Complex Adaptive Systems Modeling

RESEARCH

Open Access

On the real time modeling of interlocking system of passenger lines of Rawalpindi Cantt train station

Umar Khan, Jamil Ahmad*, Tariq Saeed and Sikandar Hayat Mirza

*Correspondence: jamil.ahmad@rcms.nust.edu.pk
Research Center for Modeling & Simulation (RCMS), National University of Sciences and Technology (NUST), H12, Islamabad 44000, Pakistan

Abstract

Purpose: Recent advancements in technology have enabled railway organizations to shift from manual to computer based automated interlocking systems for increasing their efficiency and profits. Since automated systems are complex and interlocking systems are safety critical systems, these systems should be modeled and verified against safety requirements to weed out any design bugs which might lead to catastrophes during their system life cycles. In this study, we model software based automated interlocking control system of a train station, located at Rawalpindi Cantt (Pakistan).

Methods: We have modeled software based automated interlocking control system using timed automata and verified its correctness using UPPAAL model checking software. Timed automata have successfully been used for the modeling and verification of real-time systems.

Results: We constructed a real-time model of railyard interlocking system by employing a model-checking approach to determine behavior of the model under various conditions. The model checker ascertains the absence of errors in a system by inspecting all the possible states or scenarios of the modeled system. The results show that important properties related to the safety of the designed interlocking system of the railyard management system can be verified using our presented approach. These properties ranged from collision and de-railment avoidance to checking the correct error handling functionality of the timed automata models.

Conclusions: The final modular design can easily adapt to the route upgrades and changes within the station by simple variable adjustments. Based on the laid down methodology and verification techniques, this study can be further built upon, extended and linked to cover the shunting aspect of the train station operations, run through operations, introducing automatic train stop (ATS) functionality and recommend three to four aspect traffic signaling for the train station. This study takes a first step in providing an indigenous solution to an indigenous problem of designing an upgraded and verified signaling infrastructure for Pakistan Railway's Rawalpindi Cantt train station.

Keywords: Railway, Interlocking, CTL, UPPAAL, Real Time, Verification, Timed automata

Background

This paper is an extension of work originally presented in Proceedings of the Frontiers of Information Technology Conference 2015 (Khan et al. 2015). This extension includes route cancellation requests and addition of more verification properties for the designed system.

Railways around the world have either introduced or they are in the process of introducing automation in their day to day operations by utilizing the state of the art commercial off the shelf (COTS) information and technology (IT) solutions. Automation of processes is a major contributer to efficiency and reduction in operating and maintenance costs. However, in the interest of avoiding faults during design and development of safety critical IT applications such as interlocking systems, it is always recommended to carry out modeling and verification of the target system (Brown 2000). It is considered an enormous challenge to weed out faults in a safety critical real time system. The methods of *peer review*, *dynamic testing and simulation* (Baier and Katoen 2008) of software cannot be relied upon to completely remove all the errors which in these cases might lead to accidents causing loss of man and material. The present study attempts to model and verify interlocking system of passenger lines of Rawalpindi Cantt Train Yard using a formal method modeling technique, which will be a first step in charting out an indigenous solution to the problem of automating yard operations for Rawalpindi Cantt train station.

The recent major work in automata modeling of rail interlocking systems has been done in Turkish National Railway Signalization Project (Söylemez et al. 2011; Turk et al. 2011; Dincel and Kurtulan 2012) which explain in detail techniques in how to model rail interlocking system modules, simultaneously pointing out the pitfall of state explosion for large models. They have provided a case study for employing their modeling methods to a small rail yard. Ferrari et al. (2011) discuss the interlocking systems modeled and verified in NuSMV (Cimatti et al. 2002) and SPIN (Holzmann 2004) model checking tools. Our study is unique in the fact that modeling and verification of mid to large size railway yard is being undertaken in UPPAAL model checker (Larsen et al. 1997) using timed automata.

One of the major influences of this study is a paper by Szpyrka (2008) in which he gives the methodology to extract relevant data out of a train station by making relationship tables from its interlocking tables. Kanso et al. (2009), Fokkink et al. (1998) and Fokkink (1996) present a detailed verification strategy for the railway interlocking systems.

Railway signaling

Railway lines are divided into sections, called blocks (Satish and Agarwal 2007), each of which is guarded by a traffic signal at its entrance. Hence, a block begins and ends with signals. To enter a block by a train, a permissive signal must be indicated by the traffic signal. Railway traffic signals work in conjunction with point machines to organize safe passage of trains through the tracks. Following are the main elements of interest in a train station.

Track circuit

A track circuit is a sensor, which is a simple electrical logic circuit which detects the presence of a train on a track. AC or DC current is present in the two parallel rail lines and whenever a train moves over the tracks, the circuit is closed and the presence of train is detected.

Signal

Signals authorize the movement of trains. The main signals involved in Rawalpindi Cantt train station are of the two aspect semaphore type (stop and go).

Points

Unlike cars, trains cannot change their direction independently. They require mechanical devices called points to provide passage from one track to another.

Traffic control center

Train movements are controlled and monitored from this facility. At present a simple display shows the position of trains in the yard using track circuits. With the help of electrical slotting, the final adjustment of light signals and switches are carried out by the cabin operators.

Control cabins

These cabins execute switches and signals through mechanical levers which are connected to these elements via steel wires. The cabin operators drive these elements after consulting the interlocking tables. These cabins are the *middle men* which the proposed software based automated interlocking system design aims to remove.

Rawalpindi Cantt train yard

The Rawalpindi train yard has ten platforms in total. Five platforms are reserved for passengers while the rest are for goods. Due to the decline in railway operations, only the passenger platforms have remained operational. Furthermore, only the passenger platforms and their lines have a track detection circuit installed. This station is equipped with warner, home, starter, advance starter, shunting and outer signals (Satish and Agarwal 2007).

Rawalpindi Cantt station has a distributed control for its rail management system, with four lever frame cabins, three of which are interlock enabled. This distributed control system employs mechanical lever frames with slides as an interlocking system, providing fixed block interlock signaling services within the train yard. The traffic signals are of two aspect semaphore type and the points are mechanically engaged. The layout of Rawalpindi Cantt train yard with only the five passenger lines is shown in Fig. 1.

For the conducting safe movement (excluding shunting) for five passenger lines of the yard, there are 25 semaphore signals and 27 mechanically operated points which are controlled via three lever frame cabins.

Interlocking system

Interlocking system (IS) is the personification of safety regulations which governs the safe operation of any system, which in our case is safe movement of trains through a train yard. Its job is to filter out the safe from unsafe inputs, which might cause harm to the system or its users.

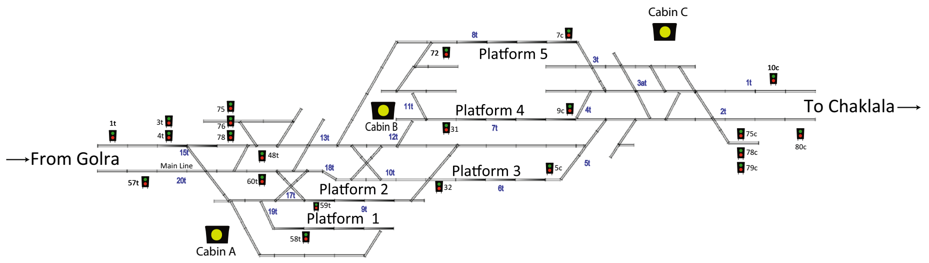


Fig. 1 Layout of passenger Rawalpindi Cantt train yard. Five passenger lines of the Rawalpindi Cantt train yard are shown with the position of signals, their alphanumeric designations and the position of lever frame cabins. Layout is not per actual scale

Rail management system architecture

Interlocking is a safety layer which lies between the input receiving layer (interface layer) and the output layer (infrastructure layer), making sure only the safe inputs are sifted through to become outputs, rejecting those instructions which might cause collision or derailing. Figure 2 accurately shows the position of the interlocking layer within the overall rail management system (Fokkink et al. 1998; Moler et al. 2012; James et al. 2014). Interface layer is the one from which the operator assigns tasks to the program e.g. select and assign a route to a train. It maybe called the user interface (UI) of the (software) system. The interlocking layer checks the inputs from the interface layer i.e. if they are correct in terms of not violating safety which might result in collision or derailing. After checking the input, relevant orders to the field elements are given to the infrastructure layer, which embodies the field elements of the yard. The feedback of the states from the infrastructure layer is relayed to the interface layer via the interlocking layer. This study focuses on the interlocking layer of the automated control and management system of a rail yard.

Design of safety critical systems

EN 50128 (Boulanger 2015), is a European derivative standard of IEC 61508 (Brown 2000) which focuses on safety management techniques specifically for design and development of railway applications. The umbrella standard IEC 61508 (Brown 2000) holistically recommends different methods for modeling safety related systems. Formal

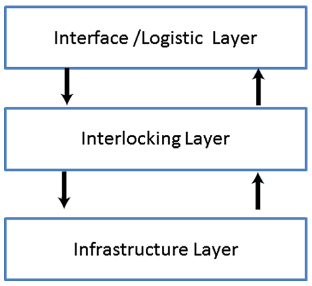


Fig. 2 System architecture. This diagram indicates the position of interlocking system in a rail yard management system hierarchy

methods are highly recommended for modeling safety critical systems such as railway interlocking systems which fall under SIL 4 category (Brown 2000; Vu et al. 2014).

Organization of the paper

The paper starts with the background knowledge of railway interlocking system. Then, the methodology used in this study is described in "Methods" section. The design is proposed in "Design of interlocking system" section. The results acquired by using the modeling and verification approaches for the interlocking system are illustrated in "Results and discussion" section along with the discussions based on the safety aspects of the designed system. Finally, fifth section concludes this paper.

Methods

Timed automata

"An automata is a machine which evolves from one state to another under the action of transitions" (Bérard et al. 2013).

Definition 1 (*Timed Automata*) A timed automaton (TA) is a structure (Olderog and Dierks 2008), $\langle L,B,X,I,E,l_{ini}\rangle$, where

- L is a finite set of locations
- B is a finite set of channels with elements *a*,*b* or any other name like *input* as in Fig. 3. For each channel *a* there are two actions: *a*? denotes an input and *a*! is the corresponding output on the channel a.
- X is a finite set of clocks. A clock is a continuously evolving variable with a rate of 1.
- I: $L \rightarrow \oint(X)$ is a mapping that assigns to each location a clock constraint, its invariant
- $E \subseteq L \times B_{?!} \times \oint(X) \times P(X) \times L$ is the set of directed edges. An element $(l, \alpha, \varphi, Y, l') \varepsilon E$ describes an edge from location l to location l' labeled with the action α, the guard φ, and the set Y of clocks that will be reset.
- $l_{ini} \in$ initial location of L

Figure 3 shows a simple watchdog timer TA (Olderog and Dierks 2008), where
$L = \{start, error\}$
$B = \{input\}$
$l_{ini} = start$

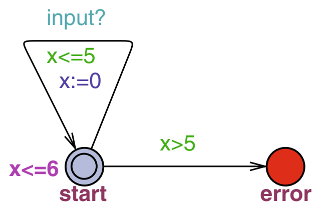


Fig. 3 Watchdog timed automaton where *start* and *error* are the two states, *input* is the reception channel and *x* is the clock variable

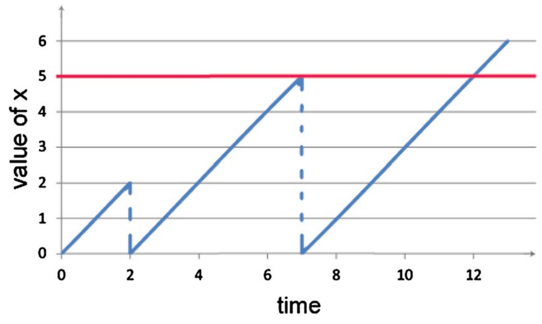


Fig. 4 Clock values

A watch dog timer is a timer which is employed as a forward error correction mechanism to detect and recover from system malfunctions. This simple example illustrates a timed automaton with two states i.e. "start" and "error". The timed automaton is initially in the "start" state, and has the invariant value of ≤6 time units. The system waits to receive input for five time units. If input is received within this time, clock is reset and the timed automaton remains in this location. If, however, an input is not received within five time units, then the timed automaton moves to the next ("error") location, exiting from a potentially never ending wait. Figure 4 shows values of clock for this watchdog timed automaton.

