Chapter 7
Formal Validation of Interlocking Under Signaling Rules

Pengfei Sun, Simon Collart-Dutilleul, and Philippe Bon

7.1 Introduction

The railway principles and standards used to be validated at the national level where each country has its own "language " for railway and its own requirements for managing trains on its network. Now, to promote the European rail market for passengers and freight, European Union has provided a solution called the ERTMS (European Railway Traffic Management System) that aims to create a common, harmonized, and standardized management of rail traffic and signaling in Europe in order to have a seamless network at the European level.

This brand new standard is easier to apply in the new lines, where wayside signaling cost is kept to a minimum, but all the vehicle fleets that operate on these lines must be equipped with the ERTMS on-board system. However, for the existing lines, there is an alternative "Mixed operation " solution. This is a strategy where the wayside signaling is equipped with both ERTMS and conventional systems. Normally, the conventional one is the legacy line used during the upgrade program. The main reasons for applying such a mixed solution are: financial and organizational constraints make it impossible to install ERTMS in the whole network in a short time. In addition, not every train has to go across the border line, and ERTMS-equipped trains sometimes have to run on the conventional lines. Most national companies prefer to gradually deploy the ERTMS in order to replace the conventional systems with a unified European system.

P. Sun (✉)
Southwest Jiaotong University Chengdu, Chengdu, China
e-mail: pengfeisun@home.swjtu.edu.cn

S. Collart-Dutilleul · P. Bon
COSYS/ESTAS, Université Gustave Eiffel, Villeneuve d'Ascq, France
e-mail: simon.collart-dutilleul@univ-eiffel.fr; philippe.bon@univ-eiffel.fr

Every conventional signaling system is the result of historical evolution, which was boosted by progressively technological development and lessons of accidents. Generally, its safety is ensured by engineering experiences, rather than by systematic methodology and their evaluations. So far, there has not been a lot of engineering experiences of ERTMS, which means it is impossible to evaluate the new system in the traditional way. Meanwhile, the management of railway signaling in ERTMS is based on the local rules pertaining to each country and not on global ones, which makes it difficult to evaluate the combined system in terms of safety. However, as a signaling system, its most important responsibility is to maintain transportation safety. Therefore, any implementations before being put into use should have detailed verification and validation (V&V), especially the compatibility of ERTMS and conventional signaling standards and systems.

One of the basic requirements of the railway safety is that a system must prevent trains from collision. For this reason, there is a mechanism, called railway interlocking system (RIS), which is a collection of associated devices, complying with explicit signaling principles. The purpose of the RIS is to maintain the transit safety by connecting and arranging the points and signals, so that a hazardous condition cannot arise. The specification and analysis of the RIS is an important part of the deployment of ERTMS. The evaluation of their global consistency is needed, which concerns the consistency between the conventional system and the ERTMS-equipped system, with regard to safety. This issue is crucial, and yet it has scarcely been covered by scientific literature. In fact, one of the difficulties of this problem comes from the lack of formal representations of both systems that could enable the validation of different aspects through test scenarios. So some new methodologies that are more systematic and formal need to be adopted.

In order to maintain high-level safety with deterministic scope, a project, called “PERFECT,” was launched to develop the safety specification and verification of French railway interlocking systems in the context of national rules and the influence of implementing ERTMS laws on the original systems (Bon et al. 2013; Collart-Dutilleul et al. 2014; Sun et al. 2014). This chapter will introduce the low-level part and the fundamental phase of the project. It focuses on the formal validation approach of the French railway RIS based on the computer-controlled relay-based system. This study aims to provide a methodology for a comprehensive assessment of the consistency of the following two aspects: the operating rules of local signaling systems and the additional safety requirements (which means the ERTMS).

With this methodology, we are able to follow the safety analysis: the safety assessment of new systems, the analysis of given scenarios, and the evaluation of safety requirements of system updates. After this method is recognized by railway experts, we will develop the method in an automatic methodological tool easily applied in practice. Then, we will provide a methodology for translations between the exclusive model train and classical Petri net model. This will allow us the opportunity to apply our research results in actual practice.

7.2 State of Art

Nowadays, the design of railway systems increasingly benefits from advances in computer science, information technology, mathematics, and other engineering disciplines. Most of the railway devices are computer-related devices, which means either of these systems includes some software or is controlled by software. But software is notorious for having unpredictable bugs that may threaten its correct functioning. With the rising complexity, for a system that is composed of multiple computing elements, it is unfeasible to demonstrate the safety of a collection of behaviors with traditional safety assessments. "The employment of very stable technology and the quest for the highest possible guarantees have been key aspects in the adoption of computer-controlled equipment in railway applications" (Fantechi 2012; Fantechi et al. 2012). Therefore, the development and the implementation of formal proof and verification of system safety have been seen as a necessity for the railway domain.

So far, the railway signaling-related domain has been considered the most suitable and the most fruitful areas for formal methods (Fantechi et al. 2012). It is because railway signaling is safety critical. It has discrete nature and absence of hard real-time need. The broad use of FMs in this field has already been witnessed by over 182 references in an early review (Bjørner 2003). Some recent surveys and reviews (Bacherini et al. 2006; Fantechi 2012, 2014; Fantechi et al. 2014, 2012) focus on the advances in both formal method approaches and railway signaling applications. Still, lots of related work that has been performed by railway companies are not published because of confidentiality considerations.

In this chapter, our candidate is colored Petri net (CPN), a graphical modelling language, whose basic concept, Petri net, is first introduced by C. A. Petri (1966). The basic Petri net (or element Petri net) has the advantage of expressing discrete event control systems, and studies of Petri nets in railway can be traced back almost 20 years ago. However, the descriptive ability of basic Petri net seems not to meet the needs of complex systems. Many derivatives of Petri net have been introduced in this research area, such as colored Petri net (Jensen 1981, 1987).

With the help of such high-level Petri net, there comes a large-scale application—Oslo Subway (Bjørk 2006; Hagalisletto et al. 2007; Moen and Yu 2004; Yu 2004)—that integrates CPNs into the system development to simulate the Oslo subway and analyze schedules of trains. This project developed a specification tool for specifying and automatically constructing large CPN models of railroads. One of the important project experiences shows that CPN is a good specification language for communication because the research group collaborated with chief engineers from railroad infrastructure and traffic department. Although none were specialists in Petri nets nor formal methods, they understood the models and were able to provide suggestions for improving the system.

The specification, analysis, and implementation of railway control logic are always a hot research topic. In work, Fanti et al. (2006), Giua and Seatzu (2008) discuss the control of the railway network using CPNs. A resource-oriented CPN method is introduced in Wu and Zhou (2004), which could deal with the deadlock of automated guided vehicle (AGV) systems. Cheng and Yang (2009) use a fuzzy Petri net for railway traffic control. A similar solution can be found in Kaakai et al. (2007) using a hybrid Petri net.

The level crossing (LC) is also a critical crux in both road and rail infrastructures. Stochastic Petri nets are applied in Ghazel (2009), Huang et al. (2010) in order to precisely reflect the system's dynamics. Furthermore, stochastic Petri nets could be used to evaluate the real-time system in railways, such as data processing (Zimmermann and Hommel 2003) and device-to-device communication (Lei et al. 2013).

Besides ETCS, there is another advanced train control system, called "communication-based train control (CBTC)," which has been applied to many metros. Its protocols and services have been studied by CPN (Chen et al. 2007; Xu and Tang 2007), deterministic and stochastic Petri net (Zhu et al. 2012), and timed Petri net (Wang and Bai 2010).

