assumptions

In order to allow for a more complete design of the system, reducing complexity and allowing easy testability, the assumptions below were made. These assumptions represent behaviours of the system that our controller will not have to prove as they are supposed to be true.

1. Two or more trains cannot arrive at the intersection at the exact same moment.

2. When a third train arrives at the intersection the barriers will have already been closed.

3. No Signals or warning lights can change, unless the ARCS's system commands the signals to change.

4. In case of any sensor failure, the sensors will be fixed manually. No trains can arrive at the intersection at the time when Manual Reset is being done after the Manual fix.

5. **The conceptual initial state** is defined as the state in which no trains have arrived, the traffic lights are green, warning lights are off, barriers are open and the bell is off.

3 List of requirements

List of general requirements specify the behavior of the system described in the introduction. The ARCS controller is designed to verify all of them. These requierements are classified based on different parts of the system, i.e., sensors, signals, bell and barriers.

3.1 Sensors

1. A sensor break can be detected at all times except, in between a sensor break and a system reset.

2. When a sensor breaks, eventually, the signals will turn “Red”, the warning lights will “Flash High”, the bell will Turn “On” and the Barriers will close.

3. After a sensor breaks and the barriers close, they will remain closed until the sensor is fixed and the system is manually reset.

4. When the system is not in the conceptual initial state, when a train arrives and the signals are green, the speed of the train is acquired, unless another train arrives or a sensor break happens in between.

5. When the train arrives and the system is in the conceptual initial state, the speed of the train is acquired unless another train arrives or a sensor break happens in between.

3.2 Signals

6. When a train approaches and there is no train at the intersection, eventually, the traffic lights should turn “Yellow” and the Warning lights should “Flash Low”.

7. When a train approaches and there is a train already present at the intersection, traffic lights will turn “Yellow” eventually, if they have not already turned “Yellow”.

8. When a train approaches and there is a train already present at the intersection, warning lights will “Flash Low” eventually, if they are already not “Flashing Low”.

9. When the timer T1 expires, eventually, traffic lights must turn “Red” and the warning lights must “Flash High”.

10. Before the barriers are closed, the traffic lights must be “Red” and the warning lights must be “Flashing High”.

11. The traffic lights must remain “Red” and the warning lights must remain “Flashing High” as long as the barriers are closed.

12. After a reset, traffic lights must eventually turn “Green” and the warning lights must eventually turn “Off”.

13. When the timer T3 expires, the traffic lights must eventually turn “Green” and the warning lights must eventually turn “Off”.

14. The traffic lights must never go from “Red” to “Yellow” without going through “Green” and warning lights must never go from “High” to “Low” without going through “Off”.

15. The traffic lights must never go from “Yellow” to “Green” without going through “Red” and warning lights must never go from “Low” to “Off” without going through “High”.

3.3 Bell

16. Before the barriers are closed, the bell must be “On”.

17. The bell must remain “On” as long as the barriers are closed.

18. The bell is eventually turned “Off” when the barriers are opened.

3.4 Barriers

19. The barriers must eventually close after the traffic lights turn “Red”.

20. The barriers must eventually open after a reset.

21. The barriers must open when every train that approached the intersection has left the intersection.

3.5 Deadlock

22. The system must be deadlock free.

4 List of external actions

The following table shows a list of all the external actions ***of the control system.***

No.	Name (data type)	Description	flow
E1	eActTrainSensData (status,TrainNo)	Notifies the status and number of trains. Status can be arrives, leaves or broken. TrainNo is **the number of trains in the intersection at any given time after this action takes place**. It is -1 when the sensor is broken	INPUT
E2	eActSetBarrier(status)	Commands the barriers to be in the given status. Status can be open or close.	OUTPUT
E3	eActSetBell(status)	Commands the bell to be in the given status. Status can be on or off.	OUTPUT
E4	eActSetSignal(status)	Commands the traffic signals to be in the given status. Status can be green, yellow or red.	OUTPUT
E5	eActSetWarnLight (status)	Commands the warning lights to be in the given status. Status can be high, low or off.	OUTPUT
E6	eActTime(time)	Notifies when the timer has expired. Time can be: time1, time2, time3, time4, notime, t0, t1, t2, t3, t4 , t5	INPUT
E7	RESET	The reset action occurs to move the system to the conceptual initial state.	INPUT
E8	MANUAL_FIX	Notifies when the sensor is broken and needs a manual fix.	INPUT
E9	RESET_COMPLETE	Notifies that the system reset is complete and system returns to conceptual initial state.	INPUT
E10	eActGetSpeed(speed)	Gets the train speed speed can be s0, s1, s2, s3, s4, s5, noSpeed	INPUT