Parallel composition

More than one automata representing subsystems can be combined together in a parallel composition to form a larger system.

Definition 2 The parallel composition (Olderog and Dierks 2008) TAi($TA_1 \parallel TA_2$) of two timed automata TA_1 and TA_2 is expressed as

$$TAi = (L_i, B_i, X_i, I_i, E_i, l_{ini,i})$$

i = 1, 2, with disjoint sets of clocks X_1 and X_2 yields the timed automaton

$$TA_1 \parallel TA_2 = (L_1 \times L_2, B_1 \cup B_2, X_1 \cup X_2, I, E, (l_{ini,1}, l_{ini,2}))$$

Conjunction of location invariants: $I(l_1, l_2) \Leftrightarrow I_1(l_1) \wedge I_2(l_2)$.

The transition relation E is constructed by the following rules:

Handshake communication: synchronizing a! with a? yields τ (internal action), i.e. if $(l_1, \alpha, \varphi_1, Y_1, l'_1)\ \varepsilon\ E_1$ and $(l_2, \widetilde{\alpha}, \varphi_2, Y_2, l'_2)\ \varepsilon\ E_2$

$\{\alpha, \widetilde{\alpha}\} = \{a!, a?\}$ then also

$((l_1, l_2), \tau, \varphi_1 \wedge \varphi_2, Y_1 \cup Y_2, (l'_1, l'_2))\ \varepsilon\ E$.

Asynchrony: if $(l_1, \alpha, \varphi_1, Y_1, l'_1)\ \varepsilon\ E_1$ then for all $l_2\ \varepsilon\ L_2$ also

$((l_1, l_2), \alpha, \varphi_1, Y_1, (l'_1, l_2))\ \varepsilon\ E$

and inversely, if $(l_2, \alpha, \varphi_2, Y_2, l'_2)\ \varepsilon\ E_2$ then for all $l_1\ \varepsilon\ L_1$ also

$((l_1, l_2), \alpha, \varphi_2, Y_2, (l_1, l'_2))\ \varepsilon\ E$

Model checking

Model checking is an automated formal verification technique to ascertain the absence of errors in a suitable system through systemic inspection of all the states of the model (Baier and Katoen 2008). The absence of errors is checked via *computational tree logic* (CTL) (Clarke and Emerson 1982) or *linear temporal logic* (LTL) (Pnueli 1977) statements. CTL is a branching time logic where as LTL, as the name signifies has a linear time perspective.

The branching perspective of CTL is more suitable for verifying correctness of a safety critical system because all possible states in all possible computational paths are ascertained for the absence of a safety negating state. A CTL statement or formula expresses the properties and perspective behavior of the model. The CTL statements are formed by using temporal and logical operators.

Temporal operators

The temporal operators (Bérard et al. 2013) of CTL are

A: In **A**ll possible computational paths, a property will always be satisfied.
E: There **E**xists a path where a property will always be satisfied.
F: The exits a state in the unspecified **F**uture where a property will be satisfied.
G: A property will **G**lobally be satisfied in the future of a path.

Logical operators

The mostly commonly used logical operators are

$\neg$: Negation or the not operator
$\wedge$: Logical And operator
$\vee$: Logical Or
$\Rightarrow$: Implication
$\Leftrightarrow$: Equivalence/double implication

Real time systems

A real time system is a reactive system, which responds to an input within a definite amount of time (Olderog and Dierks 2008). A real time system can be a safety critical system, where the real time constraints are extracted from the safety requirements of that system. A real time system has the following properties:

Definition 3 (*Safety property*) This property states that a bad event should never occur.

In CTL it expressed by the temporal combinators as (Bérard et al. 2013)

$$\mathrm{AG}\,\neg\varphi$$

where φ denotes a bad event.

Definition 4 (*Liveness property*) Liveness property states that something good will eventually occur.

This property is expressed (Bérard et al. 2013) in CTL form as

$$\mathrm{AG}(\psi \Rightarrow \mathrm{AF}\varphi)$$

where φ is an event which eventually occurs after ψ.

Definition 5 (*Bounded response property*) Bounded response property states that a desired response due to an input will occur within a defined time interval.

In timed CTL it can be expressed as

$$\mathrm{AG}(\psi \Rightarrow \mathrm{AF}_{<5s}\varphi)$$

where <5 s denotes that when ψ occurs, φ should be observed in less that 5 s.

Definition 6 (*Duration property*) This property states that the amount of time a real time system will be in its critical condition, that time should be a bounded interval.

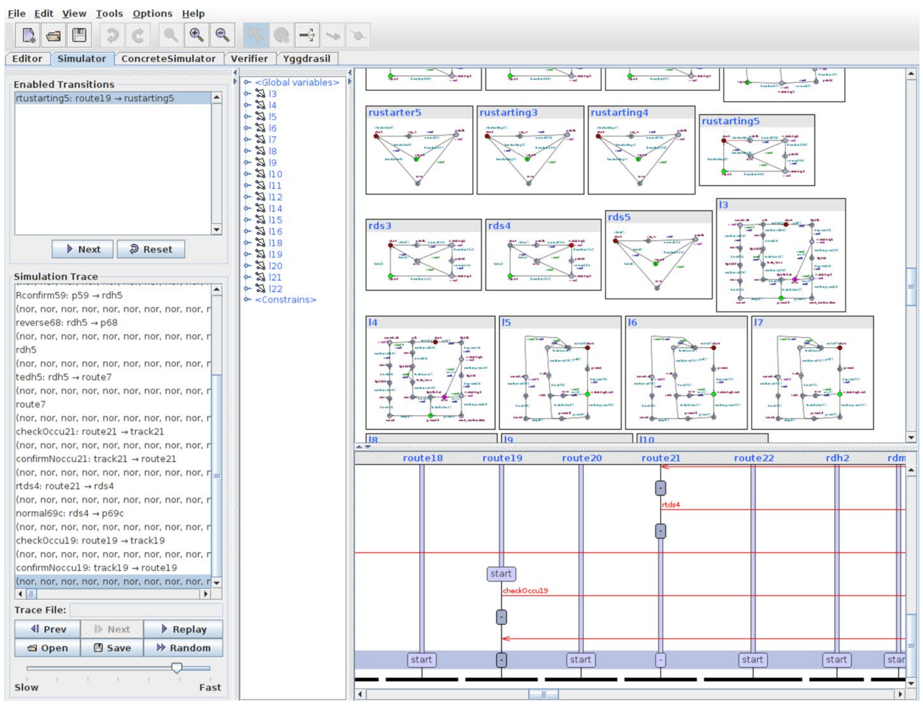
In timed CTL it can be expressed as

$$\mathrm{AG}_{\leq 5s}\varphi$$

where φ is the critical condition for at least five 5 s.

UPPAAL

UPPAAL (Larsen et al. 1997) is a multi-platform timed automata model checker which is used for modeling, simulating and verifying real time systems. Apart from a graphical user interface (Fig. 5) for performing simulation and verification of the model, it also has



```
rtustarting5: route19 → rustarting5
```

```
Rconfirm59: p59 → rdh5
(nor, nor, nor, nor, nor, nor, nor, nor, nor, r
reverse68: rdh5 → p68
(nor, nor, nor, nor, nor, nor, nor, nor, nor, r
rdh5
(nor, nor, nor, nor, nor, nor, nor, nor, nor, r
tedh5: rdh5 → route7
(nor, nor, nor, nor, nor, nor, nor, nor, nor, r
route7
(nor, nor, nor, nor, nor, nor, nor, nor, nor, r
checkOccu21: route21 → track21
(nor, nor, nor, nor, nor, nor, nor, nor, nor, r
confirmNoccu21: track21 → route21
(nor, nor, nor, nor, nor, nor, nor, nor, nor, r
rtds4: route21 → rds4
(nor, nor, nor, nor, nor, nor, nor, nor, nor, r
normal69c: rds4 → p69c
(nor, nor, nor, nor, nor, nor, nor, nor, nor, r
checkOccu19: route19 → track19
(nor, nor, nor, nor, nor, nor, nor, nor, nor, r
confirmNoccu19: track19 → route19
(nor, nor, nor, nor, nor, nor, nor, nor, nor, r
```

Fig. 5 UPPAAL tool box. This figure shows the UPPAAL toolbox in simulation mode

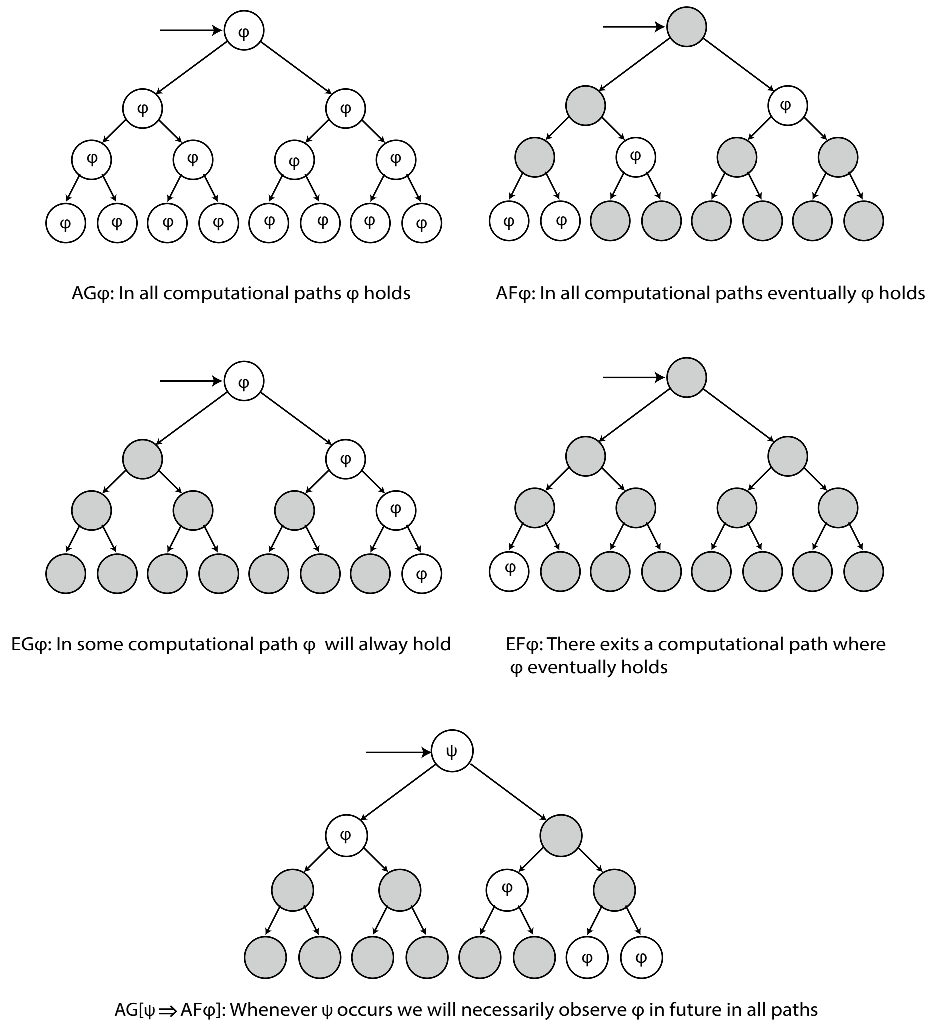

AGφ: In all computational paths φ holds
AFφ: In all computational paths eventually φ holds
EGφ: In some computational path φ will alway hold
EFφ: There exits a computational path where φ eventually holds
AG[ψ ⇒ AFφ]: Whenever ψ occurs we will necessarily observe φ in future in all paths

Fig. 6 Temporal combinators

command line utility, *verifyta*, for performing verification in a Linux or windows terminal with the help of a query file. A query file contains the CTL formulas which are required to be checked by the model checker.

