International Journal on Software Tools for Technology Transfer (2018) 20:243–262
https://doi.org/10.1007/s10009-018-0482-9

Modelling and simulating a Thai railway signalling system using Coloured Petri Nets

Somsak Vanit-Anunchai[1]

Published online: 17 January 2018

Abstract

It is well known that formal verification of a large railway signalling system encounters the state explosion problem. To alleviate the problem, researchers usually concentrate on only route interlocking and abstract away other properties. Besides the route interlocking, there are also other vital properties to which failure to comply could potentially lead to danger. According to our experience, most of train accidents often involve human error and errors in other properties rather than errors in the route interlocking. Thus, we encounter a dilemma between fully automated validation of an incomplete model or partial validation of a complete model. We argue that formally modelling the complete model will be more valuable for the on-going projects of the State Railway of Thailand (SRT) because it provides insights and can be used to train new signal engineers. This paper focuses on the complete Coloured Petri Net model of a typical Thai railway signalling system: a double-line station with one passing loop. The model includes train movements that can be simulated and graphically visualized. According to SRT's signalling principles, we have identified nine properties: route interlocking, alternative overlap, flank protection, aspect sequence, approach signal release, approach lock, back lock, sectional route release and quick route release. Lessons learnt from using CPN Tools to model and validate the railway signalling systems are also discussed.

Keywords Interlocking tables · Route-based Interlocking · Approach lock · Back lock · Visualization extension

1 Introduction

Currently, the State Railway of Thailand (SRT) is undertaking several projects expanding and upgrading their railway lines. These projects involve design and installation of new signalling systems. Although SRT and contractors have been working together, the line of responsibility between both parties is quite clear. Usually SRT works on functional requirement and specifications that are arranged in the form of interlocking tables. The tables are the tabular representation specifying the routes (the allowed passage of the train) and the states/actions of all related signalling equipment. Taking the approved interlocking tables as their input, the contractors are more interested in providing the hardware and software that conform to the tables. For example, based on the approved interlocking table, the contractors create the ladder logic diagrams and rigorously test them using their simulators. Based on the tested ladder logic diagrams, the hardware and software of the interlocking equipment are manufactured. Hence, the contractors focus on verifying the ladder logic diagrams against the interlocking table, while SRT focuses on verifying the interlocking tables against their rules and regulations (signalling principles).

During the last 30 years, many research groups have been working on the application of formal methods to railway signalling systems. A comprehensive review of the research in this area can be found in [5,6]. Although most researchers focus on formal verification and validation, this paper places emphasis on formal specification instead for two reasons. First, the motivation for using formal methods is to reduce costs and increase productivity. To achieve this, the tools must be used by signal engineers. However, it is likely that signal engineers will be unable to comprehend theorem provers or model checking algorithms, so that formal validation and verification processes should be hidden. On the other hand, the signal engineers are more interested in developing the requirement specification and simulating the critical scenarios. Thus, they require modelling and simulation tools that

✉ Somsak Vanit-Anunchai
somsav@sut.ac.th

[1] School of Telecommunication Engineering, Institute of Engineering, Suranaree University of Technology, Muang, Nakhon Ratchasima, Thailand

are easy to use and have high expressive modelling power. Second, to alleviate the state explosion problem, researchers usually abstract away a lot of the details of operating procedures and concentrate only on the route interlocking[1] that prevents train collision. Besides the route interlocking, there are other vital safety properties to which failure to comply could potentially lead to danger. There are also other properties that reduce train delay and increase flexibility as well as efficiency. Are these properties independent or could they interact? Could an error in a property mask out or cause an error in another property? According to our experience, most of the train accidents often involve human error and errors in the other properties rather than errors in the route interlocking. Thus, we believe that formal specification and simulation of the complete model are more valuable for the on-going projects of the State Railway of Thailand.

1.1 Related work

One of the good candidates for the modelling of railway signalling systems is Petri nets. There have been many researchers using Petri Nets for modelling railway signalling systems, e.g. [1,8,17], but they transform the Petri Net models into other tools in order to conduct simulation or verification. For example, Sun [17] transformed Coloured Petri Net (CPN) model to a B-machine model. Hagalisletto et al. [8] also transformed their CPN model into Maude [3]. The verification and validation of the railway interlocking challenge drew a lot of attention from model checking researchers. A comparison study of applicability bounds when using NuSMV and SPIN was conducted in [7]. Their result showed that the verification of medium and large interlockings was still out of reach. To push the applicability bounds further, several techniques have been proposed. Winter [22] pursued a significant improvement in run-time and memory usage by optimizing variable and transition orderings. A pioneering work by Haxthausen [10] systematically compared modelling and verification approaches developed by two different research groups: DTU/Bremen [9,11] and Surrey/Swansea [12,13]. Both approaches were able to detect all injected errors. Haxthausen et al. [9] proposed to apply bounded model checking combined with inductive reasoning for verification and validation of interlocking systems. James et al. [13] suggested that the nature of railway systems involved events (e.g. train movement) and state-based reasoning (the interlocking). To combine event-based with state-based modelling, James et al. proposed to use CSP||B for the modelling language. Incidentally, we point out that Petri Nets already combines event-based and state-based modelling. In [13] James et al. also suggested three abstraction techniques. First, they reduced the verification problem for any number of trains to that of a two-train scenario. Second, they decomposed a large scheme plan into a set of smaller ones such that the safety of all smaller scheme plans implied the safety of the original scheme plan. Third, they abstracted a scheme plan such that checking the abstract scheme plan was enough to ensure that the required safety properties hold for the concrete plan. Although James et al. [13] assumed that the train's length was shorter than a track segment, in [12] they pushed further by allowing trains to span any number of track segments.

1.2 Previous work

In 2009 we [18] used CPNs [14] and CPN tools [2] to model and analyse the signalling system of a single track railway station. To build the model, we needed two pieces of data: the signalling layout and the interlocking tables. The CPN model was divided into two parts according to the data. First, a CPN model that mimics the signalling layout and simulates the train movements was created. Second, a generic model of the interlocking tables coded the content of the interlocking table into ML functions which are called on arc inscriptions or in guards. Modelling interlocking tables of other railway stations was simply done by changing the content of the ML functions. These ML functions were automatically generated from the interlocking table using Extensible Stylesheet Language Transformations (XSLT). By exhaustively searching for the states where trains collide, we formally verified this CPN model [18]. Nevertheless [18] had two problems. Firstly, where we had many signalling devices working together, the CPN diagram became too complex. Secondly, although the system was safe, the signalman could give sequences of route setting instructions that led the train traffic into a deadlock. Using state space generation, our CPN model generated a lot of safe terminal states that had no train collision but in which the train traffic was in a deadlock. This was inconvenient when investigating terminal states. To eliminate the first problem, we modelled in [19] the signalling layout by encoding the geographic information into *tokens*, data values that can have an arbitrarily complex (user defined) data structure. When the signalling layout is modified or rebuilt, we simply change the initial state of the model without having to modify the model structure itself.

To avoid the traffic deadlocks in the second problem, the CPN model in [20] included the automatic route setting and automatic route cancelling functions. The automatic route setting is when the route setting is triggered by track occupancy. According to SRT's rules and regulations, the signalman can set the route only when the train is approaching. The track occupancy will alarm the signalman so that

[1] No conflicting route can be used at the same time by multiple trains.

he will set a route accordingly. On the other hand, if a route has been set but no train arrives, the signalman must cancel that route. Instead of manually setting and cancelling the route by a signalman, it is possible to implement both automatic route setting and automatic route cancelling functions. However, these functions are not defined in the current SRT interlocking tables.

Despite having solved the two problems of [18] in [19] and [20], there are still another two fundamental problems. Firstly, encoding the geographic information into *tokens* in [19] made the CPN model too difficult to read. Our counterparts, SRT's signal engineers, prefer the CPN model that mimics the signalling layout and simulates the train movements. To solve this problem, in [21] we used the visualization extension[2] of CPN Tools to display the status of signalling equipment representing train movements and signal aspects. Secondly, because of abstraction and assumptions, none of our previous CPN models had included all properties in the interlocking table. Thus, we rebuilt a complete model of a double-line station with one passing loop, named Don Si Non. Our new model has included all properties in the interlocking table.

1.3 Contributions

For the sake of clarity, in [21] we avoided the complex ML functions and illustrated the unfolded version of the CPN model by instantiating arc functions with the contents excerpted from the interlocking tables. This paper is an extension version of our work in [21]. The contribution of this paper is threefold. First, this paper summarizes SRT's signalling principles and identifies nine desired properties. Second, in addition to the unfolded CPN model in [21], this paper illustrates the folded version of the CPN model which covers nine desired properties. Third, this paper introduces the generic CPN model of the train movements. It has been further refined from [19] so that the simulated train can span over one or two track circuits. By replacing the configuration data (tokens), we can reuse the same CPN diagram with other station layouts as well.

The rest of this paper is organized as follows. Section 2 summarizes the SRT's signalling principles and desired properties. Section 3 describes the CPN model of the interlocking and the train movements. Section 4 discusses lesson learnt and perspective. Section 5 presents conclusions and outlines suggested future work. We assume that the readers have some knowledge about Coloured Petri Nets and CPN Tools [14,16].

2 Introduction to Thai railway signalling system

2.1 Railway signalling plan

Typically, a railway signalling system divides the railway tracks into multiple *sections*. Inside each *section* only one train is allowed at a time. An example of signalling plan shown in Fig. 1a[3] comprises a collection of railway tracks and signalling equipment such as track circuits, points and signals. Each signalling equipment has an identification number and the operating function as follows.

Track circuits A track circuit is used to detect the presence of a train. A grey track circuit in Fig. 1a (e.g. 63T, 2-2T) is *cleared* indicating no train on the track. A yellow track circuit in Fig. 1a (e.g. 61T, 4-4T) is *set* indicating that it is reserved. A red track circuit is *occupied* indicating the possible presence of a train.

Warner signals A warner signal (e.g. 2-2, 4-2) displays two aspects: *yellow* or *green*. It informs drivers about the status of the next signal.

Inner and outer home signals An inner home signal (e.g. 1-5, 3-5, 2-4, 4-4) or an outer home signal (e.g. 1-3, 3-3) displays three aspects: *red*, *yellow* or *green*. The *red* aspect forbids the train entering the *section*. The *yellow* aspect allows the driver moving the train into the *section* but preparing to stop at the next signal. The *green* aspect allows the driver moving the train passing the *section* and entering the next *section*. The square below the red lamp in Fig. 1a is a *"call-on"* signal. The call-on signal informs a driver that he could pass the home signal at stop but the track may be occupied. The driver shall proceed the train at caution speed and watch out for any object blocking the passage of the train.

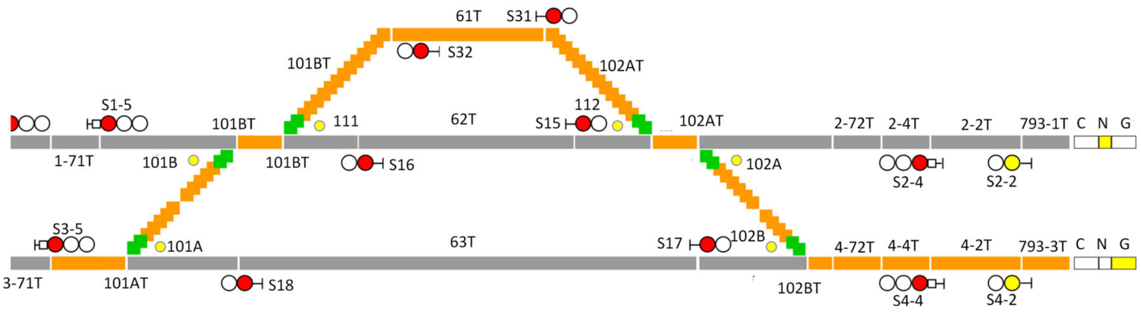
Starter signals A starter (e.g. 15, 16, 17, 18, 31, 32) display two aspects: *red* or *green*. The *red* forbids the train leaving the *station area*. The *green* allows the train leaving the *station area*.

Points A point (e.g. 101A, 101B, 111, 112, 102A, 102B) or switch is an installation used to guide a train from one to another track. Usually a point has a straight through track called "mainline" and a diverging track called "loop line". A point in the normal position lets the train move straight through but a point in the reverse position diverges the train into a loop line.