Table 1: List of external actions

Note:
time1 is the time to wait between the traffic signals turn yellow to the time they turn red
time2 is time to wait between the traffic signals turn red and the start of closing the barrier.
time3 is the time the barrier needs to open.
time4 is the time the barrier needs to close.
notime means that an action is imminent.
t0 to t5 denote the time intervals that the system must wait depending on the speed read.

5 Requirements expressed in terms of actions

In this section, the requirements listed in section 2 will be translated in terms of the actions described in section 3.

5.1 Sensors

1. (a) If a first **eActTrainSensData(BREAK,-1)** has not happened, it can happen at any state.

   (b) After any **eActTrainSensData(BREAK,-1)**, another **eActTrainSensData(BREAK,-1)** can happen again only after a **RESET_COMPLETE.**

   (c) Between a **eActTrainSensData(BREAK,-1)** and a **RESET_COMPLETE** an **eActTrainSensData(BREAK,-1)** cannot happen.

2. After an **eActTrainSensData(BREAK,-1)** happens, after a finite number of steps, **eActSetSignal(RED), eActSetWarnLight(HIGH), eActSetBell(BELL_ON)** and **eActSetBarrier(CLOSE)** will take place.

3. After a **eActTrainSensData(BREAK,-1)**, an **eActSetBarrier(OPEN)** cannot happen until **RESET** happens.

4. When an **eActSetSignal(GREEN)** happens not followed by **eActSetSignal(YELLOW)** or **eActSetSignalAction(RED)** and an **eActTrainSensData(ARRIVE,1)** happens, **eActGetTrainSpeed** action takes place unless an **eActTrainSensData(ARRIVE,2)** happens or **eActTrainSensData(BREAK,-1)** occurs in between.

5. When **eActTrainSensData(ARRIVE,1)** takes place **eActGetTrainSpeed** action occurs unless an **eActTrainSensData(ARRIVE,2)** happens or **eActTrainSensData(BREAK,-1)** occurs in between.

5.2 Signals

6. When **eActTrainSensData(ARRIVE,1)**[1] happens, an **eActSetSignal(YELLOW)** will follow , after a finite number of steps.

7. When **eActTrainSensData(ARRIVE,2)** happens, **eActSetSignal(YELLOW)** will happen after a finite number of steps, if it has not already happened.

8. When **eActTrainSensData(ARRIVE,2)** happens, an **eActSetWarnLight (W_LOW)** will follow, after a finite number of steps, if it has not already happened.

9. After **eActTime(time1)** expires, an **eActSetSignal(RED)** and **eActSetWarnLight(W_HIGH)** must happen, after a finite number of steps.

[1] - The number 1 denotes that after this action happens, there will be 1 train in the intersection. Therefore, this means that it is the first train to approach the intersection

10. Every **eActSetBarrier(CLOSE)** must be preceded by an **eActSetSignal(RED)** and **eActSetWarnLight(W_HIGH)**, without **eActSetWarnLight(W_OFF)** or **eActSetWarnLight(W_LOW)** or **eActSetSignal(GREEN)** or **eActSetSignal(YELLOW)**happening in between.

11. In between **eActSetBarrier(CLOSE)** and **eActSetBarrier(OPEN)** the actions **eActSetWarnLight(W_OFF)** or **eActSetWarnLight(W_LOW)**or **eActSetSignal(GREEN)** or **eActSetSignal(YELLOW)** cannot take place.

12. After a **RESET** action happens, **eActSetSignal(GREEN)** and **eActSetWarnLight(OFF)** should happen , in a finite number of steps.