UPPAAL (Larsen et al. 1997) tool, however, has certain limitations in using timed CTL formulas as it can only handle formulas of the type AGψ, AFψ, EGψ, EFψ and AG[$\psi \Rightarrow AF\varphi$] (denoted as $\rightarrow$ in UPPAAL), with no nesting allowed (Behrmann et al. 2004). Figure 6 shows the computational trees of the aforementioned formulas. Despite this constraint, this tool was successfully used for the real time modeling and verification of the interlocking system of Rawalpindi Cantt train yard. UPPAAL (Larsen et al. 1997) uses an extended timed automata structure (Olderog and Dierks 2008), appending more elements in the basic timed automata (TA) structure.

Definition 7 (*UPPAAL extended timed automaton*) An extended timed automaton (Olderog and Dierks 2008) A_e is a structure

$$A_e = \langle L, C, B, U, X, V, I, E, l_{ini} \rangle$$

where L, B, X, I and l_{ini} are the same as the pure timed automaton and

- $C \subseteq L$ is a set of committed locations
- $U \subseteq B$ is a set of urgent channels
- V is a set of data variables, with typical element ν
- $E \subseteq L \times B_{?!} \times \oint(X, V) \times P(X, V) \times L$ is the set of directed edges. An element $(l, \alpha, \varphi, Y, l') \varepsilon E$ describes an edge from location l to location l' labeled with the action α, the guard φ, and the set Y of clocks that will be reset.
- If $(l, \alpha, \varphi, Y, l') \varepsilon E$ and $\text{chan}(\alpha) \subseteq U$ then $\varphi = \text{true}$. This condition prevents that urgent actions are prohibited by guards.

Urgent locations

UPPAAL (Larsen et al. 1997) tool has a feature of urgent locations, which are essentially locations with invariant clock value ≤ 0. When a system is in an urgent location, time cannot pass until the system has left this location. These locations are denoted with the letter "U". Imprudent and excessive use of urgent locations may result in *timelocks* in the system.

Committed locations

Committed locations are same as that of urgent locations, except that when a system is in a committed location, the next transition must be from this location. It is represented by the letter "C". Using this location has an added benefit of not being stored in memory during running of verification of a system (default options). A drawback of using committed locations can be the observance of *timelocks* and/or *deadlocks*, if they are used imprudently.

Urgent channel

An urgent channel is a type of channel declaration offered by UPPAAL (Larsen et al. 1997) tool. When a channel is declared urgent, the transition pertaining to that channel will fire immediately upon being enabled. A drawback of using urgent channels is that the urgent channel guard cannot contain a clock variable.

Modeling scope and assumptions

In order to bring down the complexity of the model as well as to avoid the state explosion problem when analyzing railway systems, we make the following assumptions regarding train movement and signaling operations:

- The train drivers obey the signals at all times.
- Train shunting is not considered.
- This model does not include the subsidiary signals such as shunting and junction indicators.
- This model does not include level crossings.

Table 1 Route dependencies

Routes	Point position		Signal
	Normal	Reverse	
Down home 2	29t, 37t	18t, 36t	3t, 1t (down outer)
Down to main line	18t		4t, 1t (down outer)
Route from down main line to no. 3	44, 59	53, 68	75
Route from down mainline to no. 4	53, 59	40, 68	78
Route from down mainline to no. 5		59, 68	76
Up home 3	37c, 42c, 53c, 69c	39c	80c (up outer), 79c
Up home 4	39c, 42c, 53c, 69c		80c (up outer), 78c
Up home 5	44c, 57c, 67c	33c, 34c, 69c	80c (up outer), 75c
Up start 1	18t	22t	58t, 57t (up adv starter)
Up start 2	18t	22t	59t, 57t (up adv starter)
Up starter from no. 3	37t, 47, 53		32
Up starter from no. 4	37t, 38	40, 57	31
Up starter from no. 5	36		72
Up starting 3 and 4 from main line			60t, 57t
Up starting 5 from to main line	32t	30t	48t, 57t
Down start 3	69c	44c	5c, 10c (down adv starter)
Down start 4	69c	30c	9c, 10c (down adv starter)
Down start 5	69c		7c, 10c (down adv starter)

- This model caters for the five passenger lines of the yard only.
- Two aspect traffic signals are used.
- There are no conflicts of train movements with the other adjacent stations.

Control tables

Control tables are a set of tables which govern the functioning of the interlocking layer. These tables are derived from the interlocking tables of the train yard. Every train yard has its distinct interlocking tables. Table 1 shows what signals and points (with their positions) must be engaged to have a safe route charted out for the incoming train.

Conflicts table

Table 2 is the heart of the interlocking system. It lists all the routes in x and y axes and marks out the routes which are in conflict with other routes, which if left unlocked (left accessible) will result in collision of trains. Eighteen routes have been defined and their conflict with each other is demonstrated in Table 2 , with "/" denoting conflict with a particular route.

Design of interlocking system

Design overview

Interlocking system is the safety ensuring layer of a rail yard management system. Figure 2 shows information flow (Fokkink et al. 1998; Moler et al. 2012; James et al. 2014) in a rail yard management system. It's purpose is to check and restrict processes which are hazard prone.

Table 2 Conflicting routes

Routes	Down home 2	Down to main line	Route from down main line to no. 3	Route from down main line to no. 4	Route from down main line to no. 5	Up home 3	Up home 4	Up home 5	Up start 1	Up start 2	Up starter from no. 3	Up starter from no. 4	Up starter from no. 5	Up starting 3 and 4 from main line	Up start-ing 5 from main line	Down start 3	Down start 4	Down start 5
Down home 2		/							/	/				/	/			
Down to main line	/														/			
Down main line to no. 3				/	/	/					/	/	/		/			
Down main line to no. 4			/		/		/					/	/		/			
Down main line to no. 5			/	/				/					/		/			
Up home 3			/				/	/								/		
Up home 4				/		/		/								/	/	
Up home 5					/	/	/									/	/	/
Up start 1	/									/				/	/			
Up start 2	/								/					/	/			
Up starter from no. 3			/									/						

Table 2 continued

Routes	Down home 2	Down to main line	Route from down main line to no. 3	Route from down main line to no. 4	Route from down main line to no. 5	Up home 3	Up home 4	Up home 5	Up start 1	Up start 2	Up starter from no. 3	Up starter from no. 4	Up starter from no. 5	Up starting 3 and 4 from main line	Up starting 5 from main line	Down start 3	Down start 4	Down start 5
Up starter from no. 4			/	/							/							
Up starter from no. 5			/	/	/													
Up starting 3 and 4 from main line	/								/	/					/			
Up starting 5 from main line	/	/	/	/	/				/	/				/				
Down start 3						/	/	/									/	/
Down start 4							/	/								/		/
Down start 5								/								/	/	

The forward slash "/" shows that a conflict exists between routes indicated on x and y axes

Keeping in view of the aforementioned goal, different aspects of the interlocking system are designed in a modular structure using the UPPAAL tool (Larsen et al. 1997). Each module, called a template in UPPAAL tool (Larsen et al. 1997), performs a specific function in the IS. The basic process flow in the IS design is shown in Fig. 7 and described as follows:

- Whenever a route is requested, it is checked whether there are routes, which are in conflict to this route are selected. If so, then the route request is denied.

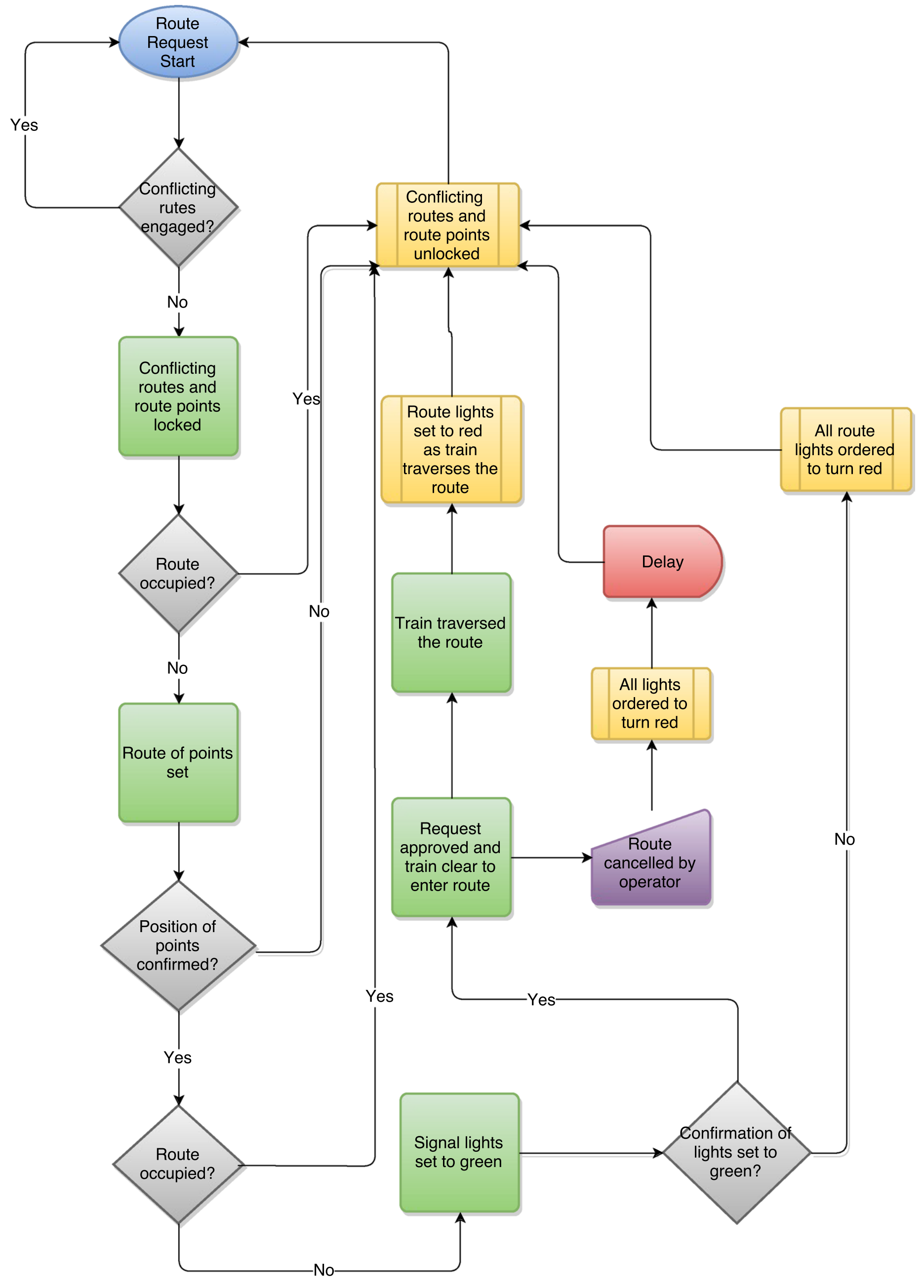

Route Request Start
Yes
Conflicting rutes engaged?
No
Conflicting routes and route points locked
Route occupied?
Yes
No
Route of points set
Position of points confirmed?
No
Yes
Route occupied?
Yes
No
Conflicting routes and route points unlocked
Route lights set to red as train traverses the route
Train traversed the route
Request approved and train clear to enter route
Signal lights set to green
Yes
Confirmation of lights set to green?
No
Route cancelled by operator
All lights ordered to turn red
Delay
All route lights ordered to turn red