In France's railway domain, the French National Railway Company (SNCF) has initiated and participated in many projects. One of the most successful projects is to develop a formal validation method and tools for new computerized RISs and existing RISs (Antoni 2009a,b,c; Antoni and Ammad 2007, 2008). This project is led by Marc Antoni, the head of Innovation and technologic pole of SNCF Infra and director of the Rail System Department of UIC. This study developed four successive DSL tools (Antoni 2012b):

1. Tools A: general way for the definition of safety properties
2. Tools B: generation of the safety properties file
3. Tools C: proving tool: formal validation tool
4. Tools D: reached system state tree and execution certificate

In Tools A and B, the safety properties are specified with interpretable deterministic Petri nets, which will be later interpreted in the target machine. This method has been accepted by SNCF Infra. Now it has been applied in real RIS of "Noisy le Roy," situated next to Paris, and also applied in a new double-track level crossing. It is said that this method will be used by UIC and will be applied in the German system (Antoni 2012a).

Moreover, in order to verify the high-level systems' safety requirements, SNCF has made some performance assessments for both local signaling rules and European signaling standards, by specification and analysis of CPNs (Buchheit et al. 2011; Lalouette et al. 2010; Gregory et al. 2010).

7.3 Preliminary of Railway Safety and Interlocking System

This chapter aims to describe and to formalize some major safety properties and the control logic of interlocking systems in French railway. The reader who is familiar with the background can skip this section.

7.3.1 Safety Management of French Railway System

The concept of safety has different explanations depending on the nature of the systems and activities. The safety of rail traffic is particularly based on "the possibility of stopping." Most of the signaling rules take this concept as the primary requirement. If no train is moving, there will not be any danger to the traffic itself. So the basic system state can be simplified as the diagram shown in Fig. 7.1. This concept of safety is also widely used in the train control procedures, such as the ATP system, which stops the train according to the radio-based signals, in order to avoid a collision.

Any signaling rules and signaling-related procedures require a full explanation of safety properties. In French, they are historically based on *determinism*. Every system state has one or more causes. If a state is undesirable, removing its causes should help to avoid it.

One commonly used method is reasoning, which is necessary to exploit for every state, and especially to ensure that an undesirable event does not take place. The deterministic reasoning can only be applied to a closed system; otherwise, there will be a risk of unforeseen system state. Thus, the principle of the organization of the external environment is to limit the number of interactions, in order to avoid introducing chaos. As the external environment is one of the foundations of safety design, only those directly related to safety should be considered. This technique has already been used for operating safety and technical safety in French railway for a long time and its result proved to be safe.

Nowadays, with the development of the computer, the computer-controlled equipment plays an important role in many industrial areas. It has some advantages such as:

- Handling of complex new functions
- Ability of long distance remote control
- Reduction in staff

But everything has its two sides. It also has disadvantages such as:

- Long development cycle and hard to modify safely once the produce is finished.
- Require qualified operating and maintenance staffs.

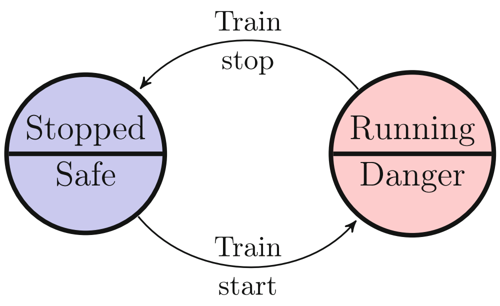


Fig. 7.1 Railway system state

- More difficult to validate and to integrate into the global system.
- The life cycle of computer devices is shorter than that of mechanical ones.

Unfortunately, many experiences show that the current development method cannot provide a safety guaranteed system according to SIL3 or SIL4. And the integration safety cannot be ensured under the global framework. A study has shown that "more than 3/4 accidents in relation with computerized systems are due to specification errors" (Antoni 2012a). Those accidents are caused by incorrect fiction descriptions, unthoughtful modifications, or improper maintenance.

In the traditional system, it was necessary to identify the failure events and to reduce their occurrence causes. When adopting the computer-controlled system, formal proof or verification is therefore regarded as a necessity. The following aspects should be taken into consideration.

- The functions and behaviors of such automated systems must be deterministic.
- Some properties should be specified rigorously:
  - Safety predicates
  - Functional predicates
  - Assumptions of interactions with the external environment
- For model checking-based formal proof, it is only possible when the reachable system states are finite.

The safety state of a computer-controlled signaling system is shown in Fig. 7.2. Transitions with red forbidden sign are the undesired system changes, which should

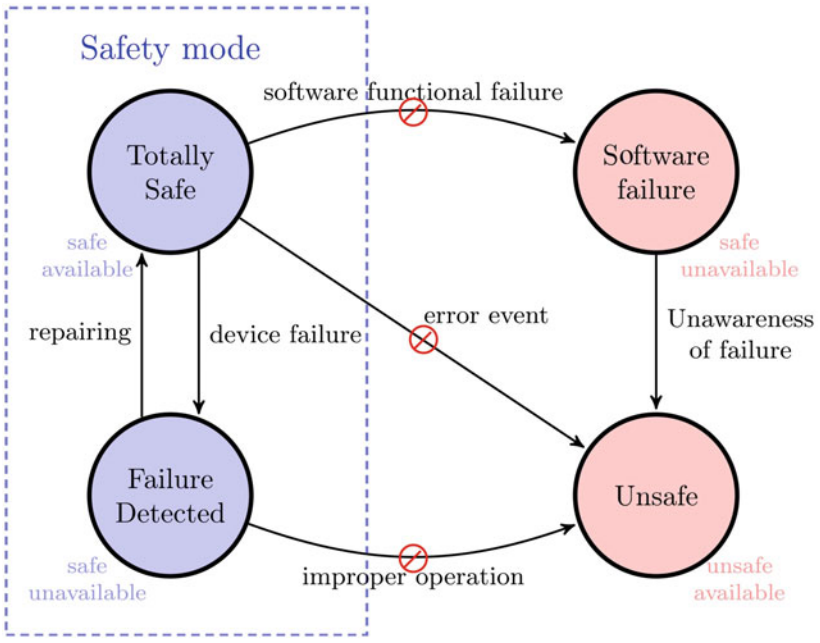


Fig. 7.2 The overall safety of a computer-controlled signaling system

be identified and reduced through formal specification & verification. In this way, the final system could operate as a "fail safe " system.

7.3.2 *French Railway Interlocking System*

One of the basic requirements of the railway safety is that a system must prevent trains from collision. For this reason, there is a mechanism, called railway interlocking system (RIS), which is a collection of associated devices, complying with explicit signaling principles. The purpose of the RIS is to maintain the transit safety by connecting and arranging the points and signals, so that a hazardous condition cannot arise.

There is a simple example of an interlocking system, as illustrated in Fig. 7.3. Track segments are represented in a topology structure, and all of them have track circuits that detect the occupation of a train. Joints of different track lines represent the points. The sign-board-like symbols are signals of various types of transition control. This example is constituted by 2 allowed routes, 1 point, 2 signals, and 3 track circuits. The interlocking route that a train can go through safely must meet the following requirements:

- All points are properly positioned and are locked.
- Conflicting routes must be protected.
- All the tracks along the route must be clear.

When all of the above conditions are satisfied, the signals can be set to green to let the train enter the route. These rules express the fundamental principles that

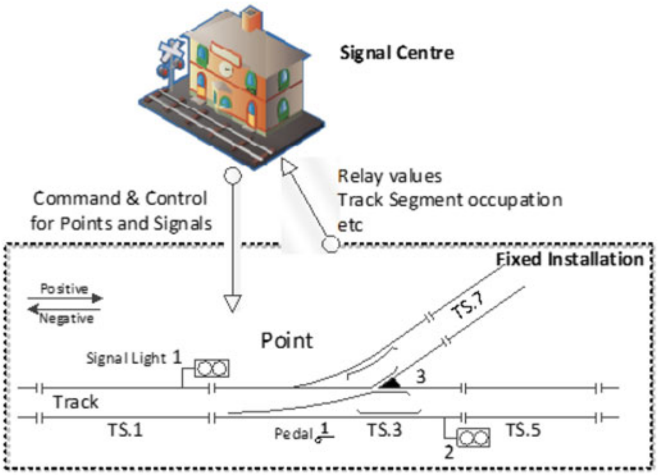


Fig. 7.3 An example of railway interlocking system

hold for all the RISs. Such rules ensure only the correct combinations of tracks, points, and signals, in order to avoid accidents. The signal indications authorize the movements of the train. They are handled by the interlocking system and can be considered as an indicator of the route establishment.