13. After **eActTime(time3)** expires, **eActSetSignal(GREEN)** and **eActSetWarnLight(OFF)** should happen, in a finite number of steps

14. Between actions **eActSetSignal(RED)** and **eActSetSignal(GREEN)** an **eActSetSignal(YELLOW)** should not take place and between **eActSetWarnLight(W_HIGH)** and **eActSetWarnLight(W_OFF)** an **eActSetWarnLight(W_LOW)** should not take place.

15. Between actions **eActSetSignal(YELLOW)** and **eActSetSignal(RED)** an **eActSetSignal(GREEN)** should not take place and between **eActSetWarnLight(W_LOW)** and **eActSetWarnLight(W_HIGH)** an **eActSetWarnLight(W_OFF)** should not take place.

5.3 Bell

16. Every **eActSetBarrier(CLOSE)** must be preceded by an **eActSetBell(BELL_ON)**, without **eActSetBell (BELL_OFF)** happening in between.

17. In between **eActSetBarrier(CLOSE)** and **eActSetBarrier(OPEN)** the actions **eActSetBell (BELL_OFF)** cannot take place.

18. After **eActSetBarrier(OPEN)** takes place, an **eActSetBell(OFF)** must happen, in a finite number of steps.

5.4 Barriers

19. After **eActSetSignal(RED)** takes place, the **eActSetBarrier(CLOSE)** should take place in a finite number of steps.

20. After a **RESET** action takes place an **eActSetBarrier(OPEN)** must take place in a finite number of steps.

21. After **eActTrainSensData(LEFT,0)** an **eActSetBarrier(OPEN)** must happen in a finite number of steps.

5.5 Deadlock

22. The system must be deadlock free.

6 Architecture

ARCS is designed to have three parallel components namely ARCS-TRAIN, ARCS-BARRIER and the ARCS-SIGNALS.

The ARCS TRAIN handles the data from train monitoring sensors and updates the other two components when trains arrive or leave the intersection. The component also sends a notification when the sensor breaks. The data from the train speed sensors are also handled by the ARCS TRAIN component and this data is used to determine the time information needed for the co-ordination of the signals and barriers.

The ARCS BARRIER component is receives the notifications form ARCS TRAIN and ARCS SIGNALS, using the data to control the opening and closing of the barriers.

The ARCS SIGNALS component controls the traffic lights, warning lights and the alarm based on the notifications from ARCS TRAIN and ARCS BARRIER.

7 List of internal actions

The following table shows all the internal actions together with a short description and the both the source and destination component:

No.	**Name**	**Description**	**Source**	**Dest**
I1	iActTrainsLeft	Notifies that the trains have left and the barriers can be opened	Train	Barrier
I2	iActResetBarrier	Notifies the system to reset to the conceptual initial state	Train	Barrier
I3	iActBarrierClose	Notifies that the barriers are closed	Barrier	Train
I4	iActTrainArrive	Notifies if a train has arrived	Train	Signals
I5	iActSendBreakSignals	Notifies that a sensor break has occurred	Train	Signals
I6	iActSignGreen	Notifies that the signals are green	Signals	Train
I7	iActBarrierOpen	Notifies that the trains have left and the barriers can be opened	Barrier	Signals
I8	iActTimeBarrier	Notifies when the time expires, the signals are red and the barriers can be closed	Signals	Barrier

Table 2: List of external actions

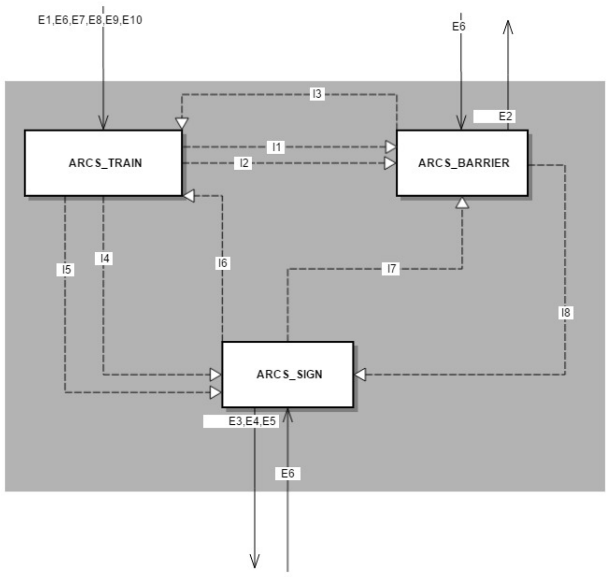


Figure 2: An overview of the internal architecture and communications

Note that the numbers in Figure 2 refer to the internal and external actions in tables 2 and 3