Fig. 7 Flowchart. This flow chart describes the sequence of events to be undertaken by an interlocking system

- If no conflicting routes are in operation then it is checked if, the tracks of the selected route are occupied or not. If they are occupied, then the request is denied.
- If, however, the intended tracks are unoccupied, then the points on that route are ordered to move to their desired positions and their feedback is awaited (watch-dog timer). If there is no feedback in a stipulated amount of time, then it is assumed that a problem has occurred with a point and the route request is again denied.
- When the points are in their desired position, they are locked. The term point lock means that the points in a route will not be able to change their positions when a train is traversing over it.
- After all the points are in their desired positions then as a precaution the route tracks are again checked for emptiness. If they are empty, then the requisite signal(s) are ordered to turn green and their feedback is awaited. If, however, no feedback is received within a stipulated amount of time, it is again assumed that a fault has occurred with the signal(s) and the route request is denied.
- When all signals are green, a train is given permission to traverse that route. During its journey, all signals which are being crossed by the train are ordered to turn red.
- A route cancellation by the train dispatcher will only be possible when the route signals are green and the train has not entered the route. When a cancellation command is initiated, the light signals of the route are turned immediately "red" and the routes are unlocked and points access granted only after a set amount of delay.

Safety design

For an interlocking system, safety is of the prime concern.

Definition 8 Safety, in terms of a railway based interlocking system is defined as (Moler et al. 2012)

$$Safety = no\ collision \cap no\ derailment$$

No collision

No collision (Antoni and Ammad 2008), as shown in Fig. 8 is an important property for safe transition within a yard, which states that whenever there is a train en-route, there is no possibility that

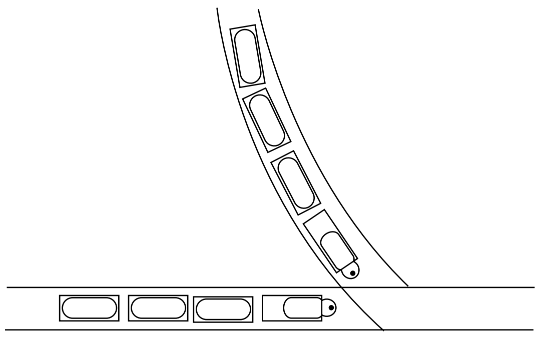


Fig. 8 Collision. Trains headed towards collision on a merging route

- Some other train will collide from rear
- Crash into some other stationary or moving train
- Collide into the main body of some other train which is moving on a merging track
- Crash with another train whose some part of the route is shared by the first train
- Collide head to head with a train coming from opposing direction

No derailment

No derailment means there is no possibility that whenever a train is moving over a point enroute towards its destination, that point suddenly changes its orientation, causing the train to move in two directions at once resulting in derailment. So, safety criteria is that once the points are being put in reverse or normal position, they have to be unoccupied (Fokkink 1996). Figure 9 shows derailing of a train due to a point changing its orientation when a train is crossing over it. Table 3 shows the possibility of derailment in routes where there are shared points which can move in two directions.

Real time constraints

Another important aspect of safety is that, whenever a field actuator is ordered to do a task, it should deliver its feedback in a timely manner (Yildirim et al. 2010). In real world applications there are many possibilities of equipment malfunctions especially in hazard prone places like an open rail yard. Field equipment (called wayside equipment) malfunctions in an open rail yard can range from inherent mean time between failures (MTBF) of the equipment to slicing off of communication cables by a donkey cart which may happen to cross over the railway lines. The interlocking system (IS) can command a wayside equipment to do a job, which may or may not be received and may or may not be acted upon. The design of the IS must cater for this and the best solution to satisfy all these possibilities is the use of a watchdog timer. A wayside equipment may be able to do a job, but its failure to give feedback in a stipulated amount of time will be considered as a fault/error by the interlocking system and appropriate action will be taken in that regard (Yildirim et al. 2010). The stipulated time is defined as the max rated response time of an equipment and the system lag (delay) time. System time lag represents the adjusted inherent loss of time in receiving a response from a field equipment. These *response times* are usually rated in datasheets (Siemens 2015) of the wayside equipment. This feature is implemented in "point driver" and "signal driver" timed automata modules, which are described in the next section.

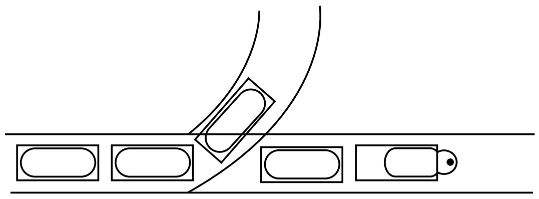


Fig. 9 Derailment. A train is being derailed due to a point changing it orientation during the transit of a train

Table 3 Points positions vs. routes

Routes/points	18t	22t	29t	30t	30c	32t	33c	34c	36t	36	37c	37t	38	39c	40	42c	44c	44	47t	53c	53	57	57c	59	67c	68	69c
Down home no. 2	\		/						\			/															
Down to main line	/																										
Route from down mainline to no. 3																		/			\			/		\	
Route from down main line to no. 4															\						/			/		\	
Route from down main line to no. 5																								\		\	
Up home no. 3											/			\		/				/							/
Up home no. 4														/		/				/							/
Up home no.5							\	\									/						/		/		\
Up start no. 1	/	\																									
Up start no. 2	/	\																									
Up starter from no. 3																			/			/					
Up starter from no. 4													/		\							\					
Up starter from no. 5										/																	
Up starting no. 5 from to main line				\		/																					
Down start no. 3																	\										/
Down start no. 4					\																						/
Down start no. 5																											/

In this table forward slash "/" denotes the normal position of a point in a route and back slash "\" shows the reverse position in a route

Design of elements

The proposed IS design is modular in structure using the UPPAAL tool (Larsen et al. 1997). Each module, called a *template* in UPPAAL toolbox (Larsen et al. 1997), performs a specific function in the IS. Templates are instantiated into processes in the *system declarations* section of UPPAAL tool (Larsen et al. 1997) and eventually, relevant processes are integrated together in a parallel composition (Olderog and Dierks 2008) using *system* command in *system declarations* section to form a complete system. All variables used in the design are global variables except for clock variables and otherwise specifically mentioned local variables. Details of different modules of the design are described below and their holistic relationship with each other via channels is shown in Fig. 10.

Point automaton

Electric point machines are devices with electrical motors. The *response time*, in case of an electric point machine is called the *throwing time*. Throwing times are usually rated in datasheets (Siemens 2015) of the machine.

The requirement of the automaton design for points is that it should be as simple and as small as possible, so as to aid in remedying the state explosion problem. The other

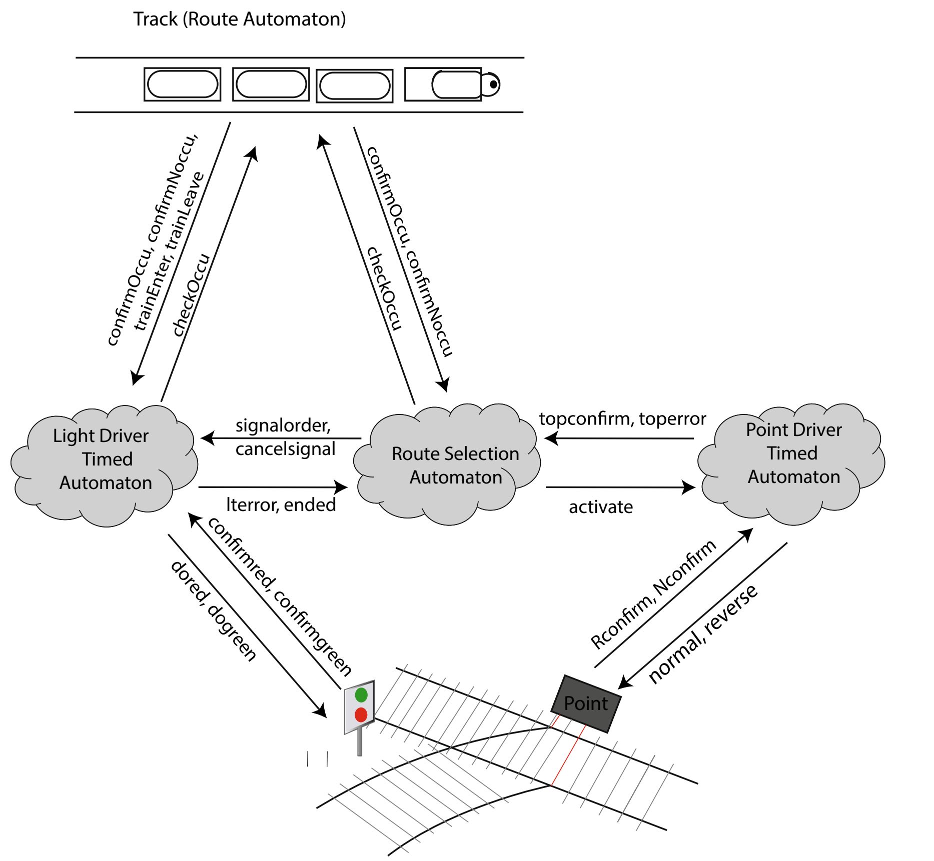

Track (Route Automaton)
confirmOccu, confirmNoccu, trainEnter, trainLeave
checkOccu
checkOccu
confirmOccu, confirmNoccu
Light Driver Timed Automaton
signalorder, cancelsignal
lterror, ended
Route Selection Automaton
topconfirm, toperror
activate
Point Driver Timed Automaton
confirmred, confirmgreen
dored, dogreen
Rconfirm, Nconfirm
normal, reverse
Point

Fig. 10 Automata communication map. The figure shows the communication of automata with each other via channels, which are described over *arrows*. The *directional arrows* originating from automata represent the output channels (!) of that automata, while the directed automata have the input channels (?)

requirement is that it should have an element of error in the design, in that there should be a possibility that the point sometimes is unable to give feedback within a stipulated amount of time (we used 6 s), simulating error. This will help in designing correct IS which caters for faulty equipment. "NormalPos" is the initial state representing normal position, which is one of two positions for a point machine, as shown in Fig. 11. "ReversePos" state stands for reverse position. The reception channels are "reverse" and "normal" while the automaton sends confirmation signals via "Nconfirm"(for normal) and "Rconfirm"(for reverse) channels.