In our research, we restrict ourselves to the modelling of RISs. To better specify this complex system, we now introduce its composition and main components. In railway signaling, the term “interlocking” has two meanings (Pachl 2002). First, “an interlocking” is an arrangement of signal appliances that prevent conflicting movements through an arrangement of tracks. Second, principles to achieve a safety arrangement between signal appliances are also generally called “interlocking.”

According to the above definition and considering the train and the operator as external interactions, the RIS could be roughly divided into two parts: the signaling operations and the fixed installations.

Signaling operations are a set of operating rules and procedures that can maintain safety and high efficiency of transits. It comprises computer automatic controls and human control processes. Normally, the computer responds to most of the device-oriented operations, such as route establishment, route auto-destruction, $\cdots$, while human control deals with decision-making, such as route selection, route mode selection, route manual destruction, $\cdots$, and some non-regular operations, such as shunting operations.

Fixed installations are a set of components of geographical routes that include straight track sections, points, signal lights, and some ground-based automatic signaling devices that could work automatically and do not need supervision from the signaling center. Thus, they should be treated as a component in the geographical route.

7.4 Formal Modelling of Railway Interlocking System via HCPN

In this section, we will study the modelling of the French railway interlocking system using hierarchical colored Petri net.

First, we describe the modelling structure of an interlocking system and its corresponding network, as well as a set of interlocking properties that this network should obey. Subsequently, we specify this interlocking system with colored Petri nets in a generic and compact structure. In this modelling framework, the high-level functions of RIS are modelled in terms of a hierarchical and modular point of view. The railway layout (networks) is modelled in a geographical perspective, in order to be easily understood by railway expert engineers. Then, for the high-level parts of RIS, we propose a modelling pattern of the French railway interlocking system, which is a parameterized model that respects the French national rules. It is a general reusable solution to this kind of problem and can be used in many different given contexts. Finally, for the low-level parts of RIS, we introduce an event-based concept into the modelling process, in order to better describe the internal interaction of low-level interlocking logic. In this process, a reduction policy is applied both before and after the state space calculation to obtain a new compact graph with the same reliability for analysis.

7.4.1 GRAFCET and Petri Net

Petri net is a formal, mathematical, well-developed theory. However, French industry still prefers to use another informal tool—GRAFCET. In order to be close to industry usage habits, and to take advantage of formal methods, we make a little comparison of GRAFCET and Petri nets to discuss why the Petri net is our best solution for modelling the French railway system.

GRAphe Fonctionnel de Commande Étape/Transition (GRAFCET) is a method of representation and analysis of automation. This is a graphical tool for describing the behaviors of the control processes. It describes the informational interactions across the system boundary. This mode of representation is independent of the technology used in the automation and reflects a consistent specification of the automatism.

This method was proposed in 1977 by the Association Français pour la Cybernetique Economique et Technique (AFCET) as a standard to represent specifications for software control systems. It was accepted in 1982 as a French standard. Latter in 1987, it was accepted as an international standard IEC 1131.3 by the International Electrotechnical Commission. The GRAFCET is also known as DFS (Diagramme Fonctionnel en Séquence) or in English, the SFC (Sequential Function Chart).

The GRAFCET has many advantages, and it already has a wide range of applications. However, with the increasing safety need of the international standards, GRAFCET has also long been criticized because of its lack of a formal foundation that allows it to ensure correctness and safety requirements. On the other hand, "it lacks adequate methodology that allows an efficient development of high quality models in the case of complex systems on the other" (Zaytoon and Villermain-Lecolier 1999).

To compensate for its deficiency, researchers began to use other formal languages to describe GRAFCET. Particularly, formal design methods of state diagrams and Petri nets are available. State diagrams are easy to learn and can be converted into many existing programming languages of GRAFCET without any problem. However, some complex structures, such as a parallel, cannot be well represented. Petri nets can achieve almost all the structures of GRAFCET (René and Alla 1992, 1997). The models can be extensively analyzed by PNs in order to prove formally. Also, the model of PNs can be converted into GRAFCET. Furthermore, PNs are also accepted by some French industries. Here is a comparison of the structure of GRAFCET and PN in Table 7.1.

Based on Table 7.1, we can easily transform a GRAFCET model into a PN model as shown in Fig. 7.4. Their notation formalism is so close that the engineer who is

Table 7.1 Structure comparison between GRAFCET and Petri net

GRAFCET		Petri net
Step	⇔	Place
Transition	⇔	Transition
Link	⇔	Arc
Receptivity	⇔	Guard
Action	⇔	Auxiliary place

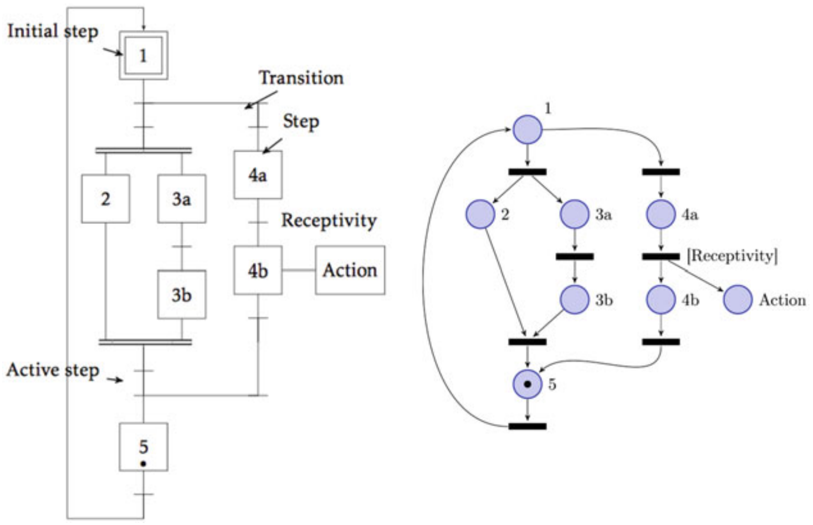


Fig. 7.4 Model comparison between GRAFCET and Petri net

familiar with the GRAFCET will easily understand the models of PNs. In other words, if a system is specified by PNs, it can be validated both by PN tools and by experienced expert engineers. In this way, the designed system can be considered as a strong formal proof. As a result, the PN is the most appropriate formal language to continue our research. Currently, the PNs are accepted by some French industries, such as the French National Railway Company (SNCF) (Antoni 2012b; Buchheit et al. 2011; Lalouette et al. 2010).

The major difference between these two languages is the mechanism of "concurrency." The GRAFCET allows all the enabled concurrent transitions to be fired at the same time, while in PNs, there will be a "choice" of firing one or another transition, thus reaching different new markings. Further information on the differences between GRAFCET and PNs can be found in Giua and DiCesare (1993).

7.4.2 *Initial Colored Petri Net Specification of Railway Interlocking System*

In our research, we focus on the traffic safety aspect and suppose that all the fixed infrastructures are both reliable and robust. The only threat to safety comes from the imperfect signaling rules or the incompatible international standards.

The modelling framework of the whole railway interlocking system could be divided into three parts: the signaling operations, the fixed installations, and rolling stock, as in Fig. 7.5. The train driver communicates with the dispatcher and requests an interlocking route. Train movement is a series of interactions with fixed installations (such as stopping at red lights and actions on track circuit). In response to train requests, the signaling operations send certain commands to fixed installations (such as points and signal lights) according to its operating principles:

- Signaling operations is a set of operating rules and control procedures of an interlocking system. It comprises computer automatic control and human manual control. Normally, the computer processes are responsible for most of the device-oriented operations, while human dispatchers deal with decision-making and non-regular operations.
- Fixed installations include track segments, points, signal lights, and other automatic facilities that could be self-acting without the instruction from the train controlling center. Whereas the critical safety results are always represented in the fixed installations, the safety verification for all the fixed ones' function is needed.
- Rolling stock runs on interlocking routes and is supervised by both route conditions and operating instructions.