Blocks Usually a railway signalling system is divided into "within station area" and "between two stations". "Block section" is the section between two station yards. The signalling equipment that allows the train entering the block section is called *Block Instruments*. The possible states of block sections are `Normal` (no train in the block section), `Coming`

[2] Visualization Extension v.0.9 Developed By M. Westergaard and M. Assiri.

[3] For the complete picture see also Fig 4a.



(a)

TYPE	ROUTE NO.	TO	REQUIRES ROUTE NORMAL	DETECTS POINTS	Require Track Circuits		
					CLEAR	OCC	TIME
OUTER HOME	3-3(M)	3-5	4-4(1M) 4-4(1C) 18	[<N101> OR <R101 N111 AFTER 3-5(2M)R 3-5(2C)R> OR <R101 R111 AFTER 3-5(1M)R 3-5(1C)R>]	3-3T 3-71T 101AT (N102 OR 101BT)	(3-1BT OR 3-1AT OR 3-5R)	60 Sec
INNER HOME	3-5(1M)	31	2-4(2M) 3-3(C) 32(1)	R101 R111 R112 [<N102> OR <R102> AFTER 31(2)R]	101AT 101BT 61T 102AT (N102 OR 102BT) = 2-72T 2-4T OR *U = N102 OR 4-72T 4-4T OR *U	(3-71 T OR 3-3T)	60 Sec

(b)

Fig. 1 **a** The signalling plan when routes 31(2) and 3-5(1M) are set consecutively. **b** The content in the interlocking tables for setting routes no. 3-3(M) and 3-5(1M)

(a train coming into the station) and `Going` (a train going out of the station).

2.2 Signalling principles and desired properties

A collection of track circuits between the entry and the exit signal along the reserved *section* is called "*route*". The train may enter the route only when the entry signal is cleared (either yellow or green). Unlike the road traffic signals, after a train passes the entry signal, the signal is replaced to red in order to prevent another train entering that section. The route entry permission is decided by the interlocking system using safety rules and control methods specified in the agreed interlocking tables shown in Fig. 1b. The second column of the table ("ROUTE NO.") lists the route identifications which are labelled by the entry signal. The third column of the table ("TO") lists the ends of the route or the exit signals. From the entry signal 3-5, to the exit signals 31, 15 and 17 are the routes 3-5(1M), 3-5(2M) and 3-5(3M), respectively. Instead of the main signal, when the call-on signal is used, the route identifications become 3-5(1C), 3-5(2C) and 3-5(3C), respectively. The state of an unreserved route is called *"Normal"* otherwise *"Reverse"* (e.g. 3-5(1M)R, 3-5(2M)R, 31(2)R in Fig. 1b).

For ease of understanding, we define the state transition diagram of a *route* illustrated in Fig 2. The route is initially in the *Route Normal* state. After receiving the route setting command, the route enters the *SigClearing* state and the interlocking attempts to clear the entry signal. After the signal clearing conditions are fulfilled, the entry signal is cleared and the route moves to the *SigCleared* state. When the train is approaching and the approach locked conditions are fulfilled, the route enters the *ApproachLock* state. After the train passes the entry signal, the interlocking checks the *route normalization initiated* conditions. When the conditions are met, the route enters the *Route Releasing* state and the interlocking starts releasing the route section by section. When the train arrives at the exit signal, the whole route is released. The route returns to the *Route Normal* state.

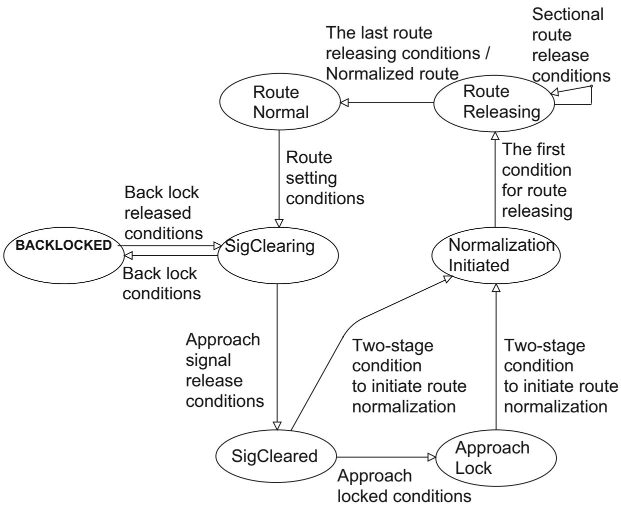

Sectional route release conditions
The last route releasing conditions / Normalized route
Route Normal
Route Releasing
Route setting conditions
The first condition for route releasing
Back lock released conditions
BACKLOCKED
SigClearing
Normalization Initiated
Back lock conditions
Approach signal release conditions
Two-stage condition to initiate route normalization
Two-stage condition to initiate route normalization
SigCleared
Approach Lock
Approach locked conditions

Fig. 2 Route's life cycle

2.2.1 Route setting

When a signalman attempts to set a route for a passage of a train, to assure the safety the interlocking system must verify the status of related track circuits, points as well as the state of other routes. The entry signal shall not display a proceed aspect (green or yellow) unless the reserved route is proven safe. Hence there is no train collision or derailment. This is called *"route interlocking"* property.

Firstly, the interlocking checks that the opposing routes listed in Column "REQUIRES ROUTE NORMAL" of Fig. 1b are not already set. These opposing routes require the same lying of point positions and track circuits, while the other conflicting routes not listed in this column are already protected by the different setting of the point positions. Secondly, the points along the route are locked in the correct position. If the related points are in the incorrect position and unlocked, the controller will attempt to move and lock them in the correct position (Column "DETECTS POINTS" of Fig. 1b). Thirdly, the track circuits along the required route are all cleared or unoccupied so that nothing obstructs the passage of the train (Column "Require Track Circuits" of Fig. 1b).

Alternative Overlap Overlap is a section beyond a stop signal that must be cleared, and points must be locked before the reserved route is set. The overlap is required because sometimes the train stops beyond a stop signal. It is possible that a route may have more than one possible overlap depending on the previous route locked with its point lying positions.

For example, route 3-5(1M) in Fig. 1b requires track circuits 101AT, 101BT, 61T, 102AT unoccupied as well as point 102 in the normal and locked position. However, if route 31(2) is already set (31(2)R) so that point 102 is already reverse and locked, route 3-5(1M) requires an additional track circuit 102BT to be cleared. Track circuits 102AT and 102BT are called *"alternative overlap"*.

Flank protection This is an important class of fail safe requirement. The equipment within the surrounding area (outside) of the reserved route that may cause an accident shall be in the safe position even if no train is expected to pass such signals, points or tracks. The points must lay in a position such that they do not give any immediate access to the route. The tracks must be either unoccupied or occupied by the object that is idle or moving in the safe direction. Even though those equipments are not located on the required route, when the route is set, they shall be locked in the safe position until the route is released.

The symbol "=" in the interlocking table (Fig. 1b) means the *"flank protection"*. The symbol "*U" means the train is going in the upward[4] direction. For example, route 3-5(1M) requires track circuits 2-72T, 2-4T to be cleared or occupied by the train going in the upward direction. In the case of alternative overlap, route 3-5(1M) also requires track circuits 4-72T, 4-4T to be cleared or occupied by the train moving in the upward direction.

After the route is set, the route's state moves from *Normal* to *SigClearing*.

Block system Although this paper focuses on the signalling system within the station area, without the *block system* it is not complete. The State Railway of Thailand uses various kinds of Block Instrument with different operating procedures. This paper selects a simplified procedure as follows. Setting the outgoing route from the starter signal into the block section requires the associated block to be `Normal`. After the outgoing route is set, the block state is changed to `Going`. The home signal can be cleared regardless of the state of associated block section. The block normalization process starts when the train passes the home signal of the designation and the last wheels are cleared from the berth[5] track.

2.2.2 Signal clearing

When the interlocking changes any signal aspects, it must maintain the *"aspect sequence"* such that the driver must see a yellow aspect before a red one otherwise he cannot stop the train at the red signal. After the mainline route [e.g. 1-5(2M), 3-5(3M), 2-4(1M), 4-4(1M)] is set, the entry signal is immediately cleared. However, when the train diverges to the loop line, the turnout speed must be significantly less than the mainline speed, otherwise the train may derail. The speed restriction is enforced by keeping the entry signal red until its berth track is occupied for 60 s. This condition is called "*approach signal release*".

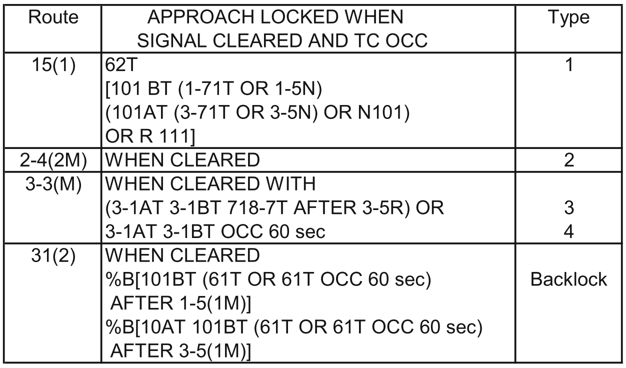
The first row of the table of Fig. 1b, and columns "Occ" show two conditions on how to clear the outer home signal 3-3. Firstly, the train occupies the berth track for 60 s. Or secondly, 3-5R means that the next inner home signal 3-5 is cleared. After the entry signal is cleared, the route's state moves to the *SigCleared* State.

2.2.3 Approach lock

When the driver sees a green aspect at the entry signal, the train usually approaches the entry signal with the full speed. If the signalman cancels this route in order to set another route instead, the entry signal suddenly turns into the red aspect. The driver will not be able to stop the train at the entry signal. To prevent a collision with other conflicting routes, the route must be maintained and locks all related signalling devices until the train stops at the exit signal. This function is called *"approach locking"*. The approach lock condition usually is the track circuit in front of (approaching) the entry signal being occupied. During the approach lock state, if the entry

[4] Outward from Bangkok.

[5] The track circuit in front of the (approaching) home or starter signal.



Route	APPROACH LOCKED WHEN SIGNAL CLEARED AND TC OCC	Type
15(1)	62T [101 BT (1-71T OR 1-5N) (101AT (3-71T OR 3-5N) OR N101) OR R 111]	1
2-4(2M)	WHEN CLEARED	2
3-3(M)	WHEN CLEARED WITH (3-1AT 3-1BT 718-7T AFTER 3-5R) OR 3-1AT 3-1BT OCC 60 sec	3 4
31(2)	WHEN CLEARED %B[101BT (61T OR 61T OCC 60 sec) AFTER 1-5(1M)] %B[10AT 101BT (61T OR 61T OCC 60 sec) AFTER 3-5(1M)]	Backlock

Fig. 3 Examples of approach lock for routes 15(1), 2-4(2M) and 3-3(M) and back lock for route 31(2)

signal is changed to red before the train passes, the route will be held for 2 min before it can be released.

Among all of the properties, the approach lock property is the most complex because the approach lock conditions vary and depend not only on the track layout but also the terrain and the train speed. However, we classify the approach lock conditions into four types illustrated by examples of the interlocking table in Fig. 3.

Type 1 is when the entry signal of the mainline route is cleared. For example, route 15(1) (mainline) after the starter signal 15 is cleared, the approach lock conditions are the following:

(1) (Track 62T is occupied) or;
(2) (Track 101BT is occupied) and (Point 111 is normal) or;
(3) (Track 1-71T is occupied) and (Point 111 and 101 are normal) and (Signal 1-5 is cleared) or;
(4) (Track 101AT is occupied) and (Point 111 is normal) and (Point 101 is reverse) or;
(5) (Track 3-71T is occupied) and (Point 111 is normal) and (Point 101 is reverse) and (Signal 3-5 is cleared)

Type 2 is when the approach signal condition is used to clear the signal. For example, the approach lock of the diverging route 2-4(2M) occurs when the home signal 2-4 is cleared.

Type 3 and Type 4 are when the outer home signal is cleared. For example, route 3-3(M), there are two approach lock conditions:

1) Track 3-1AT or 3-1BT or 718-7T[6] is occupied and the inner home 3-5 is cleared (Type 3).
2) Track 3-1AT or 3-1BT is occupied for 60 s (Type 4).