Signals light automaton

Figure 12 shows the signal light automaton. To model railway signal light, it is important to keep in mind the construction of railway signal light to accurately model its mechanism and to avoid the unnecessary addition of states resulting in state explosion. The requirements are similar as that of the point machine. The difference being that error induction is with the green signal aspect only. Due to the fail-safe nature of traffic light equipment, which translates to any scenario where there is doubt/issue with communication with the control center, the light will always display a most restrictive aspect i.e. red (Signals 2013a). To confirm the occurrence of red signal i.e. "confirmred", the "confirmred" channel is defined as an *urgent* channel. No error is assumed in going Red for

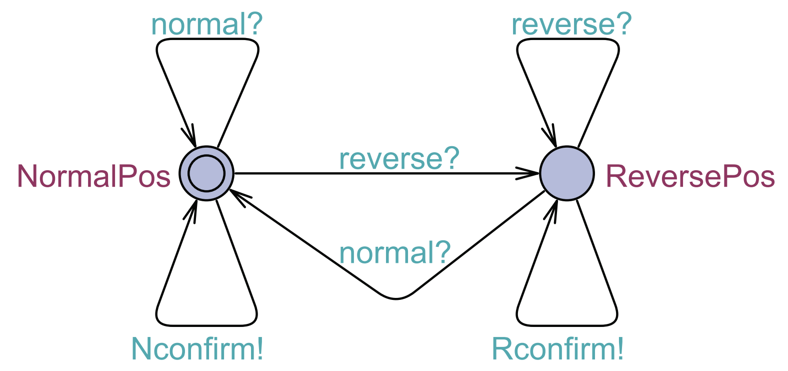


Fig. 11 Point automaton where *NormalPos* and *ReversePos* are the two states and *reverse, Rconfirm, normal* and *Nconfirm* are the communication channels

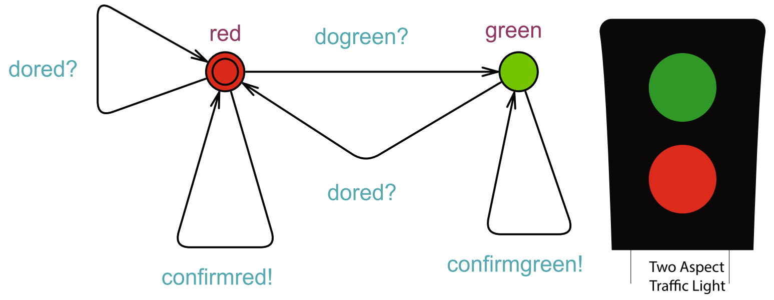


Fig. 12 Light automaton where *red* and *green* are the two states and *dored, confirmred, dogreen* and *confirmgreen* are the communication channels

light signal, however for the Green aspect, a possibility of error is made to be present in the shape of a regular channel "confirmgreen", due to which a transition may or may not be fired as soon as the light automaton is given command to turn green. This presents a possibility of error which the interlocking system is designed to handle. Since the response time of electric lights to change their aspects is almost instantaneous, the timeout used here is 2 s representing system communication lag time

Route automaton

Route automaton, shown in Fig. 13 is a holistic abstraction of the track circuits present in a particular route. Its purpose is to simulate occupation of a route with the entering and leaving of a train in a particular route. If the train enters the beginning part of the route, as was intended, then the first track circuit present in that route detects its presence, sends a message "trainEnter", and moves to "trackOccupied" state. When it leaves the route by passing over the last track circuit in the route, the "trainLeave" message is fired and the automaton comes in "emptyTrack" state.

If, however, track circuits other than the first track circuit of the route detects the presence of a train, a transition is fired which takes the automaton in the "trackOccupied" state with setting the value of a boolean variable "tmp" as 1. As described earlier, this is to randomly simulate the occupation of the route and judge the response of the interlocking design. When this random train leaves the tracks, it takes automaton to its "emptyTrack" state by resetting the "tmp" variable to zero.

The message channels "checkOccu" is used by the route selection automaton to check if the route is occupied or empty, with the message channels "confirmOccu" and "confirmNoccu" giving the desired feedback respectively.

An addition can be be made, with a new place in the route, which symbolizes a train docking at a platform for embarking/disembarking it's load. Only those routes which are arriving at platforms are expressed with this deviation as shown in Fig. 14. Whenever a train is docked, and has not left a route, another train cannot be received in that route, protecting it from collision. The place *docked* is arrived after the trainLeave signal is fired. This ensures that all conflicting routes are released, but any train coming into this track is forbidden due to built-in safety which checks un-occupancy of every route, before it is being assigned to it.

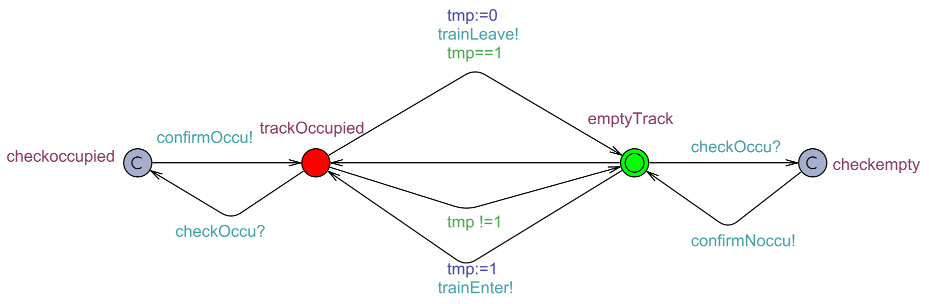


Fig. 13 Route automaton where *trackOccupied, checkoccupied, emptyTrack* and *checkempty* are the locations, *trainEnter, trainLeave, checkOccu, confirmOccu* and *confirmNoccu* are the communication channels and *tmp* is the local boolean variable

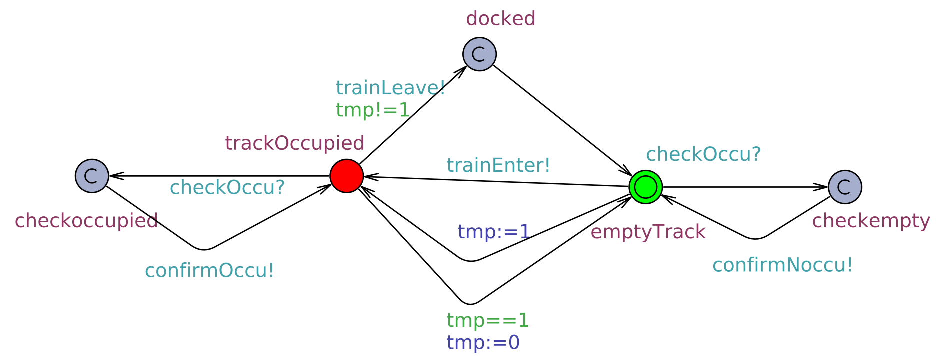


Fig. 14 Route automaton with platform docking where *trackOccupied, checkoccupied, emptyTrack,docked* and *checkempty* are the locations, *trainEnter, trainLeave, checkOccu, confirmOccu* and *confirmNoccu* are the communication channels and *tmp* is the local boolean variable

Committed locations are used to reduce the unnecessary state space. This has the draw back of assuming that the trains will move instantaneously from the docked place.

Route selection automaton

This is the main driver automaton which calls other automata to perform specific functions which give feedback on the actions taken.

This automaton has versions dependent upon the number of conflicting routes involved. The version in Fig. 15 handles three conflicting routes. The number of boolean variables used in this automaton depends upon the total number of conflicting routes with the additional one extra boolean variable.

The additional variable is "TickS", which handles the self assignment of the route, which is put out of use by setting of its flag. The first transition from place "start" to "occupation" has three important properties. The first is the guard, which restricts this transition unless all route boolean variables are unset (having zero value). This guard is represented as

$$\text{Guard: tickS} == 0 \,\&\, \text{tick} == 0 \,\&\, \text{tick2} == 0 \,\&\, \text{tick3} == 0$$

This important feature runs route boolean variables against the logic AND, ensuring that the automaton remains in its initial place until this logic is satisfied, meaning no other conflicting routes are ever engaged which might cause collision or derailment of the train.

When the guard is satisfied, the automaton renders, the route in question and its conflicting routes, out of action by setting (update value to 1) the boolean variables.

$$\text{Update: tickS} = \text{tick} = \text{tick2} = \text{tick3} = 1$$

The third property of the transition is the action message "checkOccu". This message checks (by calling out to *Route automaton*) whether the track segments of the route in question are empty or occupied. If the route is occupied, the message "confirmOccu" is received, taking the automaton to "clearance" state, which ultimately takes the automaton to it's initial state by resetting all the route boolean variables, allowing conflicting routes to be engaged.

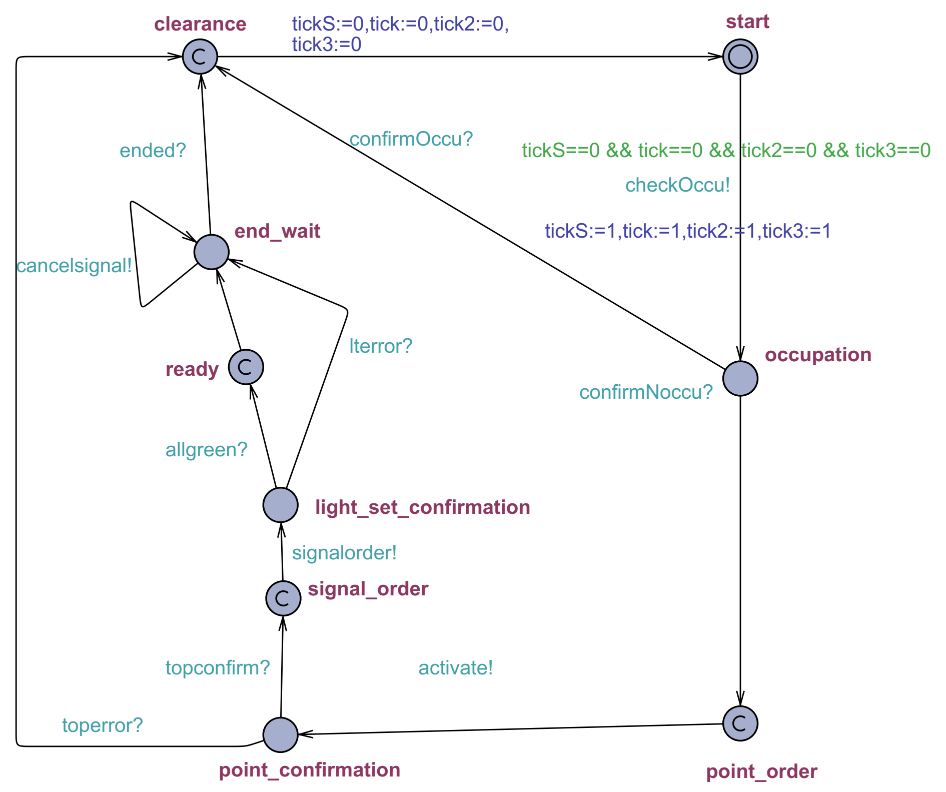


Fig. 15 Route selection automaton with three conflicting routes where *start, occupation, point_order, point_confirmation, signal_order, light_set_confirmation, ready* and *clearance* are the states, *checkOccu, confirmNoccu, confirmOccu, cancelsignal,activate, topconfirm, toperror, allgreen, lterror* and *ended* are the channels and *tickS,tick,tick2* and *tick3* are boolean variables

After having received confirmation via "confirmNoccu", that the intended route is not occupied, the automaton moves to next state "point_order" which directs the required points to assume their positions via "activate" action message. This message calls upon the *point driver timed automaton* into action which passes the message of "topconfirm", in-case of successful operation or "toperror", in-case of failure to do the required task. The failure leads the automaton to the "clearance" state and then eventually to the initial state by releasing all the conflicting routes for operation by resetting the boolean variables. After successful operation of the points, "signal_order" state is reached, which finally orders the traffic signals to assume green color via "signalorder" action message to allow trains to enter the route. "signalorder" message instantiates the *signals driver timed automata*, which responds to *route selection automaton* with the messages of "allgreen" and "lterror" for success and failure, respectively. The failure to change lights to their green colors results in the automaton coming in the "end_wait" state. In this state the automaton is waiting for all the lights to be turned red (some might have been turned green previously in the sequence) via the "ended" synchronized reception channel which leads to the "clearance" state. All the out of play routes are again made available by resetting the boolean route variables and the automaton comes back to its initial state. Upon successful change of lights to green, the automaton comes in the "ready" state, which immediately leads to "end_wait" state, where the automaton is waiting for the train to

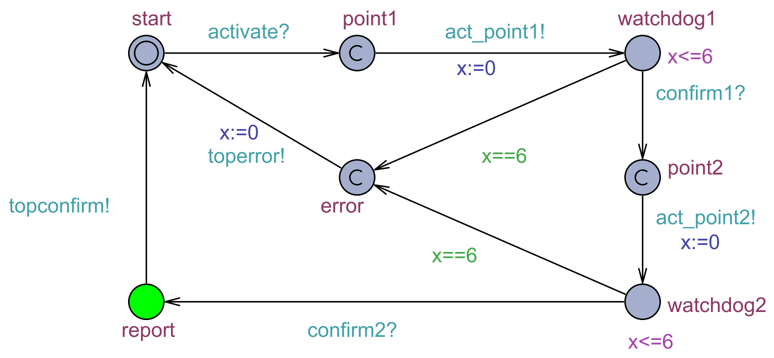


Fig. 16 Point driver timed automaton with two point engagement where *start, point1, point2, error, watchdog1, watchdog2* and *all_done* are the states, *activate, act_point1, confirm1, act_point2, confirm2, toperror* and *topconfirm* are the channels and *x* is the clock variable

pass the route and lights be turned to red via the "ended" synchronized reception channel. The automaton arrives at its initial state by releasing all the previously locked routes.