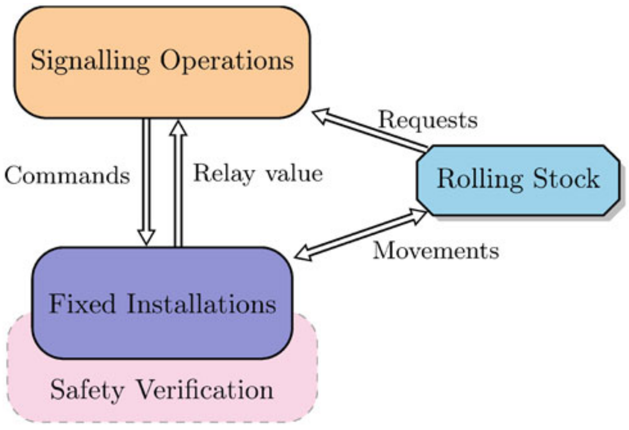


Fig. 7.5 Specification framework of railway interlocking system

Considering the large scale and the space complexity of interlocking systems, one feasible solution is to model the RIS by HCPN. The signaling operations and the fixed installations are represented by the topology structure of PN, in order to express complex connections and logical relations between different devices, while each train is defined as a colored token that can move along the network of track work.

To distinguish between various syntactic parts of a Petri net model, we classify different nets into 3 types. The first two basic types in RIS are the signaling operations and the fixed installations. The net $N^o = (T^o, P^o, A^o, \varepsilon^o)$ represents the operation part that implements the route management process and movement authority control. The net $N^i = \left(T^i, P^i, A^i, \varepsilon^i\right)$ represents the installation part where the train movements are realized by the transitions $t \in T^i$. The notation $N^s = (T^s, P^s, A^s, \varepsilon^s)$ denotes the supplemental part that is used to ensure the integrity of the model simulation and safety analysis. It could realize the initial simulation inputs or actions from the human operators, where $p \in P^s$ may be a compound place existing in other nets.

In order to standardize our modelling process, we have definitions below:

Definition 7.1 A basic unmarked RIS net is a connected Petri net

$$N_{RIS} = N^o \bigcup N^i \bigcup N^s,$$

where $N^o \cap N^i = P_{equip}$, N^o should not be an empty set $N^o \neq \varnothing_{pn}$ and N^i is strongly connected.

The common parts P_{equip} of the operation nets and the installation nets are signal equipment, such as signal lights and position of points. They perform the role of indicator in the operation nets and conduct the train movement in the installation nets.

7.4.3 A Geographical Approach of Railway Interlocking System

As a first approach, the RIS is specified into a CPN in a hierarchical and geographical perspective. This study can be found in our previous work (Sun et al. 2014). The basic hierarchy of the HCPN model framework is described in Fig. 7.6 The main net is the topmost net, which is the carrier of the whole model, "storing" all the subsystems and their interactions.

In the following subsections, we introduce the specifications of signaling part and installation part separately.

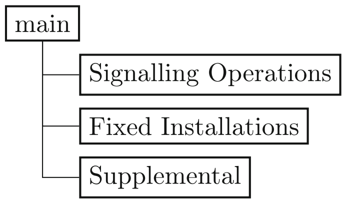


Fig. 7.6 Basic specification framework of railway interlocking system

7.4.3.1 Signaling Operation Specification

RIS signaling operations are a system with multi-input and multi-output. Their operating processes are involved with the functions in distributed levels. When modelling such a system, a specific model for system functionality seems not suitable for achieving the modelling objective. Successful experience in modelling the European Train Control System using CPN (Janhsen et al. 1997; Jansen et al. 1998) could give us some useful inspiration. In such systems, there are three aspects that should be integrated: *components*, *scenarios*, and *functions*.

When modelling the *component* view, the aim is to specify the communications and the interactions between different subsystems. A net of the component view shows a subsystem and its interfaces, and it could be further detailed in additional levels. The *scenario* view is the modelling of operational procedures. Its main elements are the sequence of events required to maintain operation, and the interactions between the signaling operations and the fixed installations. Individual scenarios are categorized into different groups, and in this way, they could be integrated into the corresponding component model. The *functions* represent the lower model level. They are involved in the process aspect and represent the activities or the response to interactions from the scenarios. Some of the functional modules can be used in different objects and the so-called functional blocks. These functional blocks are modelled as separate nets and can serve as functions in different scenarios, under the modelling principle of hierarchic decomposition. In this way, the subnets can be reused.

As the objective model framework needs to have so many features, an extensible framework is needed, which should also be readable, maintainable, and easy to accept by others. As a result, it should be modelled in a modular way. The hierarchical structure is the most consistent with the modelling requirements. It could integrate different functions of the system description and contain isolated modularity views in the model. Besides, their advantages are easy to comprehend and adapt and modular models can be reused. Meanwhile, the hierarchical ability of CPN provides a good basis for setting up the model in a straightforward way.

In order to structure the main component models of the signaling operations, a layered approach, proposed by Janhsen et al. (1997), Jansen et al. (1998), is adopted. Dynamics and functionality are expressed by both scenarios and functions. Scenarios show the behaviors of the system in its external environment, which means the railway operation context. Functions can process data received from external components or internal ones. The difference between scenarios and functions is that

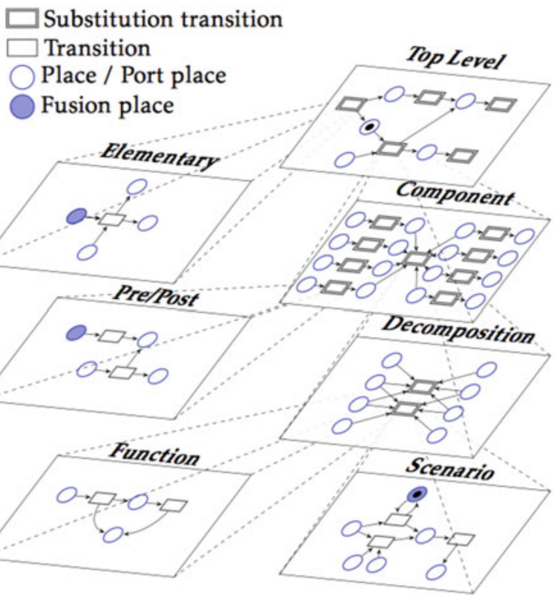


Fig. 7.7 Hierarchical model structure of signaling operation

functions are not subordinated to any scenarios. They are independent of scenarios and can be used within arbitrary scenarios.

Moreover, the concept of function in this thesis is not restricted to the very basis mathematical functions but can also represent procedures (may complex ones). To be more precise, each function represents a task. However, given the nature of their functionalities, we continue to use "functions" to refer to them. The corresponding vertical decomposition model is in Fig. 7.7 with several levels. The generic structure has a "Top Level" to store all the components. It shows the connections between components and their corresponding communications. The "Composition" layer shows the detail of the component models. The "Decomposition" layer represents the decomposition of the component model because some components are too complex to represent in one single model. The "Function" layers and the "Scenario" layers represent the function view and scenario view, respectively, and they may be further decomposed if necessary.

Moreover, for simulation purposes and compatibility reasons, two supplementary levels should be added into the hierarchical structure. The "Elementary" level is used to replace the preliminary transition of the top level. The "Pre/Post"-level concerns relations between components. They are used for preprocessing incoming messages and post-processing outgoing messages.