After the approach lock condition is fulfilled, the route changes its state from *SigCleared* to *ApproachLock*. Approach lock function is released when the route normalization initiates.

2.2.4 Back lock

An example of two consecutive routes 31(2) and 3-5(1M) is shown in Fig. 1a (yellow line). When route 3-5(1M) is set after routes 31(2) is set, route 3-5(1M) must use the alternative overlap. If route 31(2) could be cancelled, point 102 would be in the reverse position. Consequently, route 3-5(1M) would violate the flank protection requirement. This situation is unsafe. Thus, route 31(2) cannot be cancelled unless route 3-5(1M) is *Normal*. This is called *"Back Lock"* condition. If route 31(2) is in *SigClearing* and route 3-5(1M) is set, route 31(2) changes its state to *BACKLOCKED*. It returns to *SigClearing* after 3-5(1M) returns to *Normal*. An example of the back lock release conditions for route 31(2) is shown in Fig. 3.

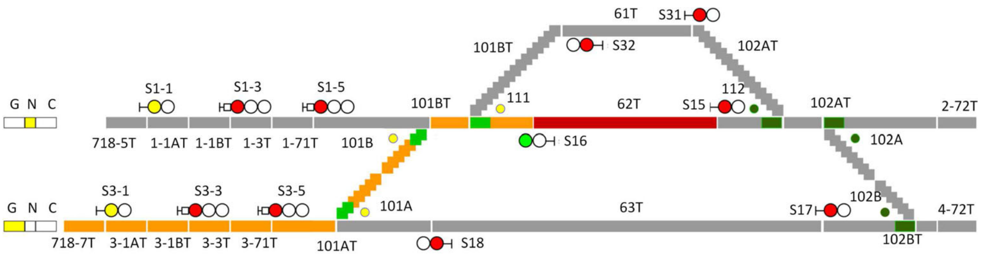
2.2.5 Route normalization

We classify the *"route normalization"*[7] process into two procedures: automatic route release by the train movement and manual route cancellation by the signalman. The manual cancellation is divided into three categories. First, the emergency route release is used when some signalling equipment fails. After the signalman issues this emergency command, the entry signal turns to red immediately and the interlocking will delay for 4 min before normalizing the route. Second, when the signalman attempts to cancel the approach locked route, the entry signal turns to red immediately and the interlocking will delay for 2 min before normalizing the route. Third, when the signalman cancels the route without approach lock, the entry signal turns to red and the route is normalized immediately. This paper focuses on only automatic route release by the train movement.

Figure 4a shows the station yard when route 16(1) is set. The tracks in yellow are the reserved route. The track in red is occupied by a train. Figure 4b illustrates an example of the conditions to release route 16(1) excerpted from the interlocking tables. The fourth and fifth columns of Fig. 4b show two-stage condition to initiate normalizing procedure: (Track 101BT occupied and cleared) and (Track 101AT occupied). After this two-stage condition is fulfilled, the route enters *RouteReleasing* state. The routes that uses the call-on signal do not have the approach lock property. Therefore, the two-stage condition could also be applied to the routes using call-on in the *SigCleared* state.

[6] See Fig. 4a.

[7] Route normalization: bring the route to the *Normal* state. Route Release: cease the route reservation so that the other routes can reuse the released resources.



(a)

TYPE	ROUTE NO.	TO	ROUTE NORMALIZATION INITIATED		SECTIONAL ROUTE RELEASING		ROUTE LOCKING RELEASED BY	
			TC OCC and CLEAR	TC OCCUPIED	RELEASES POINT	REQUIRES TC CLEAR	WHEN SETTING ROUTE	TC CLEAR
STATER	16(1)	696-8	101BT	101AT	101	101BT,101AT	3-5(1M), 3-5(1C)	101AT
					111	101BT	3-5(2M), 3-5(2C)	101AT, 101BT
							3-3(M), 3-3(C)	3-3T, 3-71T
							None	3-3T,3-71T 101AT,101BT

(b)

Fig. 4 **a** The station yard when route 16(1) is set. **b** The content excerpted from the interlocking table for releasing route no. 16(1)

2.2.6 Sectional route release

The route reservation enforces that all points along the route cannot be used by another train until the train clears the last point. This is inconvenient for a large yard with more concurrent train movements. A relaxation called *"sectional route release"* is adopted. While the train passes each section, it releases the locking affecting that section so that the points cleared by the train can be reused by other trains. The conditions of sectional route release for each point are shown in the sixth and seventh columns of Fig. 4b.

2.2.7 Quick route release

For shunting or track work, the train may leave the platform track into the block section and then return to the station. Before the incoming route can be set, the outgoing route has to be cancelled. The cancellation usually involves a long delay which is inconvenient and inefficient. Instead of cancelling, the outgoing route can be released earlier when the signalman attempts to set the return route. The last two columns in the table of Fig. 4b are the conditions of *"quick route release"*. Instead of driving the train beyond outer home signal 3-3 before releasing the route 16(1), the route 16(1) can be released earlier (in front of the inner home 3-5) if the signalman attempts to set route 3-5(1M) or 3-5(2M).

3 The Coloured Petri Net model

Formal methods are techniques based on mathematically defined syntax and semantics for the specification, development and verification of software and hardware systems. They remove ambiguities and are indispensable for checking correctness of high-integrity systems. Coloured Petri Net (CPN) [14,16] is a formal method widely used for constructing and analysing models of concurrent systems. Extending from classical Petri Nets, CPNs inherit the concept of places, transitions, tokens and firing. They preserve useful properties of Petri Nets and extend the formalism to allow the distinction between tokens by attaching a data value to them. This attached data value has an arbitrarily complex type and can be manipulated using a functional programming language, Standard ML. An important advantage of CPNs is its graphical notation with the abstract data types providing conciseness with a high level of expressive modelling power. Our CPN model has been created and maintained using CPN Tools [2], a software package for the creation, editing, simulation and state space/reachability analysis of CPNs. It supports the hierarchical construction of CPN models [16], using constructs called *substitution transitions*. These transitions hide the details of subnets and allow further nesting of substitution transitions. This allows a complex specification to be managed as a series of hierarchically related *pages* which are visualized in a hierarchy page, automatically generated.

Figure 5 shows the hierarchy structure of our railway signalling CPN model. It comprises 24 pages, 44 places and 56 transitions. The station yard comprises 4 point machines, 16 main signals, 6 call-on signals, 25 track circuits and 34 routes.

The CPN model is divided into two parts, interlocking and train movements. The first part is the CPN model of the interlocking comprising CPN pages: `RouteReleased`;

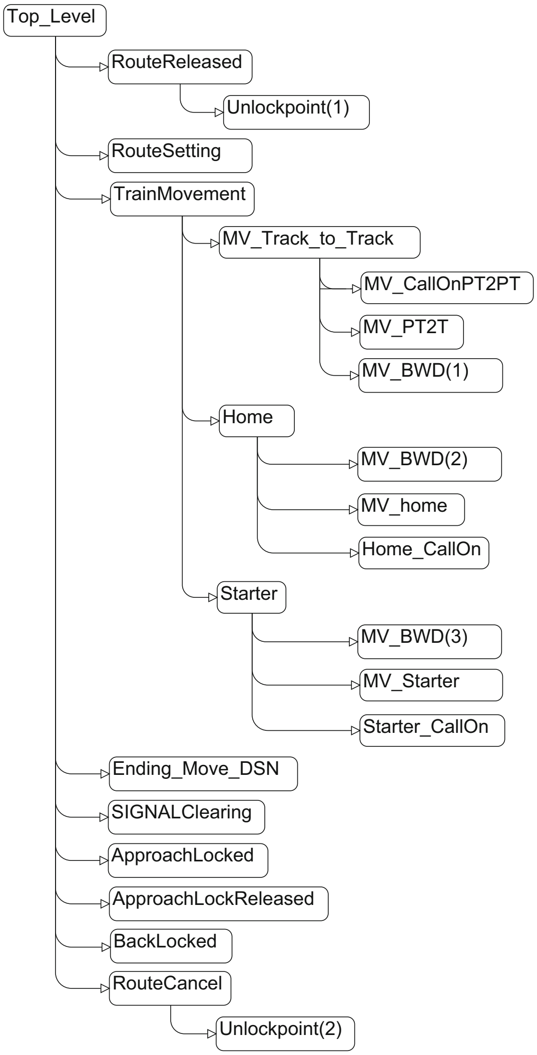


Fig. 5 The `Hierarchy` page

`RouteSetting`; `SIGNALClearing`; `ApproachLocked`; `Approach-LockReleased`; `BackLocked`; and `RoutCancel` shown in Fig. 5. In these CPN pages, the ML functions on the arc and guard inscriptions contain the information from each column of the interlocking tables. Usually, these ML functions are the “case (route) of" commands comprising the content from every row of the interlocking table that is very lengthy. To made it easier to understand in [21], we instantiated these ML functions (case command) with the content from a single row (single route). Differing from [21], this paper adds the folded version of the CPN model and its associated ML functions. These ML functions are too long to be listed in this paper so that only a portion of each ML function related to the explanation is shown in the figures.

Although the interlocking table of each railway station has different contents, their properties are essentially the same. To create a prototype model, we manually extract the content of the interlocking table and code them into the ML functions. To model interlocking tables of other railway stations, we simply change the content of the ML functions while using the same CPN net structures. In 2009 we [18] successfully extracted the ML functions (Text) from the interlocking tables (Excel). Illustrated in Fig. 6, the interlocking table in Microsoft Excel was transformed to XML and then it was transformed to ML functions using Extensible Stylesheet Language Transformations (XSLT). All operations were done automatically using Microsoft Excel and Microsoft Word. Unfortunately, the CPN model described in this paper is more complex than the one presented in [18]. Currently, we are still developing tools using XSLT script to automatically extract the contents from the interlocking tables to the ML functions. On the other hand, we discover that updating the ML function manually together with model simulation is very useful for training new signal engineers so that they can learn how the interlocking process works.

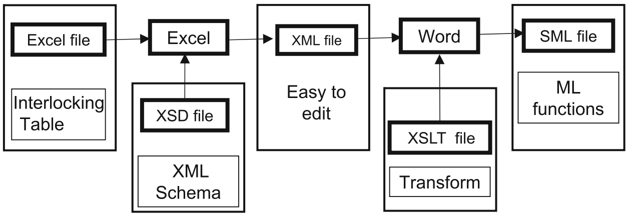


Fig. 6 Transformation of the interlocking table to ML functions using XSLT (from [18])

The second part is the CPN model of the train movements comprising CPN pages (Fig. 5): `TrainMovement` and `Ending_Move_DSN`. The model contains the track layout information used for simulating the train movements. Our previous work [18–20] assumed that the train had no length and occupied one track at a time. Instead this paper requires two conditions on the train presence. First, when the front wheels occupy the track ahead of the movement, this condition triggers the approach lock function. Second, when the last rear wheels do not occupy the track behind of the movement, this condition triggers the block normalization and turns off the call-on lamp. Thus, this paper shall assume that the train's length is shorter than a track segment and a train may span either one or two track segments.

The railway signalling model comprises nine substitution transitions (represented by double-line rectangles in Fig. 7) arranged according to the typical operating sequence of a route. Due to the space limitation, we choose to discuss `SetRoutes`; `ClearSignals`; `ApproachLocked`; `BackLocked`; `ReleaseRoutes`; `TrainMovement`; `Home`; and `MV_HOME`. These in fact cover vital information of our CPN model.