A utility function of *cancelsignal* is added into the automaton. This is added to give the train dispatcher the flexibility of operations by allowing him to cancel a route, if the intended train has not already entered the green lit route. When a route is canceled, the signal lights of the selected route are immediately turned red and the routes are unlocked after a period of set delay so as to give reaction and stopping time to the train drivers. Though this functionality is instantiated at the *route selection automata* by the triggering of the "cancelsignal" channel, but it actually transpires in the light driver timed automata.

Points driver timed automaton

This timed automaton module, shown in Fig. 16, is activated by *routes selection automaton* via "activate" reception channel, which orders it to set an array of points for a route. As opposed to *routes selection automaton*, it is a timed automaton. When this timed automaton is summoned into action, it gives messages to the designated points to assume a particular position. A point can have two positions, namely "normal" and "reverse".

When *points driver timed automaton* instructs a point to assume a particular position, it expects confirmation within a stipulated amount of time (Signals 2013b). As an example we have taken Siemens point machine, S 700 (Siemens 2015), having a throwing time of 5 s. By adding 1 s of system delay, we are arrived at six time units of delay. This delay is taken only as an example and can be as large as per requirement (Signals 2013c).

As mentioned earlier, to circumvent the possibility of critical errors which can have catastrophic results, a timed response is a necessity. With a timed response from the point, the next point is ordered to take it's designated position and so on. With successful assumption of all points to their desired position, a "topconfirm" message is passed to the *route selection automata*. Without a timed response from any point, it is assumed that something has gone wrong and "toperror" message is generated. Each route has its distinct set of points at particular positions. Hence every timed automaton is different for a particular route. The timed automaton shown in Fig. 16 can linearly expanded or

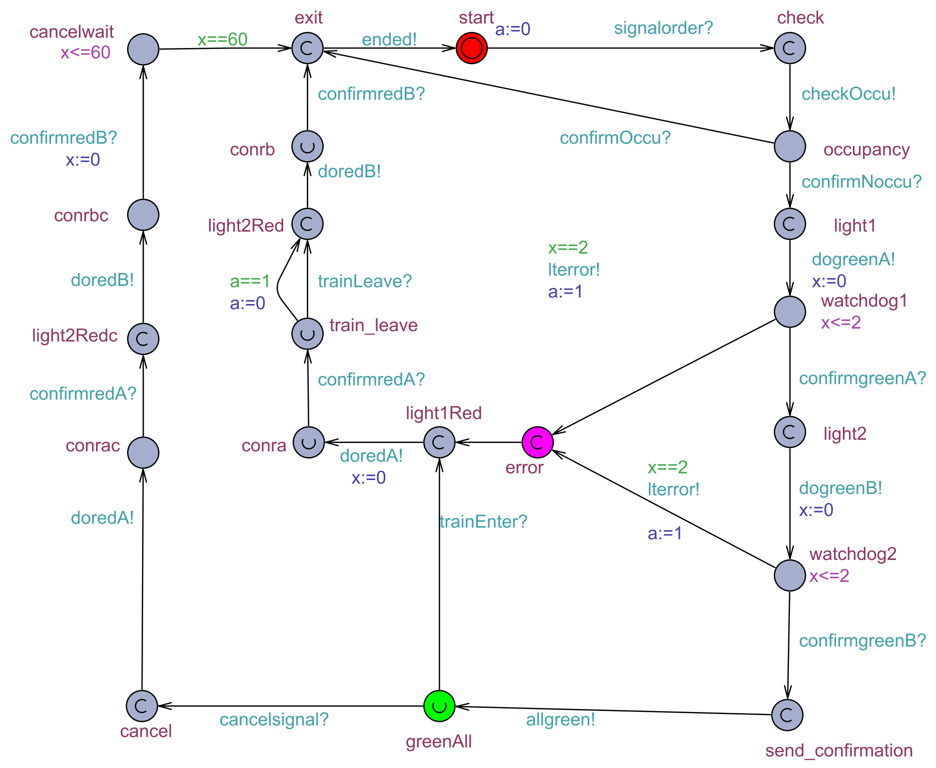


Fig. 17 Light driver timed automaton with controlling two lights. *start, occupancy, light1, light2, light1Red, light2Red, watchdog1, watchdog2,error, send_confirmation, greenall, conra, conrac, conrb, conrbc, cancelwait, cancel, light2red, light2redc, train_leave* and *exit* are the states, *signalorder, cancelsignal, checkOccu, confirmOccu, confirmNoccu, dogreen1, dored1, confirmgreen1, confirmred1, lterror, dogreen2, dored2, confirmgreen2, confirmred2, allgreen, trainEnter, trainLeave* and *ended* are the channels, *a* is a boolean variable and *x* is the clock variable

reduced for more or less than two points by simply adding or reducing locations and transitions for new points.

Signals driver timed automaton

After the successful engagement of points, *the route selection automaton*, finally summons the signals driver timed automaton module, Fig. 17, to display green aspects on the traffic signal lights. As a precaution before engaging the signals, this timed automaton again checks the route for occupancy. It only proceeds when the route is empty, other wise it generates an error and exits.

This timed automaton is complicated, as unlike *points driver timed automaton*, the signals must be turned back to their restrictive aspects (red signal) after they are turned green.

In order to design this timed automaton, the functioning of railway lights must be kept in mind. All field elements since the early era of railroading have always been designed with safety in mind. For example the semaphore signals have fail-safe functionality in that, the signal in the case of cutting of the communication steel wire, will result in automatic display the Red aspect of the signal. Hence, there is no chance of accident, as the system is designed in such a way that the traffic indication system will always fall towards a most restrictive aspect when in error (Signals 2013a). Similarly, the modern electronic lights have retained the same operating principle, in that the lights will only display green signal when there is a signal present from the control station to display this aspect. As and when this signal is lost, the light will fall to its restrictive aspect, i.e. Red.

Like the *point driver timed automaton* whenever a command is given to the light to change color, its feedback within a stipulated amount of time is awaited. In case of lights, we have used 2 s. If feedback is received within this time, the next signal is ordered to turn to green. When all the lights are turned green, a message "allgreen" is passed to the *route selection automaton*, and the train is awaited to traverse the track. If, however, at any point feedback is not received in time, then a message "lterror" is passed to the *route selection automata*, and all the signals are given a command to turn to their red positions as there is an error in the field elements of the route. After this sequence is completed, a message "ended" is passed on to the *route selection automaton* to indicate that all the *light driver timed automaton* sequences have concluded. An additional feature of route cancellation is introduced in this automaton. In a route cancellation scenario, if the train dispatcher, decides to abruptly cancel the route due any or no reason (Söylemez et al. 2011), then all lights signals in that route will be ordered to turn red. However, the points and the routes will be locked for a cushion period (we have used 60 s), to give the trains which may be moving towards or with-in a route a time to adjust to this change. This cushion time can be any amount of time recommended for a particular station or of the operator's choosing with the safety of the station in mind. This timed automaton can be reduced to act as a single light driver by removing the extra locations and transitions of the second light.

Safety properties

Safety properties relevant to our IS are expressed as:

Collision avoidance property

1. At all times when a *route selection automaton* is not in its initial state, then all other conflicting *route selection automata* are in their initial state. In CTL form this property can be expressed as

$$p1 = AG[\neg a.start \Rightarrow (b.start \wedge c.start \wedge d.start)] \tag{1}$$

where *b, c* and *d* are conflicting routes to *a* and "start" is the initial location of the automata. This property outlines collision freedom in the sense that whenever a route is engaged, i.e. a route selection automaton (e.g. *a*) is not in its initial state, then there is no chance that a train on conflicting routes (i.e. *b, c* and *d*) will be operating, as those conflicting routes will never be set, marked by the fact that all conflicting route selection automata are stuck in their initial state for the entire duration.

No derailment property

2. Until and unless a train has left a route or arrived at the destination platform, the points falling into that route at all times are at their required position (*r* and *s*).

$$p2 = AG\neg p.start \Rightarrow r \wedge s \tag{2}$$

where *p.start* signifies the start state of a *light driver timed automata* and *r and s* denote the point orientation with in that route.

The property expressed in formula 2 exists to verify that if a train is given green signal to enter a route, the underlying point(s) present in that route will not change their orientation until the train movement over that route is complete and the routes are unlocked. If the points change their position during this movement, it will result in the train going in two directions at once, causing derailment. The CTL statement utilizes the *light driver timed automata* of the routes to signify movement of a train on the tracks. When the timed automata is not in its start(initial) state, it is running the lights sequencing turning them "green" and after the trains have left, the sequence turns them "red". With using logical implication against a particular state of the point machine, it is checked that the mentioned point state is the only state which is observed through out. If there is any other state of the point than the one mentioned, then the property is negated. This signifies the aspect of the interlocking model that when signals are being engaged for a route, the points will not change their positions. When a light timed automaton for a route is in its initial state, only then the points can change their positions. Therefore, smart *point locking* is achieved, which does not allow a route to be enabled which commands a *change* of position of a point to occur when the same point is used by another route.

Assured timer expiration properties

3. Error state p in timed automata is always reached if the time out x is greater than or equal to the stipulated time. Hence the query statement takes the shape

$$p3 = AG[p \Rightarrow x \geq time] \tag{3}$$

Formula 3 checks the correct working of the timed automata. Formula 3 checks that whenever the timeout is reached or exceeded, the error state is achieved.

4. It is not possible that error state p in timed automata is reached if the time out x is less than the stipulated time. The timed CTL statement has the shape

$$p4 = EF\neg[p \wedge x < time] \tag{4}$$

Formula 4 does its job by confirming that the error state is never falsely achieved when the timeout has not been approached. Since, we are using *committed* and *urgent* locations in copious amounts, whose effect on the automata is to stop time, these two properties are vital as they check if our watchdog timers in automata are functioning correctly and the timed automata is advancing correctly.

Route cancel request safety properties

5. When a route is canceled ψ (e.g. l3.cancel) , the routes are only unlocked φ after the required *Delay* has passed.

$$p5 = AG[\psi \Rightarrow AF_{\geq Delay} \quad \varphi] \tag{5}$$

Formula 5 describes the route cancel-signal safety property. *cancelsignal* is a utility introduced to facilitate the train dispatcher to safely cancel a route when the route signals are green and the target train has not entered the route. *cancelsignal* immediately turns the signals to red but only releases the points and routes when a certain delay has elapsed.