8 Requirements expressed in modal formulas

This section contains a list of the initial requirements translated into modal formulas:

8.1 Sensors

1. (a) $[\overline{eActTrainSensData(BREAK,-1)}^{*}]\langle eActTrainSensData(BREAK,-1)\rangle true$
   (b) $[true*.RESET\_COMPLETE.\overline{eActTrainSensData(BREAK,-1)}^{*}]$
   $\langle eActTrainSensData(BREAK,-1)\rangle true$
   (c) $[true*.eActTrainSensData(BREAK,-1).\overline{RESET\_COMPLETE}^{*}.$
   $eActTrainSensData(BREAK,-1)]false$
2. (a) $[true*.eActTrainSensData(BREAK,-1)]\mu X([\overline{eActSetSignal(RED)}]X\wedge$
   $\langle true\rangle true)$
   (b) $[true*.eActTrainSensData(BREAK,-1)]\mu X([\overline{eActSetWarnLight(W\_HIGH)}]X\wedge$
   $\langle true\rangle true)$
   (c) $[true*.eActTrainSensData(BREAK,-1)]\mu X([\overline{eActSetBell(BELL\_ON)}]X\wedge$
   $\langle true\rangle true)$
   (d) $[true*.eActTrainSensData(BREAK,-1)]\mu X([\overline{eActSetBarrier(CLOSE)}]X\wedge$
   $\langle true\rangle true)$
3. $[true*.eActTrainSensData(BREAK,-1).\overline{RESET}^{*}.eActSetBarrier(OPEN)]false$
4. $[true*.eActSetSignal(GREEN).\overline{eActSetSignal(YELLOW)\cup eActSetSignal(RED)}^{*}.$
   $eActTrainSensData(ARRIVE,1)]\mu X.([\forall_{s:Speed}.\overline{eActGetSpeed(s)\cap}$
   $\overline{eActTrainSensData(BREAK,-1)}\cap\overline{eActTrainSensData(ARRIVE,2)}]X\wedge$
   $\langle true\rangle true)$
5. $[eActTrainSensData(ARRIVE,1)]\mu X.([\forall_{s:Speed}.\overline{eActGetSpeed(s)}\cap$
   $\overline{eActTrainSensData(BREAK,-1)}\cap\overline{eActTrainSensData(ARRIVE,2)}]X\wedge$
   $\langle true\rangle true)$

8.2 Signals

6. (a) $[true*.eActTrainSensData(ARRIVE,1)]\mu X.([\overline{eActSetSigna(YELLOW)}]X\wedge$
   $\langle true\rangle true)$
   (b) $[true*.eActTrainSensData(ARRIVE,1)]\mu X.([\overline{eActSetWarnLight(W\_LOW)}]X\wedge$
   $\langle true\rangle true)$
7. $[\overline{eActSetSignal(YELLOW)}^{*}.eActTrainSensData(ARRIVE,2)]$
   $\mu X.([\overline{eActSetSignal(YELLOW)}]X\wedge\langle true\rangle true)$
8. $[\overline{eActSetWarnLight(W\_LOW)}^{*}.eActTrainSensData(ARRIVE,2)]$
   $\mu X.([\overline{eActSetWarnLight(W\_LOW)}]X\wedge\langle true\rangle true)$
9. (a) $[true*.eActTime(time1)]\mu X.([\overline{eActSetSignal(RED)}]X\wedge\langle true\rangle true)$
   (b) $[true*.eActTime(time1)]\mu X.([\overline{eActSetWarnLight(W\_HIGH)}]X\wedge$
   $\langle true\rangle true)$