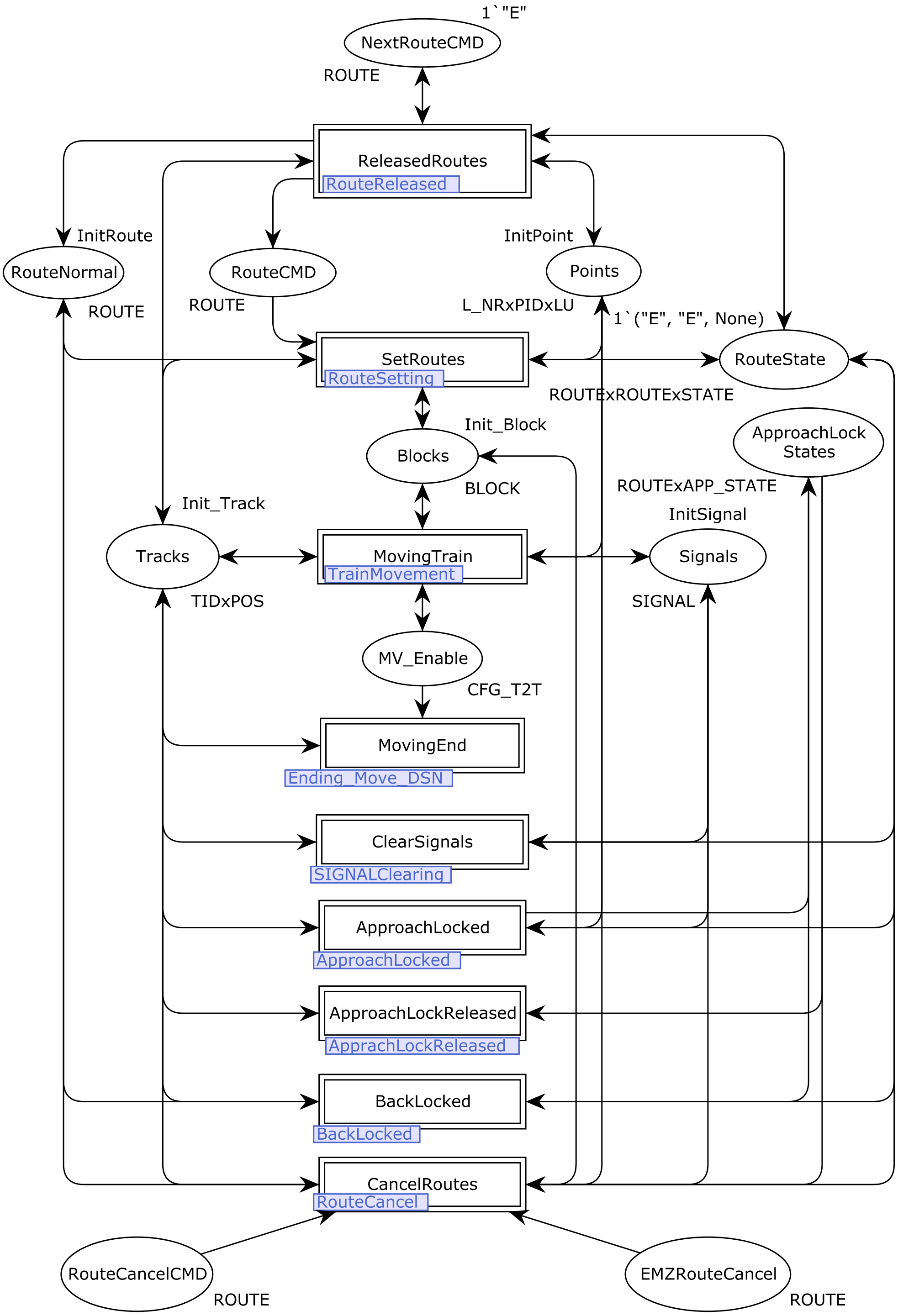


Fig. 7 The `Top_Level` page

3.1 Global Declarations and route's state

Figure 8 shows the global declarations that define the data structures associated with the model. The status of signalling apparatus is captured by four places in Fig. 7 (represented as ellipses): `Tracks` typed by `TIDxPOS`; `Signals` typed by `SIGNAL`; `Points` typed by `L_NRxPIDxLU`; and `Blocks` typed by `BLOCK`. The data types of signalling apparatus are a product or record of device identification and its status (line 1–16 of Fig. 8). State transitions (line 19 of Fig. 8) are defined according to the route's life cycle in Fig. 2.

```
(*———Point————————-*)-
colset NR = with Normal | Reverse;
colset PID = int with 98..115;
colset LU = with LOCK | UNLOCK;
colset NRxPIDxLU = product NR * PID * LU;
colset L_NRxPIDxLU = list NRxPIDxLU;
(*———Track Circuits———-*)-
colset TD = with noTrain | TrainCallOn | TrainUP
           | TrainDOWN | TrainSTOP | TrackFailed;
colset TIDxPOS = product  STRING * TD;
(*———Signal————————*)-
colset GYR = with G | Y | R | CCC;
colset SIGNAL = product STRING * GYR ;
(*———Block————-*)-
colset BLOCK_POS = with COMING | NORMAL| GOING;
colset BLOCK = record bid:STRING * pos:BLOCK_POS;
(*———Route's State———-*)-
colset ROUTE = STRING;
colset STATE = with SigClearing | SigCleared
      | AppLocked | BACKLOCKED | Normalize_Init
      | RouteReleasing | None;
colset APP_STATE = with APP_LOCKED|BACK_LOCKED
                   | APP_LOCK_Releasing;
colset ROUTExREVxSTATE =  product ROUTE * ROUTE
                        * STATE;
colset ROUTExAPP_STATE = product ROUTE
                        * APP_STATE;
colset CFG_T2T  = product STRING
                        * STRING;
```

Fig. 8 Global Declarations

3.2 Setting routes

Substitution transition SetRoutes in Fig. 7 is linked to the second level CPN page named RouteSetting which plays the central role of the route interlocking. The states of signalling equipments are contained and linked via the port–socket places: Points; Tracks; Blocks; and RouteNormal. Figure 9a shows the unfolded version of the RouteSetting page that illustrates the route setting condition of the route 3-5(1M). Using pattern matching, transition SetRoute is enabled and then executed when the states of signalling equipments (tokens) match the input arcs and guard expressions.

3.2.1 Checking the opposing routes

The expression on the input arc from place RouteNormal (Fig. 9a) illustrates that setting route 3-5(1M) requires tokens, typed by multi-set of string, 1'"2-4(2M)"++ 1'"3-3(C)" ++ 1'"32(1)" ++ 1'"3-5(1M)".[8] After transition SetRoute is executed, all tokens return to the place except "3-5(1M)" because route 3-5(1M) is no longer *Normal*. Figure 9a illustrates the conditions of route setting for only one route, 3-5(1M). This station yard has a total of 34 routes so that 34 unfolded CPN pages are required. However, we can fold the CPN diagrams of all 34 routes into one CPN diagram as shown in Fig. 9b. In the folded version (Fig. 9b), this required route listed in Column "REQUIRES ROUTE NORMAL" (Fig. 1b) is modelled by the function rroute (route).

3.2.2 Checking the points

According to the guard expression in Fig. 9a, if any of the points (101, 111, 112) are locked in normal position, the route 3-5(1M) cannot be set. If point 102 in the overlap is locked in reverse position without route 31(2) being used, the route 3-5(1M) cannot be set. The output arc expression towards transition SetPOINTLock shows that points (101, 111, 112) will be set to reverse and then locked. However, the required position of point 102 depends on whether route 31(2) is set or not.

In the folded version of Fig. 9b, we divide the required points into two groups: first the points that are located along the routes (inside) and second the points that are located within the overlap tracks and flank protection areas (outside). The required points within the route listed in the interlocking table is stored in the function rpoint(route). The required positions will be checked against their current states by the guard function ck_points_inside_route(). The required points outside the route listed in the interlocking table, is stored in the function rWhenPoint (route,rev). This required positions will be checked against their current states by guard function ck_points _outside_route(). It is possible that some routes do not have any point along their routes and some routes do not have any point inside their overlap tracks.

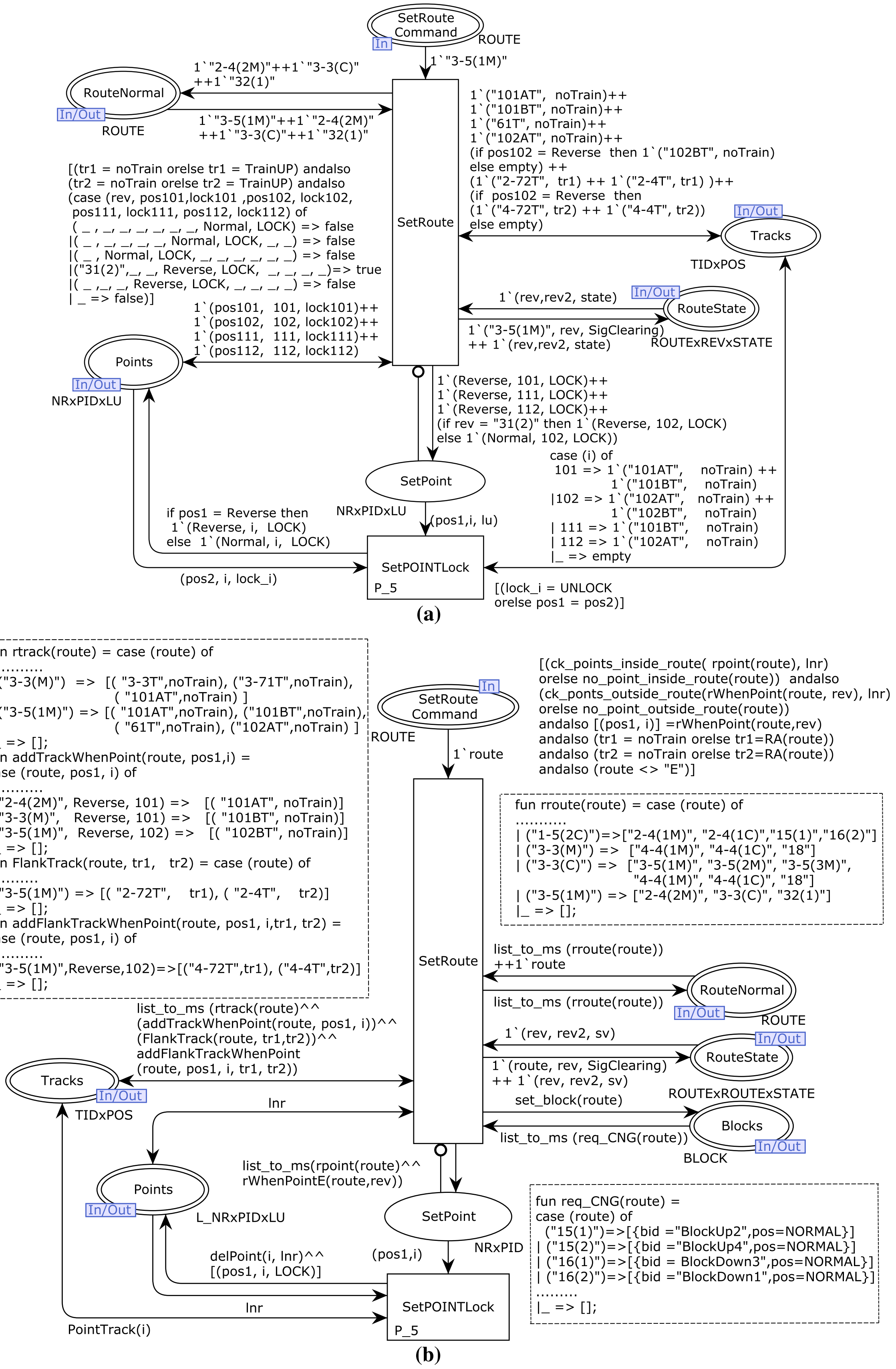
3.2.3 Checking the track circuits

According to the expression on the input–output arc from place Tracks in Fig. 9a, the required track circuit conditions can be divided into four groups. First, route 3-5(1M) requires track circuits 101AT, 101BT, 61T, 102AT unoccupied. Second, for alternative overlap when point 102 is reverse, route 3-5(1M) also requires track circuit 102BT to be cleared. Third, for flank protection, route 3-5(1M) requires track circuits 2-72T, 2-4T to be cleared or occupied by the train going in the upward direction. Fourth, for flank protection in case of alternative overlap, it also requires track circuits 4-72T, 4-4T to be cleared or occupied by the train moving in the upward direction. Figure 9b models the track circuit requirements using four ML functions on the arc connected to the place Tracks. Each function corresponds to each group in Fig. 9a, respectively.