Failsafe functionality

6. If there is an error (ψ) in setting of points, then that route will not be cleared for trains i.e. given green signal (φ).

$$p6 = AG[\psi \Rightarrow AF\neg\varphi] \tag{6}$$

Formula 6 is a safety property for points in that it states that whenever an error occurs in moving points to their desired orientation, the traffic lights for that route will never be turned green. "The lights will *never* be turned green" is accomplished by not arriving at a location "signal_order" of *route selection automata* which instantiates the *light driver timed automata* for the routes. If this location is never reached, then lights will never be turned green.

Model checking strategy

Model checking is considered an art. When verifying properties, to avoid state explosion only those states are included in the model which have any effect on the outcome of the said property. Since the system in question is a rail yard with 18 routes, 22 lights signals and 27 points, it is prone to state explosion. For performing verification of the various safety properties, only the routes which were in conflict with each other or shared common points (Table 3) were composed in parallel composition and the aforementioned safety properties were verified against each of the eighteen routes in question.

Results and discussion

The system verification was performed on a linux based computational server having 264 GB RAM. The deadlock freedom property was verified for every route in addition to the nine safety properties mentioned. The properties were verified with UPPAAL (Larsen et al. 1997) command line utility *verifyta* with option of reusing the state space for verifying multiple CTL statements. Their results for route *Down Home to Number 2 Platform* are given in Table 4.

Collision freedom property

Collision freedom property for route *Down Home to Number 2 Platform* is described by AG operators stating that whenever the *route selection automaton*, route3, will not be in

Table 4 UPPAAL model checker verification results for route *down home to number 2 platform*

Property	CTL statement	Result	RAM consumed	Time
p1	AG ¬route3.start ⇒ (route4.start∧route11.start ∧ route12.start∧route18.start∧route19.start)	Satisfied	>25 GB	>5 h
p2	AG¬l3.start ⇒ p29t.NormalPos∧p37t.NormalPos ∧ p18t.ReversePos∧p36t.ReversePos	Satisfied	>10 GB	>45 min
p3 for points	AG rdh2.error ⇒ rdh2.x≥ 6	Satisfied	>2 MB	>10 min
p4 for points	EF ¬ rdh2.error ∧ rdh2.x<6	Satisfied	>2 MB	>10 min
p3 for lights	AG l3.error⇒ l3.x≥ 6	Satisfied	>2 MB	>10 min
p4 for lights	EF¬l3.error ∧ l3.x<6	Satisfied	>2 MB	>10 min
p5	AG l3.cancel ⇒ AF (l3.x≥60∧ route3.clearance)	Satisfied	>500 MB	>1 h
p6	AG rdh2.error ⇒ AF ¬route3.signal_order	Satisfied	>10 MB	>30 min
Deadlock freeness	AG (not deadlock)	Satisfied	>25 GB	>5 h

its initial state i.e. ¬route3.start then it is implied that route4, route11, route12, route18 and route19 are in their initial state i.e. start.

De-railement avoidance property

As de-railment avoidance property is also a safety property, it is also described by the AG operators expressing that whenever the light sequence for a route is engaged i.e. *light driver timed automaton*, l3, is not in it initial state (i.e. ¬l3.start), then it logically implies that the point machines p29t and p37t will always be found in Normal while machines p18t and p36t will always be found in Reverse position.

Assured timer expiration properties

p3 property for both light driver (l3) and point driver (rdh2) timed automata states that error state (error) in timed automata is always reached when clock (x) value in these timed automata is six or greater. Conversely p4 property states that error state is never reached when the clock value is less than six.

Signal cancel properties

Whenever the light driver timed automaton is in cancel state, l3.cancel, for property p6, it implies that after 60 time units (clock value x greater or equal to 60), the locked out routes will be unlocked. This unlocking of routes (and subsequently points) is represented by clearance state in *route selection automata*, route3.clearance.

Fail safe functionality

This safety property states that whenever point driver timed automaton for *Down Home to Number 2 Platform* is in error state i.e. rdh2.error (ψ), then it implies that there is never going to be a state in which signal lights are instantiated via the *signal_order* state of *route selection automaton* i.e. route3.signal_order (φ).

Message sequencing charts

Message sequence charts (MSC) of the automata per route are obtained using Uppaal tool's simulation mode. Due to the large nature of these charts, they are spread onto two pages, with the third page showing the *cancelsignal* functionality of the timed automata. In all these charts, the error sequencing of points and lights are not included and it is assumed that all points and lights are in working order.

The point automata in the MSC begin with a letter *p* followed by an alphanumeric code and light automata begin with the word *light* followed by an alphanumeric code.

The top row in the MSC shows the name of the automata, and the rectangular boxes below them describe the locations of that automata. The horizontal red lines pointing from one location of an automaton to a location of another automaton represent transitions, with the channel name described above that horizontal line. The MSCs of route *Down home to number 2 platform* are shown in Figs. 18, 19 and 20.

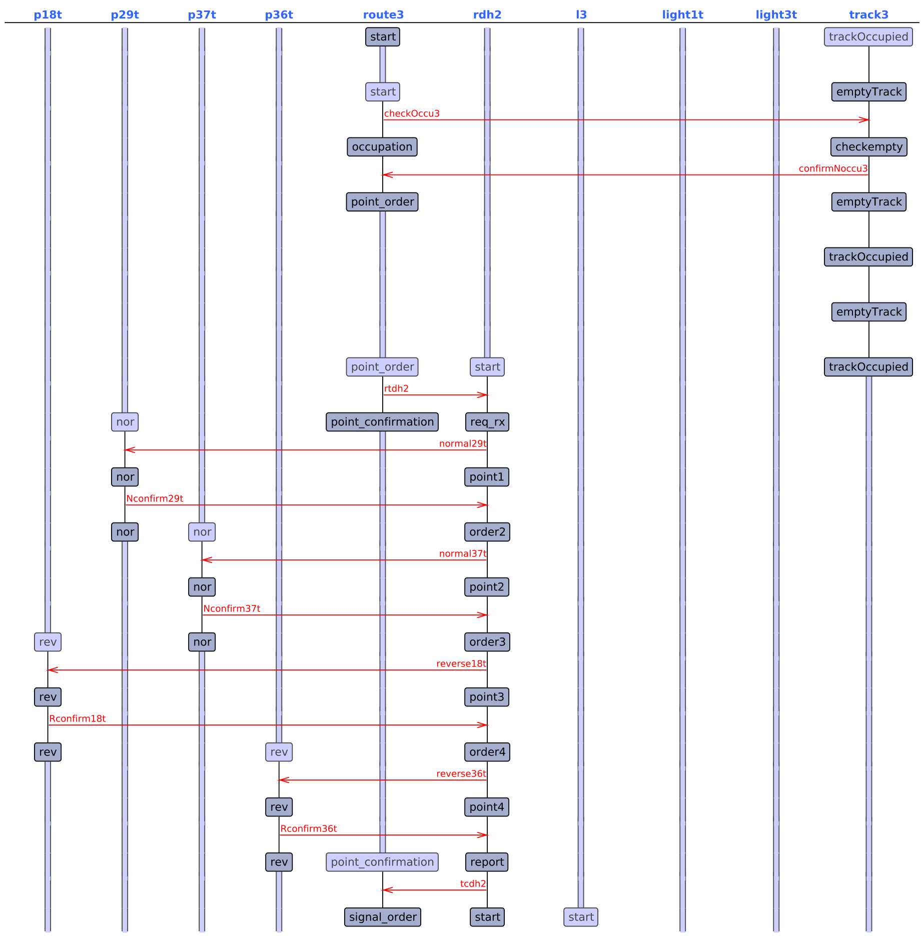


Fig. 18 Down home to number 2 platform. MSC for a route which paves the pathway for a train entering the station from the north and arriving at the number 3 platform

Conclusions

Interlocking system is a safety control system which supervises correct movement in a train yard. Since it is a real time safety critical system, it required modeling and exhaustive verification to prevent errors being introduced into the final built design. We have used time automata framework to model this system by extracting real time constraints from this safety critical system. In this paper we presented the designed model and verification of automated interlocking system model of Rawalpindi Cantt train yard in UPPAAL toolbox (Larsen et al. 1997). The simplicity and modularity of the constructed design enables us to add or delete routes in a yard or append more wayside equipment into the already existing routes with ease. Thus, if there is a new development in the yard, causing an increase in conflicting routes for a particular route, the adjustment in the design is simply an addition of a boolean variable(s) in the *route selection automata* of that route. It is advisable for every up-gradation of the routes, conflict tables should be made, as shown in this study, which will help in portraying the overall effect of an addition or deletion on the whole system.

This design removes the practice of direct engagement of points by the user and only route choices can be taken as inputs, which increases the safety aspect of the system.

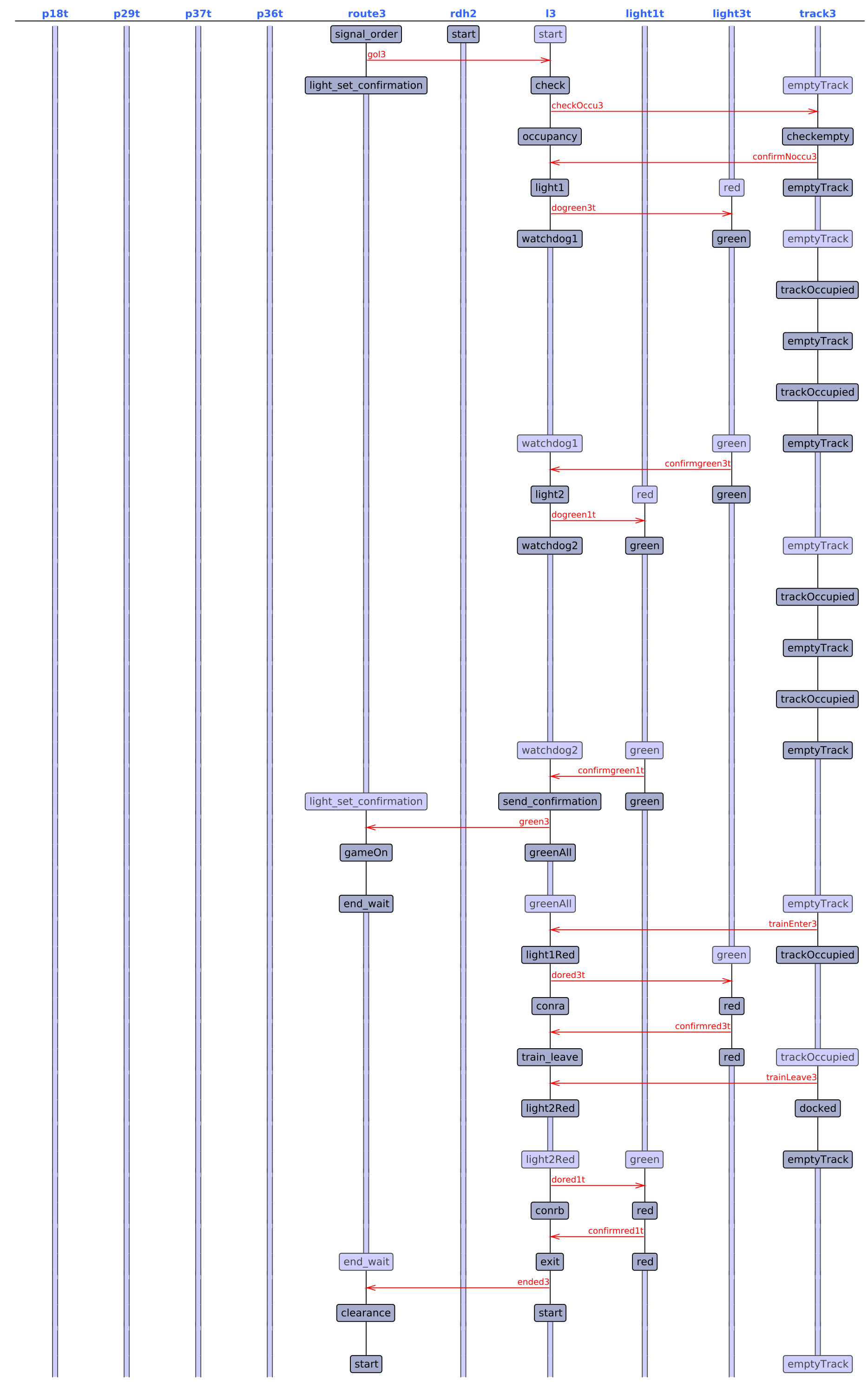


Fig. 19 Down home to number 2 platform. Sequence continued from Fig. 18

Even if a route cancellation is required in an emergency situation, the points are not directly handled by the operators and only made available after a defined delay. The engagement of points in an organized and orderly fashion is one of the hallmark features