10. (a) $[\overline{eActSetSignal(RED)}^*.eActSetBarrier(CLOSE)]false$
    (b) $[true*.eActSetSignal(GREEN).\overline{eActSetSignal(RED)}^*.$
    $eActSetBarrier(CLOSE)]false$
    (c) $[true*.eActSetSignal(YELLOW).\overline{eActSetSignal(RED)}^*.$
    $eActSetBarrier(CLOSE)]false$
    (d) $[\overline{eActSetWarnLight(W\_HIGH)}^*.eActSetBarrier(CLOSE)]false$
    (e) $[true*.eActSetWarnLight(W\_OFF).\overline{eActSetWarnLight(W\_HIGH)}^*.$
    $eActSetBarrier(CLOSE)]false$
    (f) $[true*.eActSetWarnLight(W\_LOW).\overline{eActSetWarnLight(W\_HIGH)}^*.$
    $eActSetBarrier(CLOSE)]false$
11. (a) $[true*.eActSetBarrier(CLOSE).\overline{eActSetBarrier(OPEN)}^*.$
    $eActSetSignal(GREEN)]false$
    (b) $[true*.eActSetBarrier(CLOSE).\overline{eActSetBarrier(OPEN)}^*.$
    $eActSetSignal(YELLOW)]false$
    (c) $[true*.eActSetBarrier(CLOSE).\overline{eActSetBarrier(OPEN)}^*.$
    $eActSetWarnLight(W\_OFF)]false$
    (d) $[true*.eActSetBarrier(CLOSE).\overline{eActSetBarrier(OPEN)}^*.$
    $eActSetWarnLight(W\_LOW)]false$
12. (a) $[true*.RESET]\mu X.([\overline{eActSetSignal(GREEN)}]X \wedge \langle true\rangle true)$
    (b) $[true*.RESET]\mu X.([\overline{eActSetWarnLight(W\_OFF)}]X \wedge \langle true\rangle true)$
13. (a) $[true*.ActTime(time3)]\mu X.([\overline{eActSetSignal(GREEN)}]X \wedge \langle true\rangle true)$
    (b) $[true*.eActTime(time3)]\mu X.([\overline{eActSetWarnLight(W\_OFF)}]X \wedge \langle true\rangle true)$
14. (a) $[true*.eActSetSignal(RED).\overline{eActSetSignal(GREEN)}^*.$
    $eActSetSignal(YELLOW)]false$
    (b) $[true*.eActSetWarnLight(W\_HIGH).\overline{eActSetWarnLight(W\_OFF)}^*.$
    $eActSetWarnLight(W\_LOW)]false$
15. (a) $[true*.eActSetSignal(YELLOW).\overline{eActSetSignal(RED)}^*.$
    $eActSetSignal(GREEN)]false$
    (b) $[true*.eActSetWarnLight(W\_LOW).\overline{eActSetWarnLight(W\_HIGH)}^*.$
    $eActSetWarnLight(W\_OFF)]false$

8.3 Bell

16. (a) $[\overline{eActSetBell(BELL\_ON)}^*.eActSetBarrier(CLOSE)]false$
    (b) $[true*.eActSetBell(BELL\_OFF).\overline{eActSetBell(BELL\_ON)}^*.$
    $eActSetBarrier(CLOSE)]false$
17. $[true*.eActSetBarrier(CLOSE).\overline{eActSetBarrier(OPEN)}^*.$
    $eActSetBell(BELL\_OFF)]false$
18. $[true*.eActSetBarrier(OPEN)]\mu X.([\overline{eActSetBell(BELL\_OFF)}]X \wedge \langle true\rangle true)$

8.4 Barriers

19. $[true*.eActSetSignal(RED)]\mu X.([\overline{eActSetBarrier(CLOSE)}]X \wedge \langle true\rangle true)$
20. $[true*.RESET]\mu X.([\overline{eActSetBarrier(OPEN)}]X \wedge \langle true\rangle true)$
21. $[true*.eActTrainSensData(LEFT, 0)]\mu X.([\overline{eActSetBarrier(OPEN)}]X \wedge \langle true\rangle true)$