3.2.4 Checking the block status

The details of block conditions are not included in this paper because they involve a lot of signalling equipments of adja-

[8] "++" is the addition operator over multi-set.



```
[(tr1 = noTrain orelse tr1 = TrainUP) andalso
(tr2 = noTrain orelse tr2 = TrainUP) andalso
(case (rev, pos101,lock101 ,pos102, lock102,
 pos111, lock111, pos112, lock112) of
 ( _ , _, _, _, _, _, _, Normal, LOCK) => false
|( _ , _, _, _, _, Normal, LOCK, _, _) => false
|( _ , Normal, LOCK, _, _, _, _, _, _) => false
|("31(2)",_, _, Reverse, LOCK, _, _, _, _)=> true
|( _ ,_, _, Reverse, LOCK, _, _, _, _) => false
| _ => false)]
```

```
case (i) of
 101 => 1`("101AT",   noTrain) ++
          1`("101BT",   noTrain)
|102 => 1`("102AT",  noTrain) ++
          1`("102BT",   noTrain)
| 111 => 1`("101BT",   noTrain)
| 112 => 1`("102AT",   noTrain)
|_ => empty
```

```
[(lock_i = UNLOCK
orelse pos1 = pos2)]
```

```
fun rtrack(route) = case (route) of
............
| ("3-3(M)")  =>  [( "3-3T",noTrain), ("3-71T",noTrain),
                   ( "101AT",noTrain) ]
| ("3-5(1M)") => [( "101AT",noTrain), ("101BT",noTrain),
                   ( "61T",noTrain), ("102AT",noTrain) ]
|_ => [];
fun addTrackWhenPoint(route, pos1,i) =
case (route, pos1, i) of
............
|("2-4(2M)", Reverse, 101) =>   [( "101AT", noTrain)]
|("3-3(M)",   Reverse, 101) =>  [( "101BT", noTrain)]
|("3-5(1M)",  Reverse, 102) =>   [( "102BT", noTrain)]
|_ => [];
fun FlankTrack(route, tr1,   tr2) = case (route) of
...........
|("3-5(1M)") => [( "2-72T",    tr1), ( "2-4T",    tr2)]
|_ => [];
fun addFlankTrackWhenPoint(route, pos1, i,tr1, tr2) =
case (route, pos1, i) of
............
|("3-5(1M)",Reverse,102)=>[("4-72T",tr1), ("4-4T",tr2)]
|_ => [];
```

```
[(ck_points_inside_route( rpoint(route), lnr)
orelse no_point_inside_route(route))  andalso
(ck_ponts_outside_route(rWhenPoint(route, rev), lnr)
orelse no_point_outside_route(route))
andalso [(pos1, i)] =rWhenPoint(route,rev)
andalso (tr1 = noTrain orelse tr1=RA(route))
andalso (tr2 = noTrain orelse tr2=RA(route))
andalso (route <> "E")]
```

```
fun rroute(route) = case (route) of
...........
| ("1-5(2C)")=>["2-4(1M)", "2-4(1C)","15(1)","16(2)"]
| ("3-3(M)") =>  ["4-4(1M)", "4-4(1C)", "18"]
| ("3-3(C)") =>  ["3-5(1M)", "3-5(2M)", "3-5(3M)",
                  "4-4(1M)", "4-4(1C)", "18"]
| ("3-5(1M)") => ["2-4(2M)", "3-3(C)", "32(1)"]
|_ => [];
```

```
fun req_CNG(route) =
case (route) of
  ("15(1)")=>[{bid ="BlockUp2",pos=NORMAL}]
| ("15(2)")=>[{bid ="BlockUp4",pos=NORMAL}]
| ("16(1)")=>[{bid = BlockDown3",pos=NORMAL}]
| ("16(2)")=>[{bid ="BlockDown1",pos=NORMAL}]
........
|_ => [];
```

Fig. 9 **a** An unfolded CPN page: RouteSetting [only 3-5(1M)]. **b** The folded CPN page: RouteSetting (all routes) cent stations on both sides. Figure 9a does not include place Blocks because the routes from the signal 3-5 are the routes coming inside the station area. They do not require the blocks between stations. On the other hand, when the train departs from the starter signal into the block, the routes (e.g. 15, 17, 16, 18) require the block to be Normal (no train in the block). Figure 9b models this event using the function req_CNG(route). After the outgoing route is set, function set_block(route) changes the block state to Going. The block normalization is modelled in the CPN page named MV_Home.

3.2.5 Transition SetPOINTLock

When no train occupies the associated point track, transition SetPOINTLock sets and locks the point in the required position. Setting the next route cannot proceed unless place SetPoint is empty or all points previously required are set and locked. This requirement is modelled using an inhibitor arc from place SetPoint to transition SetRoute. In our previous CPN model [18–20], we moved and locked the points before checking all route setting conditions. We found that there were a lot of states in which the required points were locked in the conflict position because they were already in use by other routes. To alleviate the state explosion, in the first transition SetRoute checks all conditions of route setting. If all conditions are matched, the route is set. Then transition SetPOINTLock moves and locks all related points. However, before transition SetPOINTLock finishes its task, no other transition can be executed. Thus, transition SetPOINTLock has the highest priority. A transition of CPN Tools has a priority inscription which is an integer expression. The lesser value of the priority inscription is, the higher priority transition has. CPN Tools provides a default priority value set to 1000. For example, in Fig. 9b, transition SetPOINTLock has a priority variable P_5 set to 5.

3.3 Clearing signals

Substitution transition ClearSignals in Fig. 7 is linked to the second level CPN page named SIGNALClearing in Fig. 10a. This CPN page demonstrates the clearing of outer home signal 3-3 and warner 3-1 when the route's state is SigClearing. The warner 3-1 is simply a repeater of the home 3-3. When 3-3 is red, 3-1 must be yellow. When 3-3 is yellow or green, 3-1 must be green. This properties are captured by transitions HOME_Y and HOME_G. The outer home 3-3 can be changed from red to yellow by two methods. First, transition TrainSTOP is when the train occupies the berth track (3-1AT or 3-1BT) more than 60 s. The token TrainSTOP is used to represent 60 s of train presence. Second, transition Sig_ahead Cleared is when the next inner home signal 3-5 is cleared (either yellow of green). After the signal 3-3 is cleared, the route's state goes to SigCleared.

Figure 10b demonstrates the folded version of SIGNAL Clearing page. Transition TrainStop models the approach signal release property of every home and starter signals. Besides approach signal release, we also bind the immediate clearing of mainline routes into this transition by assigning tc_id1 to a *dummy* track circuit which is always occupied.

3.4 Approach locked

Substitution transition ApproachLocked in Fig. 7 is linked to the second level CPN page named ApproachLocked shown in Fig. 11b. An unfolded version in Fig. 11a illustrates examples of approach locking the routes 15(1), 2-4(2M) and 3-3(M). Each transition corresponds to each example in Sect. 2.2.3, respectively.

The folded version of the ApproachLocked page is shown in Fig. 11b. Transition Type1 MainLine models the approach lock conditions of all mainline routes. The conditions vary and depend on which tracks the train approaches. The diverging route is approach locked immediately after the entry signal is cleared. Transition Type2 DivergingRoute represents this behaviour.

After the outer home signal is cleared, transition Type3 OuterHome locks the route when the berth track is occupied and the inner home signal is cleared. Another approach lock condition modelled by transition Type4 OuterHome is when the berth track is occupied with a token TrainSTOP.

3.5 Back locked

Substitution transition BackLocked in Fig. 7 is linked to the second level CPN page named BackLocked shown in Fig. 12b. An unfolded version in Fig. 12a illustrates an example of back locking the route 31(2) after 3-5(1M) is set.

The folded version of the BackLocked page is shown in Fig. 12b. In the guard of the first transition, function LockAfter(route, after) contains all possible pairs of the routes that have this back lock property. After the first transition is executed, the route's state enters the BACKLOCKED state and a token is put into place AppLock State to inhibit the route cancel command. When the second transition is executed, the route returns to SigClearing state and the token in place AppLock State is taken out.

3.6 Releasing route

Substitution transition ReleasedRoutes in Fig. 7 is linked to the second level CPN page named RouteReleased shown in Fig. 13b. An unfolded version in Fig. 13a illustrates an example of releasing the route 16(1) corre-