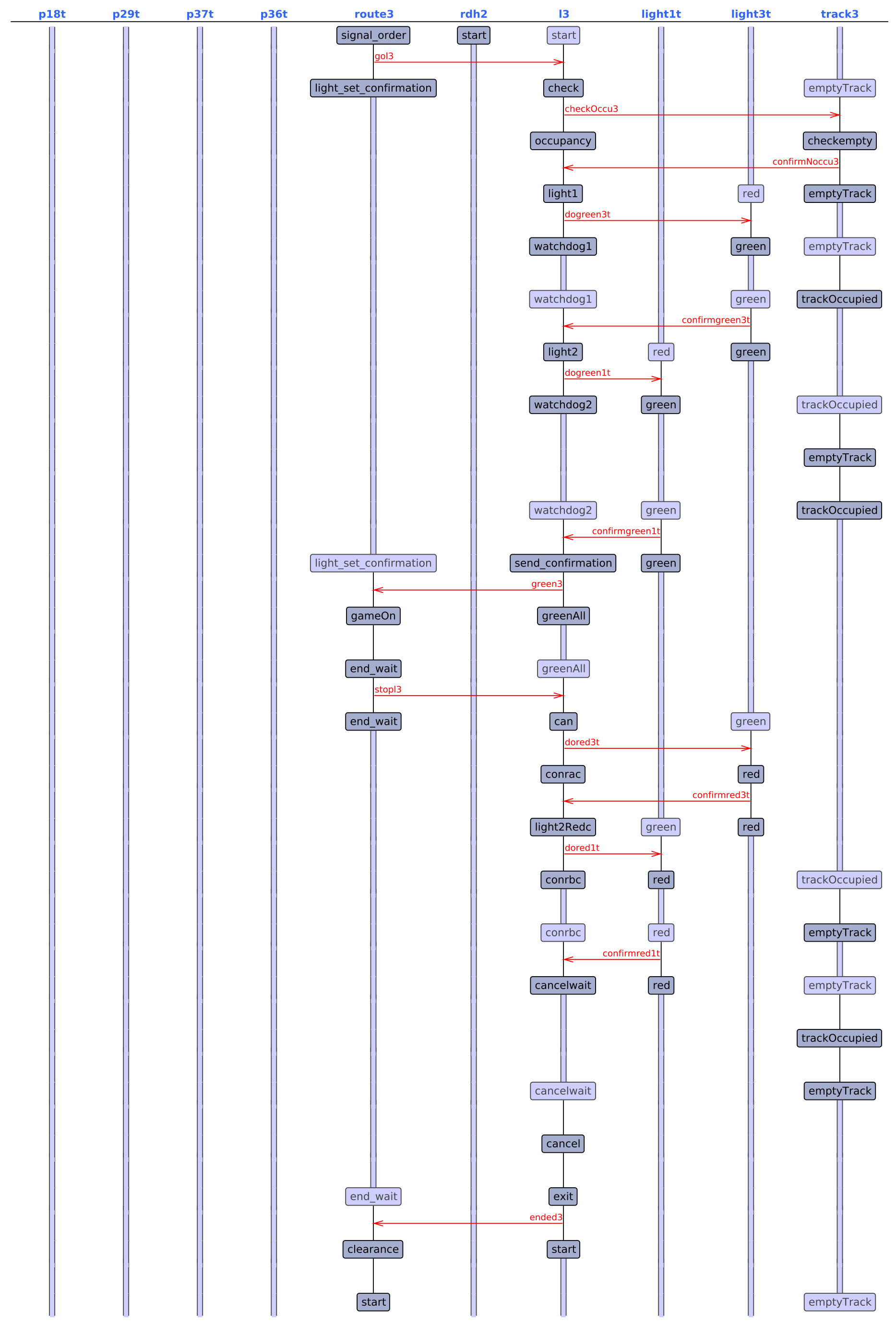


Fig. 20 Down home to number 2 platform. This MSC shows an alternative sequence starting from where Fig. 18 ends and where an order to cancel the signal is observed, otherwise the MSC remains the same as Fig. 19

of this design. Furthermore, this design does not completely handover the critical decision making in the yard, but keeps man in the loop as in the case of *cancel signal* feature.

In this paper, important properties related to the safety of the designed interlocking system of the railyard management system were successfully verified. These properties ranged from collision and de-railment avoidance to checking the correct error handling functionality of the timed automata models.

A modern rail management system for the Rawalpindi Cantt train station can be constructed by making these verified models a reference for the design and development of the interface and the infrastructure layers of the rail management system which has a huge potential in saving costs of importing international solutions for our railroading needs.

Authors' contributions

Conceived, designed and verified the models UK. Analyzed the data UK. Contributed reagents/materials/analysis tools UK, JA, TS, SM. Wrote the paper UK, JA, TS. All authors read and approved the final manuscript.

Acknowledgements

The authors would like to thank Signalling Engineering Officer and Sub Engineer of Rawalpindi Cantt train station for the provision of interlocking diagram and data related to the train station. Dr. Adnan Maqsood of RCMS, NUST for providing administration support. Dr. Prof. Umar Khattaq at GIS, NUST for access to high end server.

Competing interests

The authors declare that they have no competing interests.

Received: 1 January 2016 Accepted: 20 July 2016
Published online: 09 August 2016

References

Antoni M, Ammad N (2008) Formal validation method and tools for french computerized railway interlocking systems. In: 4th IET international conference on railway condition monitoring, 2008, IET, Derby, pp 1–10

Baier C, Katoen J-P (2008) Principles of model checking, vol 26202649. MIT press, Cambridge

Behrmann G, David A, Larsen KG (2004) A tutorial on uppaal. In: Formal methods for the design of real-time systems, Springer, Berlin, pp 200–236

Bérard B, Bidoit M, Finkel A, Laroussinie F, Petit A, Petrucci L, Schnoebelen P (2013) Systems and software verification: model-checking techniques and tools. Springer, Berlin

Boulanger JL (2015) CENELEC 50128 and IEC 62279 standards. Control, systems and industrial engineering series. Wiley, New York. https://books.google.com.pk/books?id=kEYxBwAAQBAJ

Brown S (2000) Overview of i.e.c. 61508. design of electrical/electronic/programmable electronic safety-related systems. Comput Control Eng J 11(1):6–12

Cimatti A, Clarke E, Giunchiglia E, Giunchiglia F, Pistore M, Roveri M, Sebastiani R, Tacchella A (2002) Nusmv 2: an open-source tool for symbolic model checking. In: Computer aided verification, Springer, Berlin, pp 359–364

Clarke EM, Emerson EA (1982) Design and synthesis of synchronization skeletons using branching time temporal logic. Springer, Berlin

Dincel E, Kurtulan S (2012) Interlocking and automatic operating system design with automaton method. Control Transp Syst 13:191–196

Ferrari A, Magnani G, Grasso D, Fantechi A (2011) Model checking interlocking control tables. In: FORMS/FORMAT 2010, Springer, Berlin, pp 107–115

Fokkink W (1996) Safety criteria for the vital processor interlocking at hoorn-kersenboogerd. In: 5th conference on computers in railways (COMPRAIL'96), vol 1

Fokkink W, Hollingshead P, Groote J, Luttik S, van Wamel J (1998) Verification of interlockings: from control tables to ladder logic diagrams. In: Proceedings of FMICS, vol 98. pp 171–185

Holzmann GJ (2004) The SPIN model checker: primer and reference manual, vol 1003. Addison-Wesley Reading, Boston

James P, Moller F, Nguyen HN, Roggenbach M, Schneider S, Treharne H (2014) Techniques for modelling and verifying railway interlockings. Int J Softw Tools Technol Transf 16(6):685–711

Kanso K, Moller F, Setzer A (2009) Automated verification of signalling principles in railway interlocking systems. Electron Notes Theor Comput Sci 250(2):19–31

Khan U, Ahmad J, Saeed T (2015) Real time modeling of interlocking control system of rawalpindi cantt train yard. In: 2015 13th International conference on frontiers of information technology (FIT), IEEE, 2015, pp 347–352

Larsen KG, Pettersson P, Yi W (1997) Uppaal in a nutshell. Int J Softw Tools Technol Transf 1(1):134–152

Moler F, Nguyen H, Roggenbach M, Schneider S, Treharne H (2012) Combining event-based and state-based modelling for railway verification

Olderog E-R, Dierks H (2008) Real-time systems: formal specification and automatic verification. Cambridge University Press, Cambridge

Pnueli A (1977) The temporal logic of programs. In: 18th annual symposium on foundations of computer science, 1977, IEEE, New York, pp 46–57

Satish C, Agarwal MM (2007) Railway engineering. Oxford University Press, Oxford

Siemens (2015) S700 K point machine. https://www.mobility.siemens.com/mobility/global/SiteCollectionDocuments/en/rail-solutions/rail-automation/signaling-components/s-700-k-en.pdf. Accessed 11 Aug 2015

Signals R (2013) Chief engineers division: ESG 100 signal design principles. http://www.asa.transport.nsw.gov.au/sites/default/files/asa/railcorp-legacy/disciplines/signals/esg-100.pdf. Accessed 11 Aug 2015
Signals R (2013) Chief engineers division: SPG 0719 computer-based interlocking requirements. http://www.asa.transport.nsw.gov.au/sites/default/files/asa/railcorp-legacy/disciplines/signals/spg-0719.pdf. Accessed 11 Aug 2015
Signals R (2013) Chief engineers division: SPG 0713 signalling control systems. http://www.asa.transport.nsw.gov.au/sites/default/files/asa/railcorp-legacy/disciplines/signals/spg-0713.pdf. Accessed 11 Aug 2015
Söylemez MT, Durmuş MS, Yıldırım U, Türk S, Sonat A (2011) The application of automation theory to railway signalization systems: The case of turkish national railway signalization project. In: Proceedings of the 18th IFAC world congress, pp 10752–10757
Szpyrka M (2008) Modelling and analysis of real-time systems with RTCP-nets. INTECH Open Access Publisher, Open Access, Rijeka
Turk S, Sonat A, Kuzu A, Soylemez M, Songuler O, Taralp T (2011) Automated interlocking algorithm generation from interlocking tables for railway signalization systems
Vu LH, Haxthausen AE, Peleska J (2014) Formal modeling and verification of interlocking systems featuring sequential release. In: Formal techniques for safety-critical systems, Springer, Berlin, pp 223–238
Yildirim U, Durmuş MS, Söylemez MT (2010) Fail-safe signalization and interlocking design for a railway yard: an automation petri net approach. Control Engineering Department, Istanbul Technical University, pp 1–2