To illustrate how to map from signaling operation to the CPN model, here is a small demonstration of the route formation procedure, an important part of the interlocking operation. A complete process control always involves many aspects (see Page 392 in Rétiveau (1987)). As a demonstration example, only the core of the control flow will be presented. In Fig. 7.8a, there is the control flow chart. It

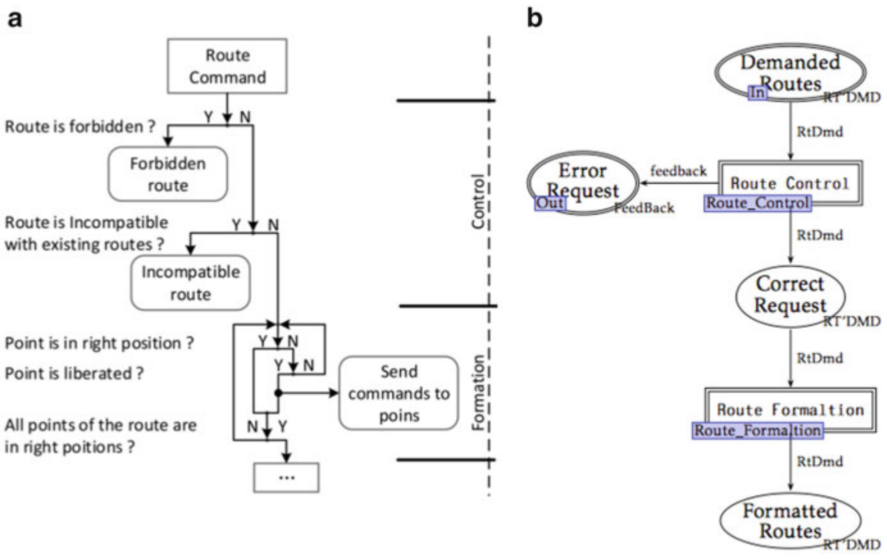


Fig. 7.8 Example of mapping signaling operations. (**a**) Route establishment flow chart. (**b**) Corresponding HCPN model

receives the route control instruction (route command) and checks whether this instruction is feasible and compatible with the existing ones. Then it will format the route according to its formation information, such as the positions of points.

The first 2 consecutive actions of interlocking route establishment are: control and formation. The control part validates the input of "Route control" instruction and acts as a filter. Only when the requested route is satisfied with current interlocking status, it is allowed to establish. Otherwise, an error message will be output and the process ends. In the French context, 2 aspects are checked concerning safety:

Forbidden route Inverse transit is forbidden and must be deactivated. When a track segment is acting as the destination of an established route, any new route originated from it is forbidden.

Incompatible route The region ahead of the signal must be free (in the case of a DA[1] route). This means that only when a route is partially destroyed because of the use of flexible transit 5, the corresponding initial signal can be used for another route. Otherwise, any new routes originating from the same signal are incompatible.

The formation part positions all the points of the commanded route. If the point is already in the expected position, no further action is performed. If the point

[1]Destruction automatic: A typical French interlocking route type that could be destructed by the passage of the train.

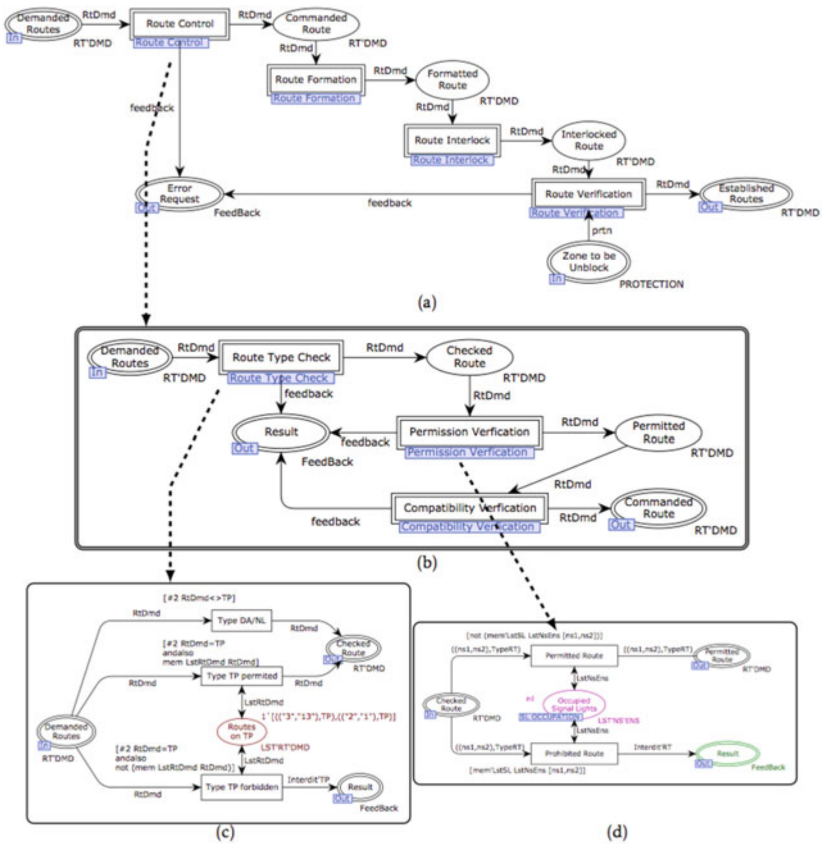


Fig. 7.9 Example of mapping signaling operations (2). (**a**) Composition net (route establishment). (**b**) Decomposition net (route control). (**c**) Scenario net (route type check). (**d**) Function net (permission verification)

is occupied by other routes, the process will wait until it is released. Only if a point is not in the right position and is liberated, an instruction will be sent to the fixed installation model to change the point. After receiving the new position, the procedure continues to confirm the next point of the route. When all the points are in the right position, this process is over.

The real corresponding model of the control flow is represented in Fig. 7.9.

Figure 7.9a represents a "Component level." It consists of hierarchical transitions for route control and route formation. The input place contains the token of route information. It could be passed through the model or output an error token. Figure 7.9b is the decomposition net of route command procedure. It still contains sequences of functions: route type check, permission verification, and compatibility verification. Figure 7.9c is a scenario net, because the place "Routes on TP" contains the configuration of a certain scenario. Figure 7.9d is a function net, because its function is independent of the scenarios.

It should be noted that in this hierarchical structure, only the scenario nets reflect the localization of the stations by their configurations (the initial tokens), while the other parts of the model are the specifications of national railway standards and do not vary with different stations. Once we have completed a model of the signaling operations, the models of other stations under the same national standards could easily be derived from the previous model by only changing the initial tokens in each scenario net.

7.4.3.2 Geographical Railroad Layout Specification

The normal solution of modelling the fixed installations is the geographical approach. This approach can be considered as distributing the knowledge of the interlocking rules to objects modelling the geographic placement of physical elements (Banci et al. 2004). Its geographical structure allows us to slice the whole railway layout into independent and distributed components that can be individually modelled and physically located next to their relevant units.

Normally, an RIS route layout is made up of multiple similar components: tracks, points, and track-side signals. A track segment is a section of straight track that contains a complete track circuit for occupation detection. It is a simple straight or Y-shape with a point. A point is a railroad switch enabling railway trains to be guided from one track to another. The direction of the point is controlled by the signaling system according to the route requests. Generally, an interlocking system is within a station yard, where trains are running at low speed, so train movements are partly directed by fixed signal lights installed along the rail. A signal light mainly uses two aspects: red (stop intermediately) and green (route clear).

Both track and point are referenced as atomic components, which could form the geo-graphical structure of the whole railway layout and compose the route for transit. These journeys are also properly controlled by signal components along the railway layout, so the signal light could be regarded as constraints for train movement.

Track Segments

Figure 7.10 shows a demo of PN of two successive track segments. Each place represents a track segment. Two transitions move train tokens between the two segments, depending on the direction of the train and supervision by the guard function of the transitions. The direction from left to right is referred to as the “odd” direction (impair in French system), and the opposite direction is called the “even” (pair in French system) direction.