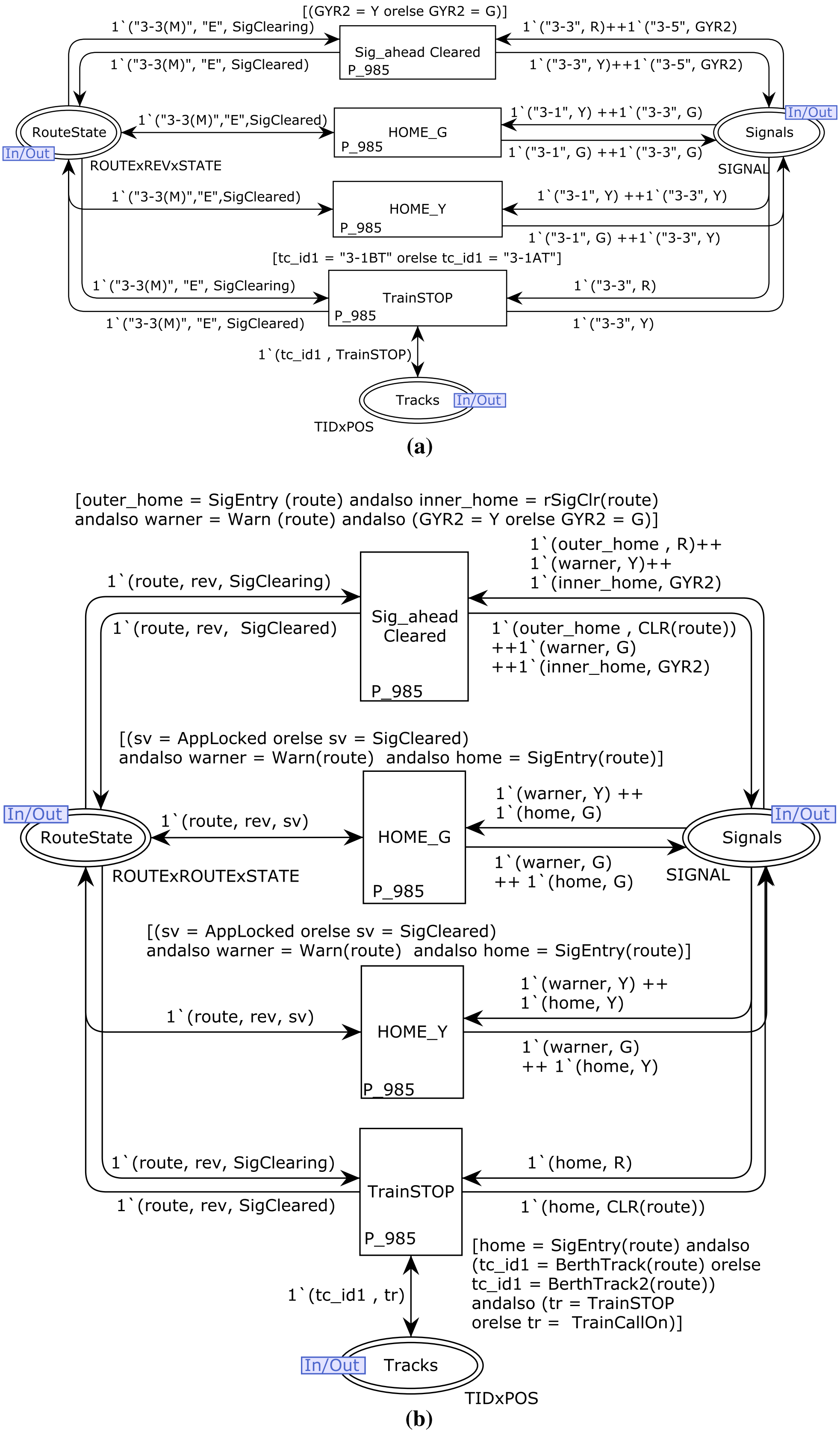

[(GYR2 = Y orelse GYR2 = G)]
1`("3-3(M)", "E", SigClearing)
1`("3-3", R)++1`("3-5", GYR2)
Sig_ahead Cleared
P_985
1`("3-3(M)", "E", SigCleared)
1`("3-3", Y)++1`("3-5", GYR2)
1`("3-3(M)","E",SigCleared)
HOME_G
P_985
1`("3-1", Y) ++1`("3-3", G)
1`("3-1", G) ++1`("3-3", G)
RouteState
In/Out
ROUTExREVxSTATE
Signals
In/Out
SIGNAL
1`("3-3(M)","E",SigCleared)
HOME_Y
P_985
1`("3-1", Y) ++1`("3-3", Y)
1`("3-1", G) ++1`("3-3", Y)
[tc_id1 = "3-1BT" orelse tc_id1 = "3-1AT"]
1`("3-3(M)", "E", SigClearing)
TrainSTOP
P_985
1`("3-3", R)
1`("3-3(M)", "E", SigCleared)
1`("3-3", Y)
1`(tc_id1 , TrainSTOP)
Tracks
In/Out
TIDxPOS
(a)
[outer_home = SigEntry (route) andalso inner_home = rSigClr(route)
andalso warner = Warn (route) andalso (GYR2 = Y orelse GYR2 = G)]
1`(outer_home , R)++
1`(warner, Y)++
1`(inner_home, GYR2)
1`(route, rev, SigClearing)
Sig_ahead
Cleared
P_985
1`(route, rev, SigCleared)
1`(outer_home , CLR(route))
++1`(warner, G)
++1`(inner_home, GYR2)
[(sv = AppLocked orelse sv = SigCleared)
andalso warner = Warn(route) andalso home = SigEntry(route)]
1`(warner, Y) ++
1`(home, G)
In/Out
RouteState
1`(route, rev, sv)
HOME_G
P_985
Signals
In/Out
ROUTExROUTExSTATE
1`(warner, G)
++ 1`(home, G)
SIGNAL
[(sv = AppLocked orelse sv = SigCleared)
andalso warner = Warn(route) andalso home = SigEntry(route)]
1`(warner, Y) ++
1`(home, Y)
1`(route, rev, sv)
HOME_Y
P_985
1`(warner, G)
++ 1`(home, Y)
1`(route, rev, SigClearing)
TrainSTOP
1`(home, R)
1`(route, rev, SigCleared)
1`(home, CLR(route))
P_985
[home = SigEntry(route) andalso
(tc_id1 = BerthTrack(route) orelse
tc_id1 = BerthTrack2(route))
andalso (tr = TrainSTOP
orelse tr = TrainCallOn)]
1`(tc_id1 , tr)
In/Out
Tracks
TIDxPOS
(b)

Fig. 10 **a** An unfolded CPN page: SIGNALClearing. **b** The folded CPN page: SIGNALClearing

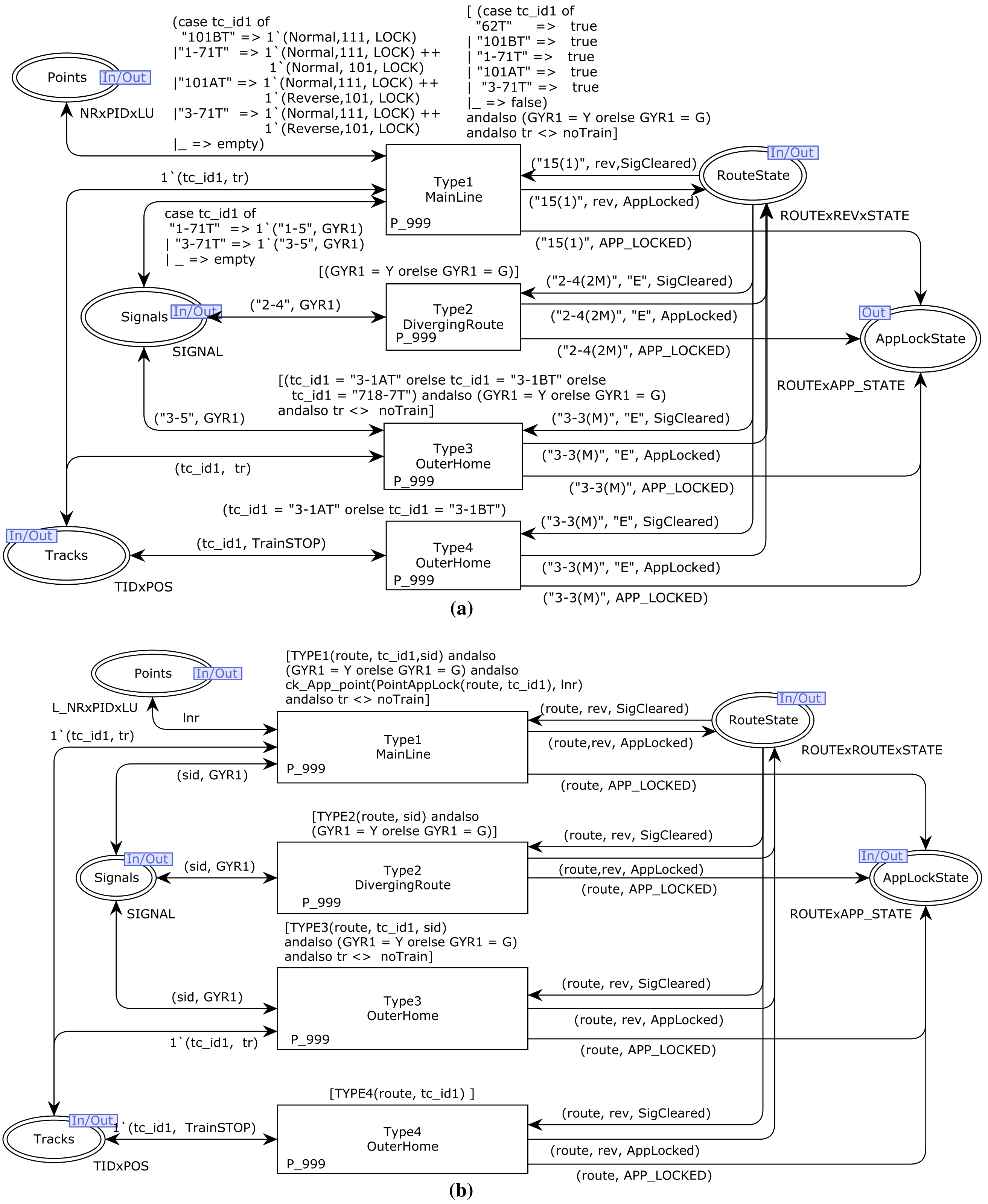


Fig. 11 **a** An unfolded CPN page: ApproachLocked. **b** The folded CPN page: ApproachLocked

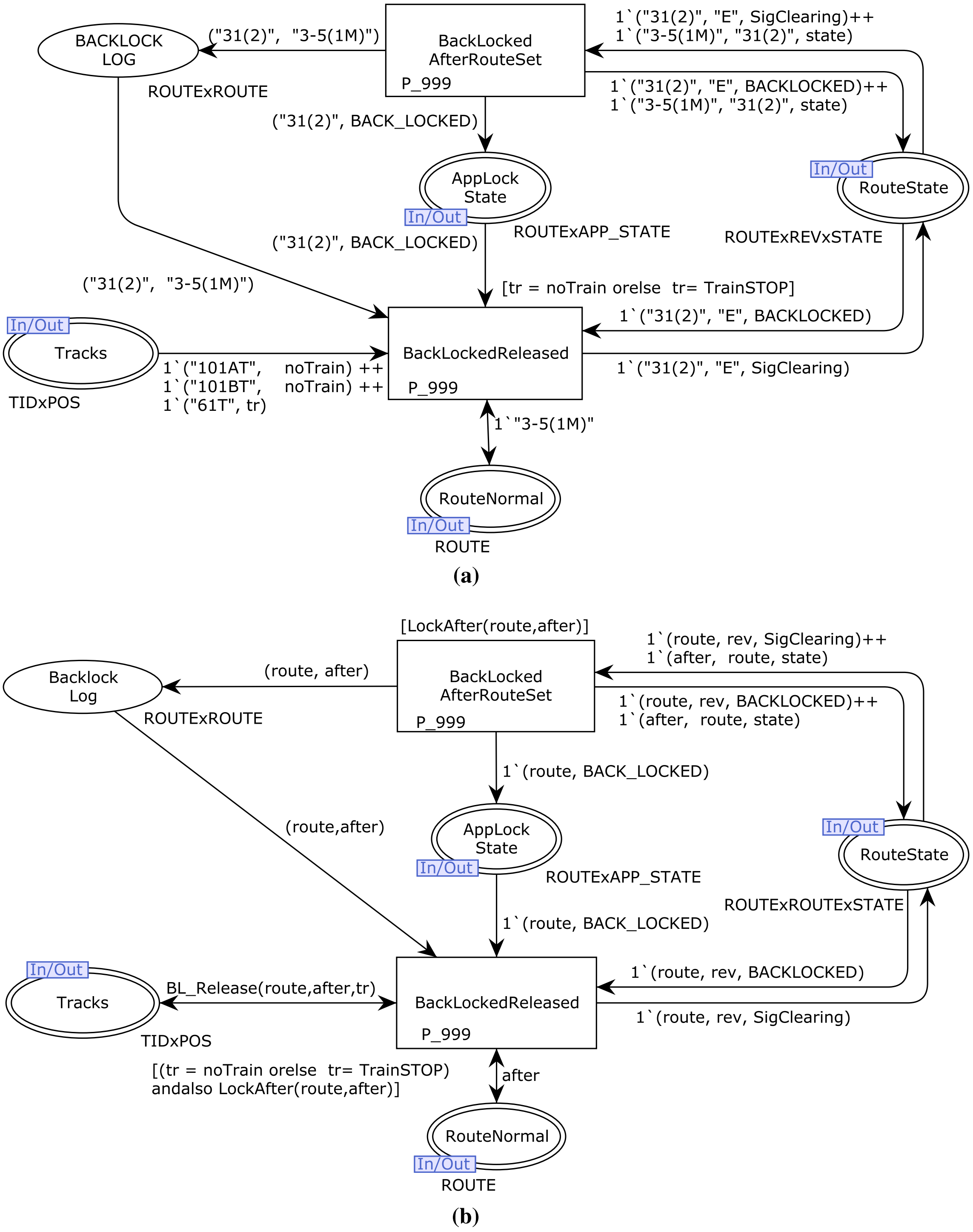
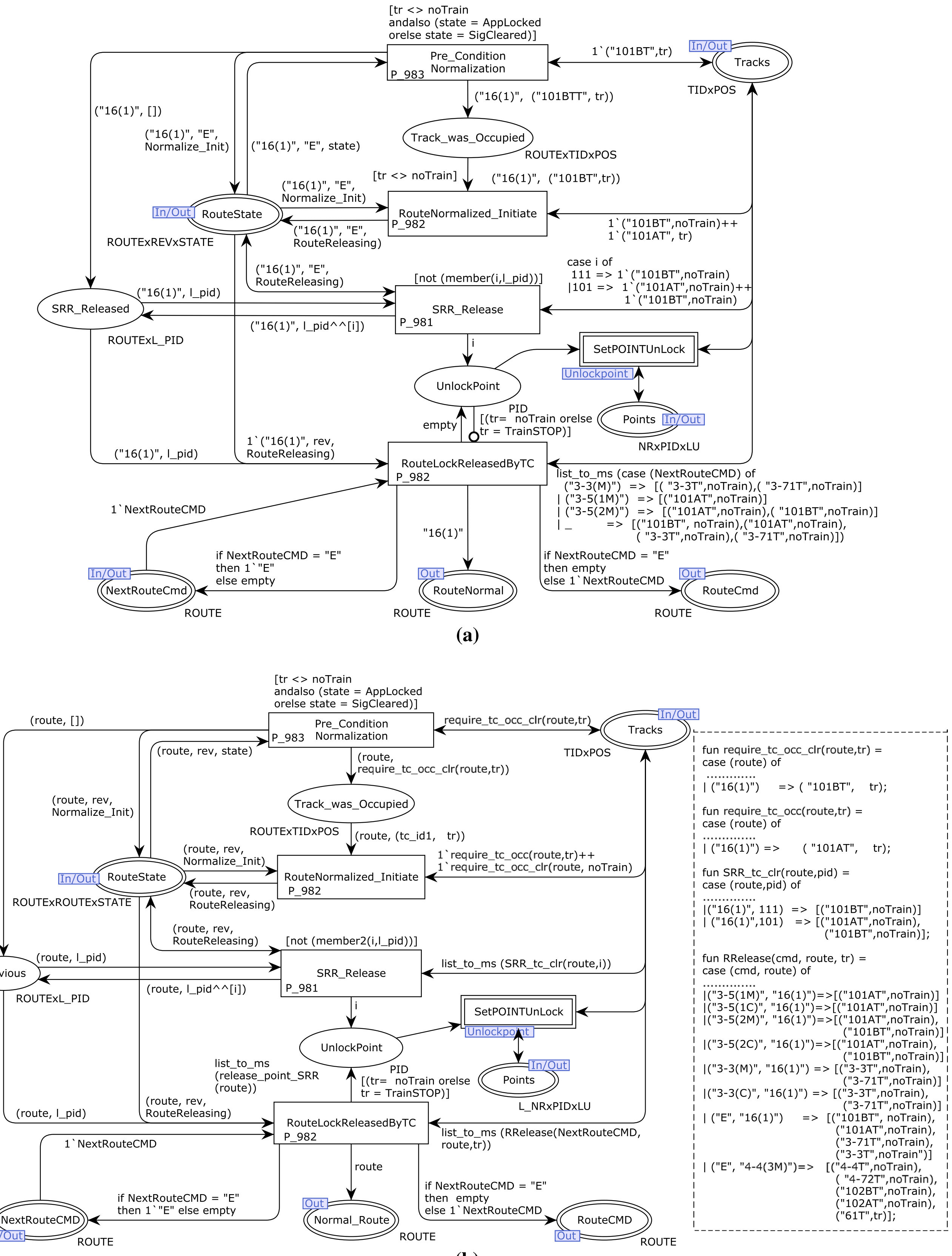