8.5 Deadlock

22. $[true*]\langle true\rangle true$

9 Verification

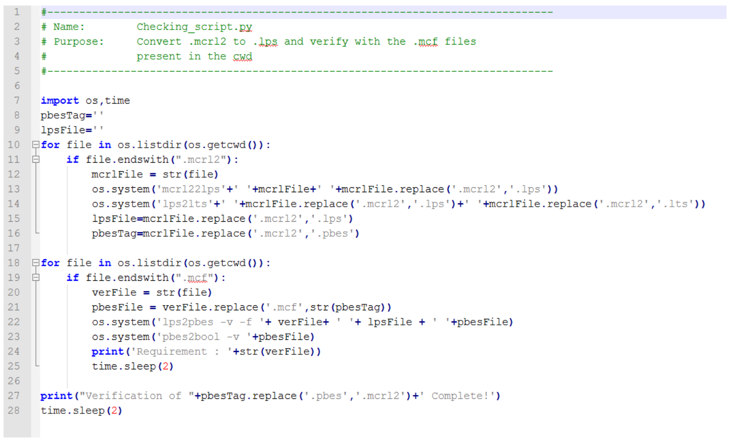
The requirements translated into modal formulas are verified using the following setup:

Operating system	Windows 8.1 64-bit (6.3, Build 9600)
Processor type	Intel Core i7-4700HQ CPU
Processor speed	2.40GHz
Memory	8GB RAM
mCRL2 version	201210.1

Table 3: Verification setup

1. mCRL2 toolset is installed
2. The model is saved with a .mcrl2 extension.
3. The modal formulas are saved with a .mcf extension.
4. The option 'mcrl22lps' is used to generate a .lps file from the .mcrl2 model.
5. A Parameterized Boolean Equation System is created for every requirement file using the 'lpsto2pbes' command with options '-v -f'.
6. The validity of the .pbes file is checked using 'pbes2bool' command with the option '-v'.
7. The output will be true if the formula holds in the system model.

A Python script was written to automatically verify all the requirements against the .mcrl2 file sequentially.



```python
#-------------------------------------------------------------------------------
# Name:        Checking_script.py
# Purpose:     Convert .mcrl2 to .lps and verify with the .mcf files
#              present in the cwd
#-------------------------------------------------------------------------------

import os,time
pbesTag=''
lpsFile=''
for file in os.listdir(os.getcwd()):
    if file.endswith(".mcrl2"):
        mcrlFile = str(file)
        os.system('mcrl22lps'+' '+mcrlFile+' '+mcrlFile.replace('.mcrl2','.lps'))
        os.system('lps2lts'+' '+mcrlFile.replace('.mcrl2','.lps')+' '+mcrlFile.replace('.mcrl2','.lts'))
        lpsFile=mcrlFile.replace('.mcrl2','.lps')
        pbesTag=mcrlFile.replace('.mcrl2','.pbes')

for file in os.listdir(os.getcwd()):
    if file.endswith(".mcf"):
        verFile = str(file)
        pbesFile = verFile.replace('.mcf',str(pbesTag))
        os.system('lps2pbes -v -f '+ verFile+ ' '+ lpsFile + ' '+pbesFile)
        os.system('pbes2bool -v '+pbesFile)
        print('Requirement : '+str(verFile))
        time.sleep(2)

print("Verification of "+pbesTag.replace('.pbes','.mcrl2')+' Complete!')
time.sleep(2)
```

Figure 3: Checking script

10 Conclusions

All the formulated requirements listed in Section 8 were evaluated as *true* in the verification process using the mcrl2 toolset for a maximum of four trains. It is concluded that the Automatic Railroad Crossing System satisfies all mentioned requirements.

11 References

1. J. F. Groote and M. R. Mousavi Modeling and Analysis of Communicating Systems 2013.

2. mCRL2 201210.1 documentation `http://www.mcrl2.org/release/user_manual/user.html`

12 System Model

DO NOT PRINT THIS PAGE. ADD THE CODE FROM THE WORD FILE