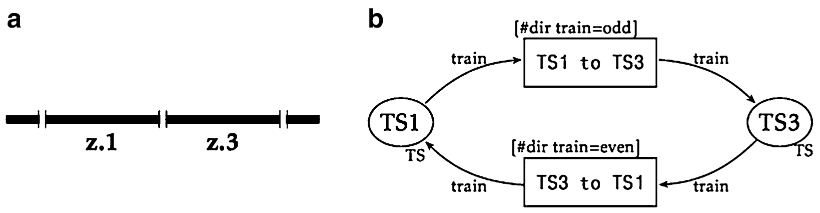


Fig. 7.10 A Petri net representation of track segments. (**a**) Track segment demo. (**b**) Corresponding CPN model

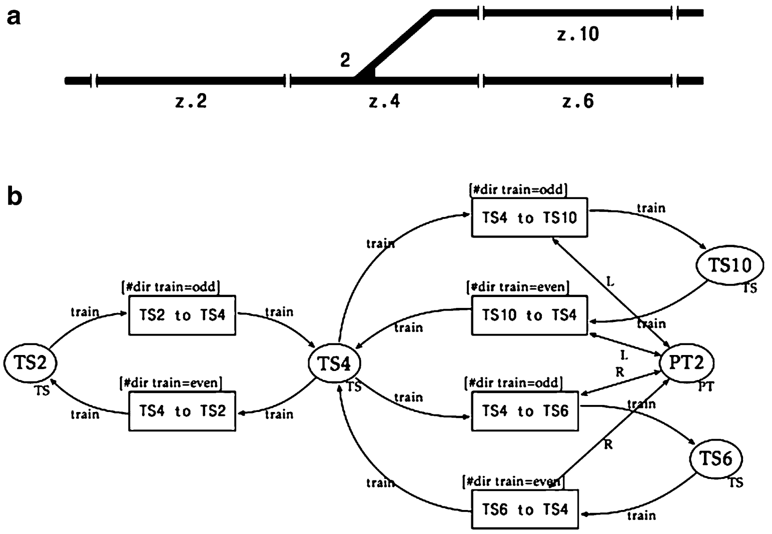


Fig. 7.11 A Petri net representation of point component. (**a**) Point demo. (**b**) Corresponding CPN model

Points

Figure 7.11 shows a CPN model of a point component. In the French railway system, a point is attached to a track segment, as shown in Fig. 7.11a. In its corresponding model, the point is represented by a single place that stores the current connection information (left or right). In the French system, the position “left” or “right” refers to the tracks on the left or right side when facing a point. This point place works as a condition place of 4 transitions (movements). However, its position will not affect the movements between TS2 and TS4 as they are constantly connected.

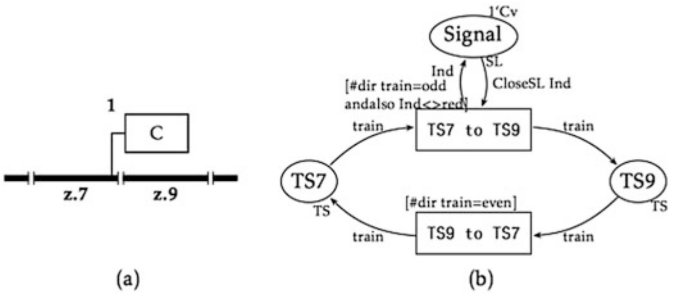


Fig. 7.12 A Petri net representation of signal light. (**a**) Signal light demo. (**b**) Corresponding CPN model

Signal Lights

Figure 7.12 shows a CPN model of a signal light component. Normally, a signal light can only be in charge of one direction of the transit. In Fig 7.12a, the movement from TS7 to TS9 is controlled by signal light. So, in Fig. 7.12b, signal place is only connected to the transition "TS7 to TS9." This transition is only enabled when the token (indicator color) of signal place is not "red." After a train passes the signal light (firing the transition), the signal light is switched off by setting the indicator to red. The operator "<>" in the guard function means "not equate to ($\neq$)."

Automatic Unlock Devices

In the French system, there is a ground-based automatic mechanism that could unlock the interlocked formation by the action of train passage (Rétiveau 1987). This mechanism is used for a flexible transit, and it is called the "DA" mode. This DA mode is fully automatic and ground-based, so we treat it as a fixed installation, rather than part of the signaling operations. The conditions of establishing a DA mode interlocking route are:

- There is a pedal (see in Fig. 7.13a.) on the track segment.
- The direction of the interlocking route is the same as the direction of the pedal.

If a route is established in DA mode, when a train passes and activates the pedal, all the upstream tracks will be automatically unlocked. Based on the original model in Fig. 7.11, this type of mechanism is represented with two additional parallel transitions. Each DA sub-model unlocks a track segment that is stored in the fusion place (see in Fig. 7.13c).

In our research, a typical station from the French railway signalization book (Rétiveau 1987) is studied, shown in Fig. 7.14. It is only half of the station that contains 5 points, 6 effective signal lights, 12 track segments, and 13 complete interlocking routes. The detailed information about this case study can be found

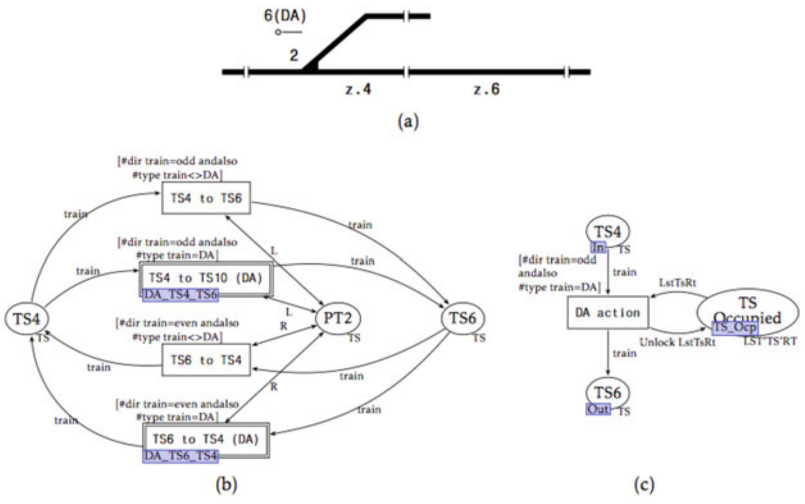

6(DA)
2
z.4
z.6
(a)
[#dir train=odd andalso #type train<>DA]
TS4 to TS6
[#dir train=odd andalso #type train=DA]
TS4 to TS10 (DA)
DA_TS4_TS6
[#dir train=even andalso #type train<>DA]
TS6 to TS4
[#dir train=even andalso #type train=DA]
TS6 to TS4 (DA)
DA_TS6_TS4
TS4
TS6
PT2
train
L
R
TS
(b)
TS4
In
TS
[#dir train=odd andalso #type train=DA]
DA action
LstTsRt
TS Occupied
TS_Ocp
Unlock LstTsRt
LST'TS'RT
TS6
Out
(c)

Fig. 7.13 A Petri net representation of “DA” mode. (**a**) DA mode. (**b**) Corresponding CPN model. (**c**) Sub-model of DS_TS4_TS6

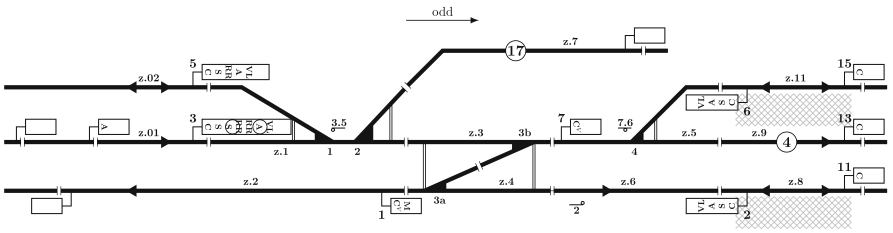


Fig. 7.14 Case study of a station layout

in Chapter 15 in (Rétiveau 1987). This case study example has been chosen as an academic benchmark by experts involved in the PERFECT project (Collart-Dutilleul et al. 2014; Sun et al. 2014).

The whole layout is represent by the CPN model in Fig. 7.15, using the basic components that have been discussed before. This layout allows all the train movements according to the interlocking routes.