Fig. 12 **a** An unfolded CPN page: BackLocked. **b** The folded CPN page: BackLocked

sponding to the interlocking table in Fig. 4b. The two-stage condition to initiate route normalization of 16(1), (Track 101BT occupied and cleared) and (Track 101AT occupied), is modelled by the two transitions on the top of Fig. 13a. While the train is passing each point and the associated track circuit is unoccupied, transitions `SRR_Release` and `SetPOINTUnLock` release and unlock each point one by one. The last transition executes the quick route release (if any) and restores route 16(1) to the *Normal* state. Similarly, Fig. 13b illustrates the folded version of `RouteReleased` page. Every route shares the same net structure by instantiating the value of “route" into the ML



```
fun require_tc_occ_clr(route,tr) =
case (route) of
 .............
| ("16(1)")     => ( "101BT",    tr);

fun require_tc_occ(route,tr) =
case (route) of
..............
| ("16(1)") =>      ( "101AT",    tr);

fun SRR_tc_clr(route,pid) =
case (route,pid) of
..............
|("16(1)", 111)  =>  [("101BT",noTrain)]
| ("16(1)",101)   => [("101AT",noTrain),
                              ("101BT",noTrain)];

fun RRelease(cmd, route, tr) =
case (cmd, route) of
..............
|("3-5(1M)", "16(1)")=>[("101AT",noTrain)]
|("3-5(1C)", "16(1)")=>[("101AT",noTrain)]
|("3-5(2M)", "16(1)")=>[("101AT",noTrain),
                              ("101BT",noTrain)]
|("3-5(2C)", "16(1)")=>[("101AT",noTrain),
                              ("101BT",noTrain)]
|("3-3(M)", "16(1)") => [("3-3T",noTrain),
                              ("3-71T",noTrain)]
|("3-3(C)", "16(1)") => [("3-3T",noTrain),
                              ("3-71T",noTrain)]
| ("E", "16(1)")     =>  [("101BT", noTrain),
                            ("101AT",noTrain),
                            ("3-71T",noTrain),
                            ("3-3T",noTrain")]
| ("E", "4-4(3M)")=>   [("4-4T",noTrain),
                            ( "4-72T",noTrain),
                            ("102BT",noTrain),
                            ("102AT",noTrain),
                            ("61T",tr)];
```

Fig. 13 **a** An unfolded CPN page: RouteReleased. **b** The folded CPN page: RouteReleased

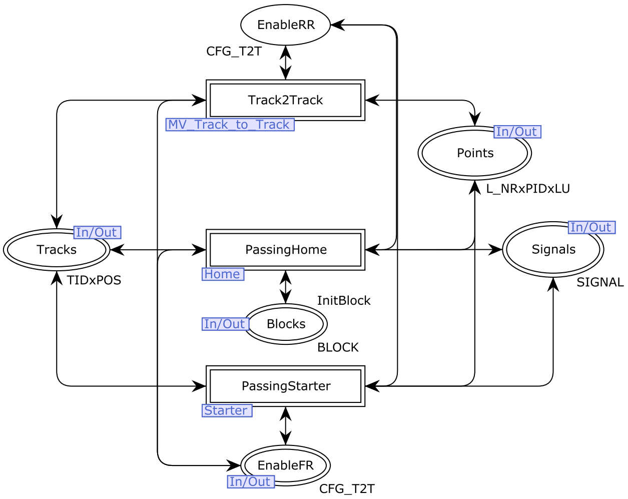


Fig. 14 The CPN page: TrainMovement

functions. Function SRR_tc_clr(route, pid) contains the sectional route release conditions listed in the sixth and seventh columns of Fig. 4b. The conditions of quick route release listed in the last two columns of (Fig. 4b) are stored in function RRelease(cmd,route,tr). When the variable cmd is “E", RRelease(cmd,route,tr) is reduced to the typical route release.

3.7 Train movements

Figure 5 does not only show the hierarchy structure of the interlocking but also of the train movement. Substitution transition MovingTrain in Fig. 7 is linked to the second level CPN page named TrainMovement shown in Fig. 14. After experimenting various styles and structures of the CPN model, we have divided the CPN model of the train movements between two adjacent track circuits into three groups: no signal; passing a home signal; and passing a starter signal. Each group is further divided into three kinds of movement: forward or facing the front of the signal; backward or facing the back of the signal; and train movement using call-on.

PassingHome Substitution transition PassingHome in Fig. 14 is linked to the third level CPN page named Home shown in Fig. 15. It comprises three substitution transitions and one executable transition. Place CONFIG_T2T stores geographical information, order pairs of two adjacent track circuits that have a home signal between them. Transition TrainSTOP represents an event when the train cannot proceed further and stops in front of the home signal. Because we do not model the time, occupying track for 60 s is modelled by transition TrainSTOP. Substitution transition BWD represents the train passing from the back of the home signal so that it is simply the movement from track to track in the backward direction. Substitution transition CallOn represents the event consisting in the train passing the home signal using the call-on signal. The call-on signal will be turned off when the last wheels are cleared from the berth track (in front of the home signal). In case of using call-on, the block normalization shall be handled manually.

Substitution transition FWD represents the event consisting in the train passing the home signal using the main aspect (yellow or green). It is linked to the fourth level CPN page named MV_HOME shown in Fig. 16. MV_HOME comprises two executable transitions: FWD_FRONT and FWD_REAR. Transition FWD_FRONT in Fig. 16 represents the front wheels occupying the track ahead of the train. After FWD_FRONT is executed, the train spans over two tracks. Place Accident is used to detect any train collision. Transition FWD_REAR in Fig. 16 represents the last wheels moving out of the track behind the train. After FWD_REAR is executed, the train spans over one track. Because train movements are folded and use the same CPN pages, each token in place EnableFR and EnableRR is used to enable and trace each train movement.

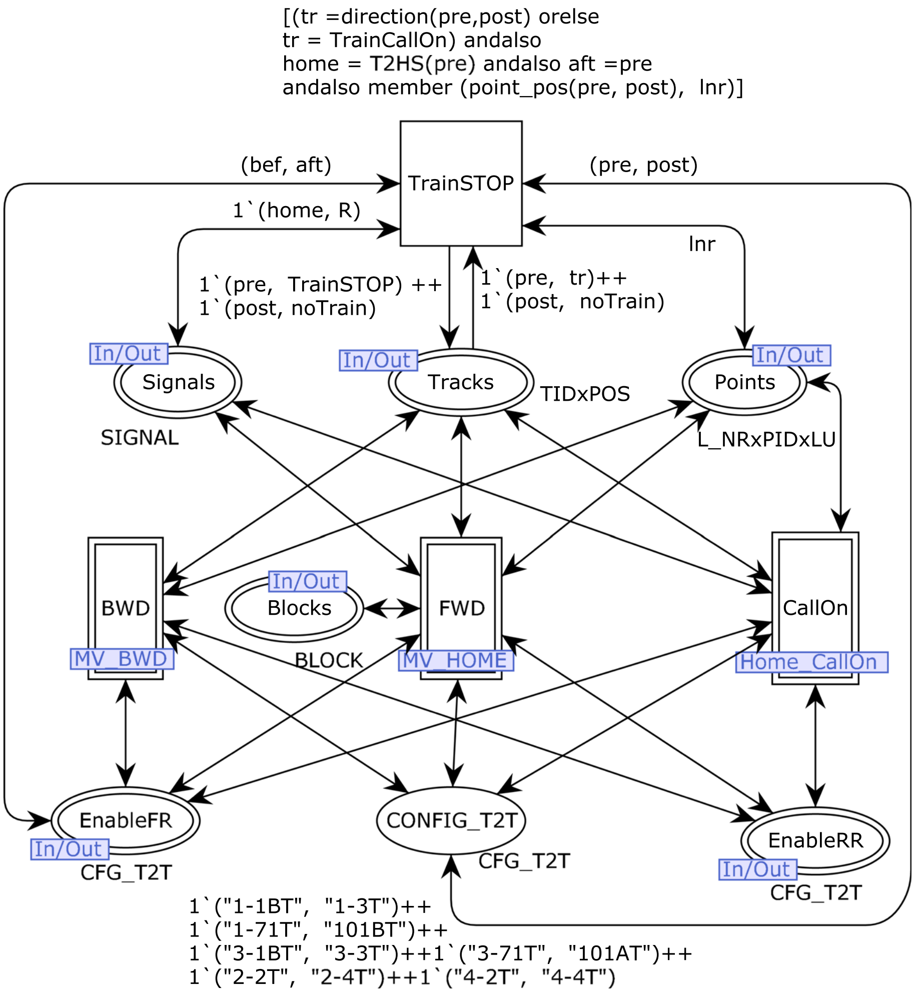
Enabling transition FWD_FRONT requires the home signal displaying the yellow or green aspect. After transition FWD_FRONT is executed, the associated signals (both home and warner) are normalized accordingly. The execution of transition FWD_REAR also normalizes the associated block equipment.

3.8 Simulation and visualization extension

Since [18] we have attempted to mimic the train movements along track circuits and signal aspects using places and transitions. The major problem is that for checking the conditions (e.g. route setting) we need all apparatus information of each kind contained in one place. Thus, we have duplicated information: one in the centralized place and the other in distributed places along the track layout. The duplication made modelling very inconvenient. Mimicking the train movement is one of the most important requirements from our counterparts. To comply with this requirement, we adopt an extension of CPN Tools, visualization extension (VE). The VE code itself is written in JAVA, but the VE is called via ML functions in the code segments of the transitions. This graphical visualization is useful when we conduct simulation for testing desired properties.

Examples of VE functions are illustrated in the code segment of the transitions in Fig. 16. The input variables are a track circuit pair (pre, post), home, warner and GYR2. Red_Track(pre, post) changes the colour of both tracks to red (occupied). Red_Home(home, Y, warner, GYR2) changes the home aspect to red and the warner aspect to yellow or red depending on GYR2. These actions occur when the front wheels occupy the track circuit behind the home signal. Grey_Track(pre) changes the colour of the track circuit

Fig. 15 The CPN page: `Home`



in front of the home signal to grey (unoccupied). Yellow_Normal(post) changes the colour of associated block indicator “N" to yellow. These actions happen when the last wheels clear the track circuit in front of the home signal. Examples of graphic visualization are displayed in Figs. 1a and 4a.

4 Lessons learnt and perspective

4.1 Route interlocking validation

Our previous work [18–20] did not successfully validate the “Route Interlocking” property because we tried to set any combination of non-conflicting routes at the same time. Of course, this leads to state explosion. However, [4] suggested that to prove “global no-collision” properties it is enough to prove only “no two-train collision”.

Thus, we test the route interlocking by adding the CPN model of Fig. 17 to the `Top_Level` page and adding an ML function call to conduct automatic route setting [20]. The stop option in the state space tool in CPN Tools is set such that detecting a train collision will stop the state space generation. The analysis result gives the size of the state space as 29,049 states and 53,853 transitions. It takes 2 h 15 min and 18 s to generate the state space. No train collision is detected.

4.2 Formal specification

With reference to the discussion in Sect. 3, the interlocking table well presents the specification of route interlocking, but the specification of the other properties are incomprehensible. Contents in the interlocking table are usually very short abbreviations and scattered across various pages. Different railways have different interlocking tables so that even the signalling experts sometimes have trouble with the tables. With pattern matching and a graphical notation, the Petri Net formalism is a natural choice for formalising interlocking tables. With the abstract data type and hierarchical structure of Coloured Petri Nets, we can fold CPN diagrams of various routes into a single CPN diagram, hence producing a much more compact and generic model. Although the content in the tables is hidden inside the ML functions, the signal

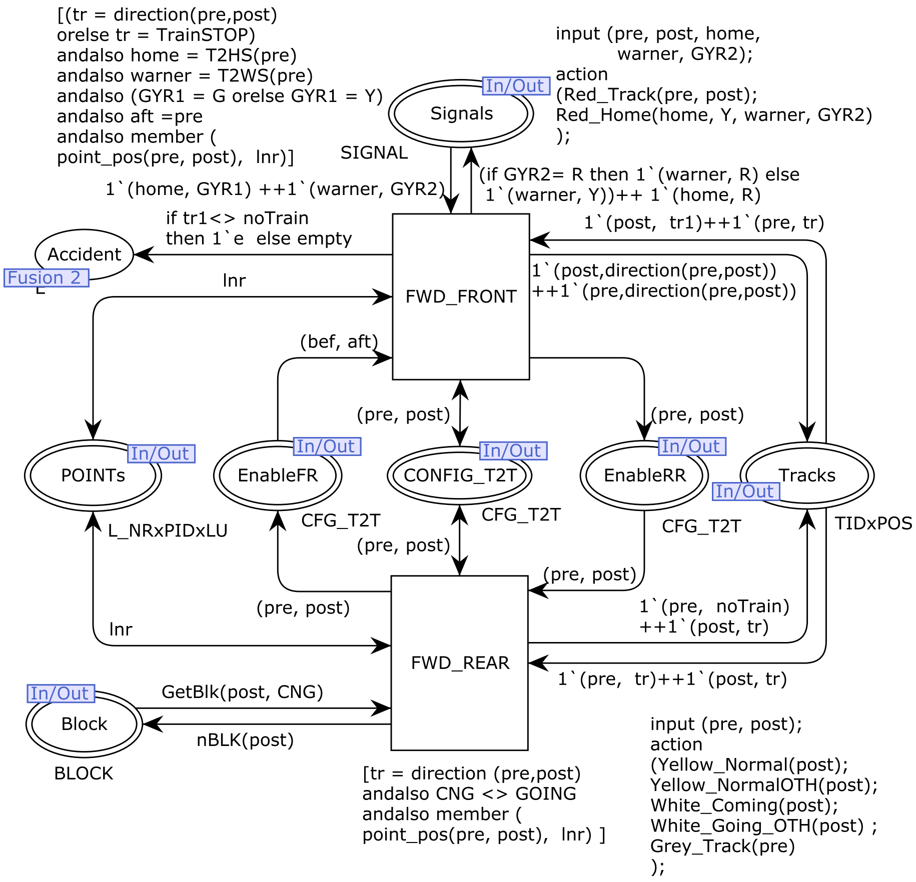


Fig. 16 The CPN page: `MV_HOME`

engineers can view the content via multiple windows on the computer. By changing the content of ML functions in arc and guard inscriptions according to the content in the interlocking table, our CPN diagram can be reused with other interlockings. It can scale up to reuse with a larger interlocking as long as no additional properties not already in the model are required. As railway networks become larger, passengers demand shorter delays, and the railway operators are asking for more efficiency and flexibility, interlocking tables have progressively become too complicated to comprehend. This argument can be witnessed by the tables of Figs. 1b, 3 and 4b. To cope with the complexity, we may need other formal forms of representation rather than the interlocking tables. Coloured Petri Nets are a promising formal specification for the railway signalling systems.

4.3 Modelling with prioritized transitions

Normally the interlocking controller works faster than train movements. To make the model compact, we assign train movements a lower priority than interlocking controllers, using prioritized transitions, hence avoiding additional net structure required to implement the prioritization otherwise. Moreover, we also adjust the order of precedence of each transition to make the model compact and work properly. It seems to be very useful but we discovered two drawbacks. Firstly, a slightly different order of precedence causes the model to behave differently. Secondly, the prioritized transitions cause a lower speed of state space generation (and simulation). Because of its drawbacks, attempts are made to revise the CPN model without prioritized transitions but using other constructions such as timed tokens and fusion places instead. We discover that regardless of the state space reduction, the use of prioritized transitions provide us a better way to construct the CPN model. Without prioritized transitions, it is very difficult to have the revised model worked out correctly.

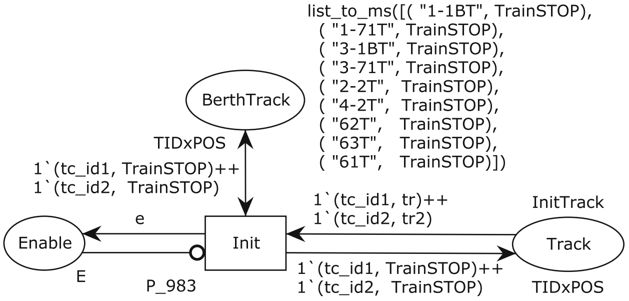


Fig. 17 Additional CPN diagram to the `Top_Level` page

5 Conclusion and suggested work

This paper presents the complete CPN model of a typical railway station of Thai railways. We illustrate how well our CPN diagrams capture all nine desired properties from the interlocking table. Route locking and flank protection properties are formally validated using state space analysis. Other properties have been simulated and visualized using a visualization extension. The paper focuses on the folded version of the CPN model. Our counterparts use this version as the example applied to other stations in order to gain insights and train new engineers. In future we have planned to analyse our model using sweep-line technique [15] and study its scalability. To assist the users and facilitate the work, we envisage the need of specific integrated development environment (IDE). The State Railway of Thailand starts to adopt the technology such as Balise, Automatic Train Stop (ATP), Cab Signalling. Of course, this will affect SRT signalling principles as well as our CPN model.

Acknowledgements This work is supported by Research Grant from the National Science and Technology Development Agency and National Research Council of Thailand. The author is thankful to anonymous reviewers and Dr. Guy E. Gallasch. Their constructive feedback has helped the author improve the quality of this paper. The author is also grateful to his colleagues from the State Railway of Thailand, Patama Sridaranop, Anan Phonimdang, Sommart Klinsukol, Ittipon Kansiri, Kittisak Siripen, Surapol Eiamsaart and Navaporn Rittisuk.

References

1. Basten, T., Bol, R., Voorhoeve, M.: Simulating and analyzing railway interlockings in exspec. IEEE Parallel Distrib. Technol. Syst. Appl. **3**(3), 50–62 (1995)
2. CPN Tools home page. http://cpntools.org
3. Clavel, M. et al.: Maude Manual. http://maude.cs.uiuc.edu/
4. Fantechi, A.: Distributing the challenge of model checking interlocking control tables. In: Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies—5th International Symposium, ISoLA 2012, Heraklion, Crete, Greece, 15–18 Oct 2012, Proceedings, Part II, pp. 276–289 (2012)
5. Fantechi, A.: Twenty-five years of formal methods and railways: what next? In: Software Engineering and Formal Methods—SEFM 2013 Collocated Workshops: BEAT2, WS-FMDS, FM-RAIL-Bok, MoKMaSD, and OpenCert, Madrid, Spain, 23–24 Sept 2013, Revised Selected Papers, pp. 167–183 (2013)
6. Fantechi, A., Fokkink, W., Morzenti, A.: Some trends in formal methods applications to railway signaling. In: Gnesi, S., Margaria, T. (eds.) Formal Methods for Industrial Critical Systems: A Survey of Applications, pp. 61–84. John Wiley & Sons (2012). ISBN:9780470876183 (print). ISBN:9781118459898 (online). https://doi.org/10.1002/9781118459898
7. Ferrari, A., Magnani, G., Grasso, D., Fantechi, A.: Model checking interlocking control tables. In: FORMS/FORMAT 2010—Formal Methods for Automation and Safety in Railway and Automotive Systems, 8th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems, Braunschweig, Germany, 2–3 Dec 2010, pp. 107–115 (2010)
8. Hagalisletto, A.M., Bjørk, J., Yu, I.C., Enger, P.: Constructing and refining large-scale railway models represented by petri nets. IEEE Trans Syst Man Cybern Part C **37**(4), 444–460 (2007)
9. Haxthausen, A., Peleska, J., Pinger, R.: Applied bounded model checking for interlocking system designs. In: Software Engineering and Formal Methods—SEFM 2013 Collocated Workshops: BEAT2, WS-FMDS, FM-RAIL-Bok, MoKMaSD, and OpenCert, Madrid, Spain, 23–24 Sept 2013, Revised Selected Papers, pp. 205–220 (2013)
10. Haxthausen, A.E., Nguyen, H.N., Roggenbach, M.: Comparing formal verification approaches of interlocking systems. In: Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification—First International Conference, RSSRail 2016, Paris, France, 28–30 June 2016, Proceedings, pp. 160–177 (2016)
11. Hong, L.V., Haxthausen, A.E., Peleska, J.: Formal modeling and verification of interlocking systems featuring sequential release. In: Formal Techniques for Safety-Critical Systems—Third International Workshop, FTSCS 2014, Luxembourg, 6–7 Nov 2014. Revised Selected Papers, pp. 223–238 (2014)
12. James, P., Moller, F., Nguyen, H.N., Roggenbach, M., Schneider, S.A., Treharne, H.: On modelling and verifying railway interlockings: tracking train lengths. Sci. Comput. Program. **96**, 315–336 (2014)
13. James, P., Moller, F., Nguyen, H.N., Roggenbach, M., Schneider, S.A., Treharne, H.: Techniques for modelling and verifying railway interlockings. STTT **16**(6), 685–711 (2014)
14. Jensen, K., Kristensen, L.M.: Colored petri nets: a graphical language for formal modeling and validation of concurrent systems. Commun. ACM **58**(6), 61–70 (2015)
15. Jensen, K., Kristensen, L.M., Mailund, T.: The sweep-line state space exploration method. Theor. Comput. Sci. **429**, 169–179 (2012)
16. Jensen, K., Kristensen, L.M.: Coloured Petri Nets: Modelling and Validation of Concurrent Systems. Springer, Heidelberg (2009)
17. Sun, P.: Model based system engineering for safety of railway critical systems. PhD thesis, Université Lille Nord de France, France, July (2015)
18. Vanit-Anunchai, S.: Verification of railway interlocking tables using coloured petri nets. In: The Tenth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools, DAIMI PB 590, pp. 139–158. Department of Computer Science, University of Aarhus (2009)
19. Vanit-Anunchai, S.: Modelling railway interlocking table using coloured petri nets. In: Clarke, D., Agha, G. (eds.) Proceedings of the 12th International Conference on Coordination Models and Languages, (Coordination 2010). Lecture Notes in Computer Science, vol. 6116, pp. 137–151. Springer, Heidelberg (2010)
20. Vanit-Anunchai, S.: Experience using coloured petri nets to model railway interlocking tables. In: Proceedings 2nd French Singaporean Workshop on Formal Methods and Applications, FSFMA 2014, Singapore, 13 May 2014, pp. 17–28 (2014)
21. Vanit-Anunchai, S.: Application of coloured petri nets in modelling and simulating a railway signalling system. In: Critical Systems: Formal Methods and Automated Verification—Joint 21st International Workshop on Formal Methods for Industrial Critical Systems and 16th International Workshop on Automated Verification of Critical Systems, FMICS-AVoCS 2016, Pisa, Italy, 26–28 Sept 2016, Proceedings, pp. 214–230 (2016)
22. Winter, K.: Optimising ordering strategies for symbolic model checking of railway interlockings. In: Leveraging Applications of Formal Methods, Verification and Validation. Applications and Case Studies—5th International Symposium, ISoLA 2012, Heraklion, Crete, Greece, 15–18 Oct 2012, Proceedings, Part II, pp. 246–260 (2012)