Together with the signal operation parts in Sect. 4.3.1, the whole HCPN model is a complete RIS specification. It can perform basic functions of an RIS by automatically arranging the routes according to different train commands, blocking the inverse path and signal light when a route is established, and enabling the route destruction function after the train passes through. The whole model is too big and not necessary for a detailed demonstration in this section. However, all the other nets are modelled by the previous methodology.

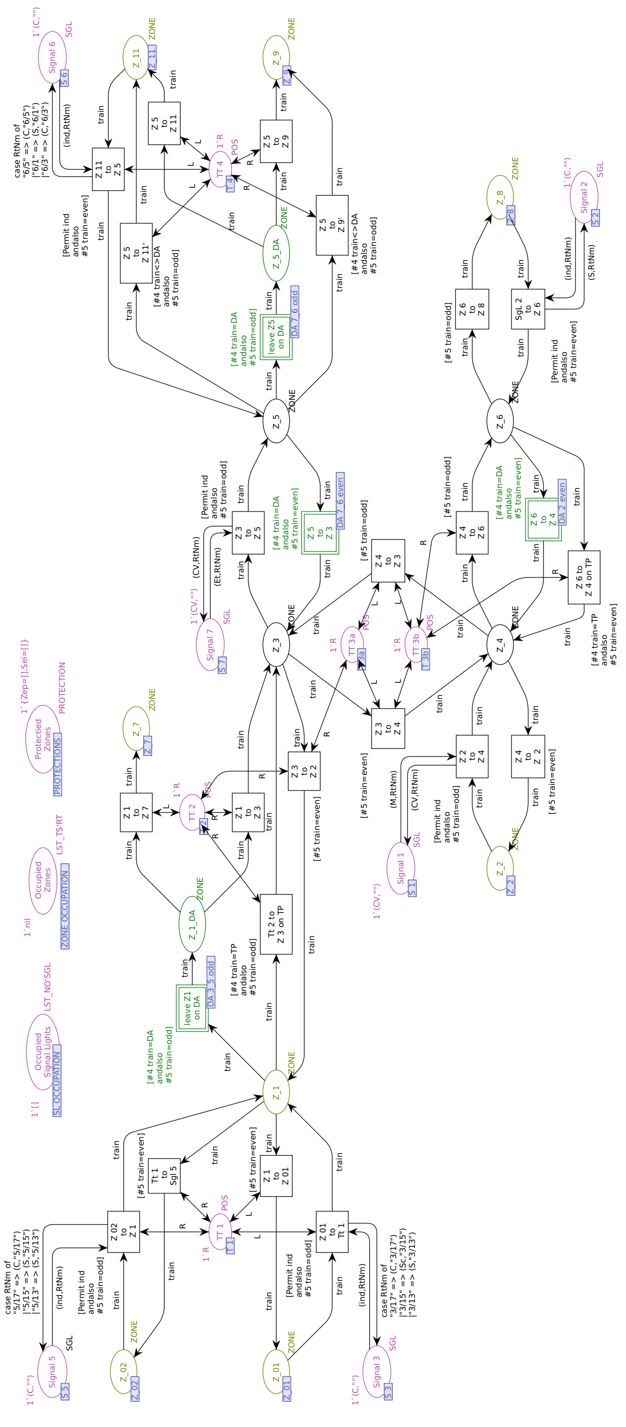


Fig. 7.15 The Petri net model of route layout

7.4.4 A Pattern of Railway Interlocking Modelling

An RIS has two main parts: the signaling operations and the fixed installations. In each station, signaling operations are localized instances of the national railway standards, which monitor and control the status of the fixed installations. It could be established via a hierarchical structure as we discussed in Sect. 4.3.1.

However, fixed installations consist of a series of track-side appliances, which are diverse in practice, as each station has its own rail route structure. Specification and evaluation of each station along a railway line is a repetitive and tedious job, and it has low efficiency and will probably introduce new errors from re-modelling processes. A feasible solution is to summarize all the common parts of the RIS and establish a parameterized model framework that can be applied to all stations. This study can be found in our previous work (Sun et al. 2015).

In this section, a generalization model pattern is presented, which is a reusable solution for the RIS with PIPC type. Models of different stations can be derived from this pattern without re-modelling, just changing the configurations in the pattern.

7.4.4.1 Generalization Concept

The stations that are equipped with the same type of RIS follow the same national rules. The only differences are the layouts of their fixed installations.

The expected structure should be both general and parameterized, which allows the specifications of stations to be derived from the same model with diverse configurations. That is to say, in this structure, the unmarked colored Petri net is a set of RIS functional rules, while the initial tokens are the concrete performance of stations. In such a model framework, the configurations (tokens) represent all the scenario information, based on the formation of the RIS layout and the "condition table" (or control table). When modelling a new station, the only job is to change the initial tokens on the expected structure.

To have this general structure, the railroad layouts cannot be performed by the physical location of places and connection of transitions. However, this information is indeed important for train movements, so all this diverse information must be represented in the token forms, ensuring the PN structure itself remains universal.

For a better understanding of the generalization concept, we use an incremental process and comparison examples to illustrate how to generalize the railroad structure.

Basis Track segments

Compared to Fig. 7.10, the new model in Fig. 7.16 has the same performance capabilities but in a parameterized form. A token in "train location" place indicates the train ID and its current location. Each time the transition occurs, the value of

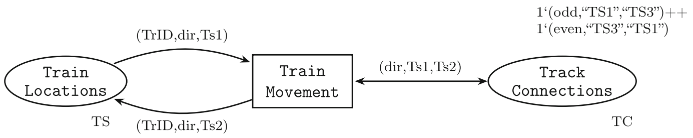


Fig. 7.16 Generalized representation of track segments

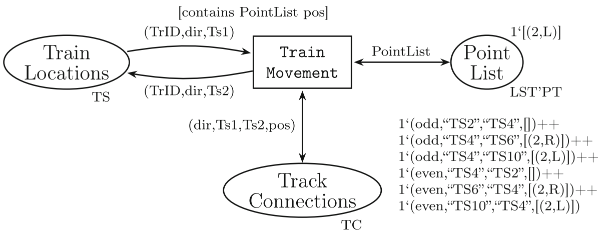


Fig. 7.17 Generalized representation including points

the train token will be refreshed according to the enabled binding elements. The “track connection” place is the constraint of train movement, which guides the train to move forward.

Adding Points

When we introduce the points into the generalized structure, it will first need a place to “store” all the point information, including point IDs and the positions. Meanwhile, the points will have an impact on the train movements, so the configuration of track connection should be modified. The new model in Fig. 7.17 is the corresponding model of the example in Fig. 7.11. The new color set of *TC* contains the point constraints. Only when the point stored in the point list place satisfies the point constraints, the train can move.

Adding Signal Lights

Similar to a point, a signal light is also the movement constraint. So the introduction of signal lights comes with a new place and a modification to the color set of *TC*. The new model in Fig. 7.18 is the corresponding model of the example in Fig. 7.19. The function *SL’Permit* checks the corresponding signal indicator. If the indicator is *red* (Cv in French), then it returns *false* to prevent train movement. Otherwise, it returns *true* to permit the transit. The function *SL’Close* switches off the corresponding signal lights after firing the transition.

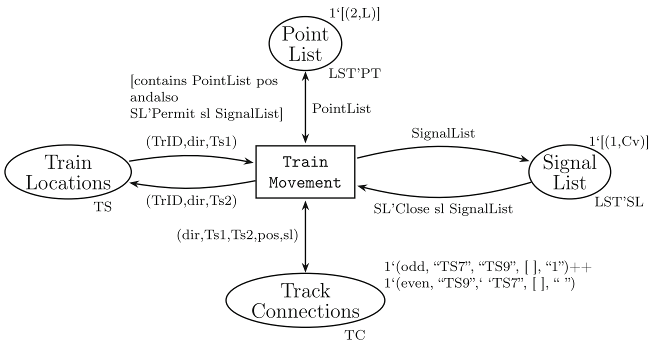


Fig. 7.18 Generalized representation including signal lights

From the above examples, we can conclude that the components of railroad and their combinations can be expressed by generalized structure, using constraint places and different transition conditions.

However, in a real practice, there are more constraints (appliances) and rules. First, we should list all the scenario-related elements and prepare their specification forms for the expected model.

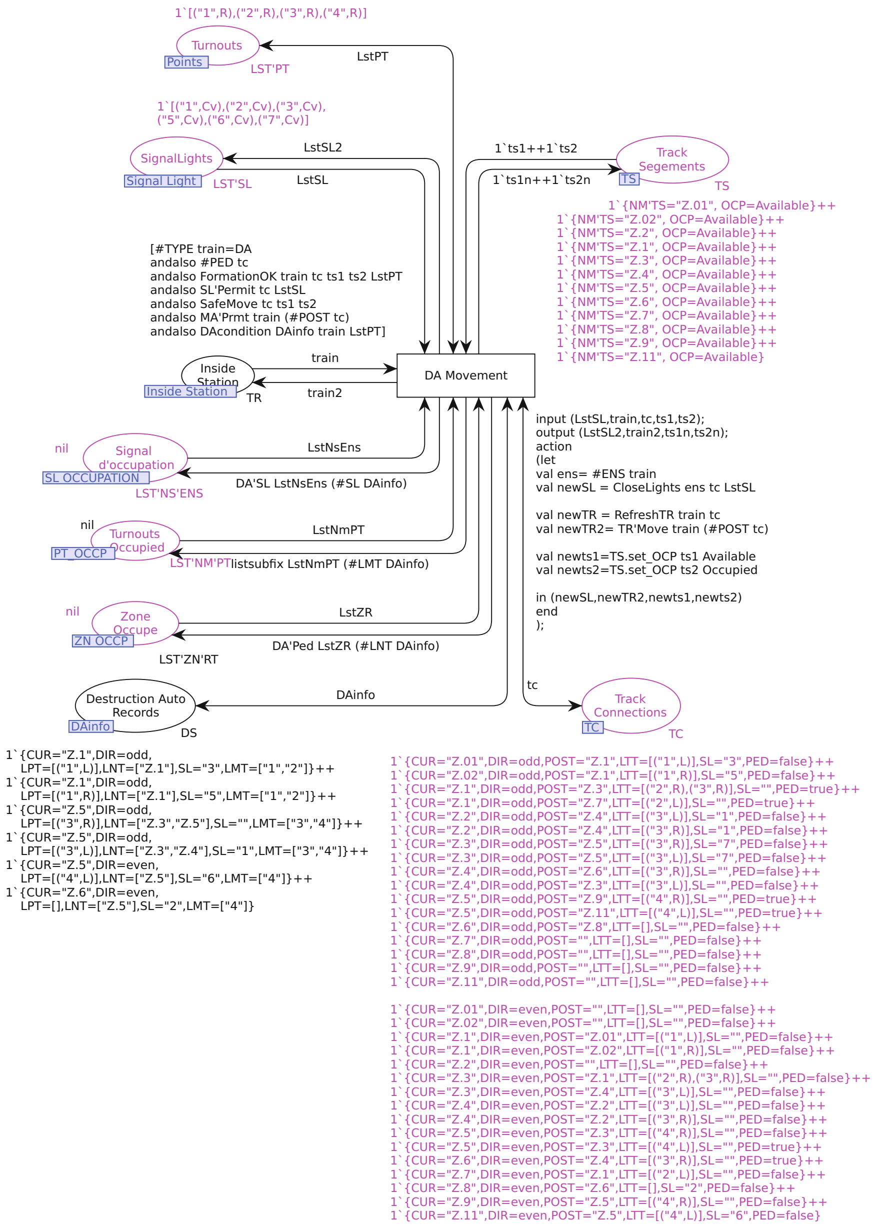
In Table 7.2, train, track, point, and signal light are normal components that we have introduced in the previous parts. In this table, we give them several attributes to distinguish between each token. The *Track Connection* stores all the connection information between different tracks, considering the constraints of points, signal lights, and formation release triggers (the pedals). The *pedal* is the prerequisite condition for "DA" mode interlocking route. The "Destruct Auto" is the automatic unlock mechanism and its devices. It contains the related unlock conditions and the unlock actions.

With all these variables and their notations, the next step is to describe the movement of a train. Although the expected model does not have visible routes, we can determine train movement by token values. If the value of the train position changes, that means this train actually moves. Generally, there are two types of routing routes, DA and TP, in the French national context.[2] We also consider the route for shunting (OM), and the "staff responsible" mode (SR) for override operations. However, due to the space limitation, only DA mode will be discussed in this section.

The conditions for enabling DA movement are:

- There should be a pedal (passage detector for DA mode) in the current track.
- Points of the route must be proper positioned.
- Signal light (if any) in front of the train should be green.
- Train's movement authority allows it to move onto the next track.

[2]DA: Destruction automatique, TP: Tracé permanent.



```
1`[("1",R),("2",R),("3",R),("4",R)]
```

```
1`[("1",Cv),("2",Cv),("3",Cv),
("5",Cv),("6",Cv),("7",Cv)]
```

```
1`{NM'TS="Z.01", OCP=Available}++
1`{NM'TS="Z.02", OCP=Available}++
1`{NM'TS="Z.2", OCP=Available}++
1`{NM'TS="Z.1", OCP=Available}++
1`{NM'TS="Z.3", OCP=Available}++
1`{NM'TS="Z.4", OCP=Available}++
1`{NM'TS="Z.5", OCP=Available}++
1`{NM'TS="Z.6", OCP=Available}++
1`{NM'TS="Z.7", OCP=Available}++
1`{NM'TS="Z.8", OCP=Available}++
1`{NM'TS="Z.9", OCP=Available}++
1`{NM'TS="Z.11", OCP=Available}
```

```
[#TYPE train=DA
andalso #PED tc
andalso FormationOK train tc ts1 ts2 LstPT
andalso SL'Permit tc LstSL
andalso SafeMove tc ts1 ts2
andalso MA'Prmt train (#POST tc)
andalso DAcondition DAinfo train LstPT]
```

```
input (LstSL,train,tc,ts1,ts2);
output (LstSL2,train2,ts1n,ts2n);
action
(let
val ens= #ENS train
val newSL = CloseLights ens tc LstSL

val newTR = RefreshTR train tc
val newTR2= TR'Move train (#POST tc)

val newts1=TS.set_OCP ts1 Available
val newts2=TS.set_OCP ts2 Occupied

in (newSL,newTR2,newts1,newts2)
end
);
```

```
1`{CUR="Z.1",DIR=odd,
   LPT=[("1",L)],LNT=["Z.1"],SL="3",LMT=["1","2"]}++
1`{CUR="Z.1",DIR=odd,
   LPT=[("1",R)],LNT=["Z.1"],SL="5",LMT=["1","2"]}++
1`{CUR="Z.5",DIR=odd,
   LPT=[("3",R)],LNT=["Z.3","Z.5"],SL="",LMT=["3","4"]}++
1`{CUR="Z.5",DIR=odd,
   LPT=[("3",L)],LNT=["Z.3","Z.4"],SL="1",LMT=["3","4"]}++
1`{CUR="Z.5",DIR=even,
   LPT=[("4",L)],LNT=["Z.5"],SL="6",LMT=["4"]}++
1`{CUR="Z.6",DIR=even,
   LPT=[],LNT=["Z.5"],SL="2",LMT=["4"]}
```

```
1`{CUR="Z.01",DIR=odd,POST="Z.1",LTT=[("1",L)],SL="3",PED=false}++
1`{CUR="Z.02",DIR=odd,POST="Z.1",LTT=[("1",R)],SL="5",PED=false}++
1`{CUR="Z.1",DIR=odd,POST="Z.3",LTT=[("2",R),("3",R)],SL="",PED=true}++
1`{CUR="Z.1",DIR=odd,POST="Z.7",LTT=[("2",L)],SL="",PED=true}++
1`{CUR="Z.2",DIR=odd,POST="Z.4",LTT=[("3",L)],SL="1",PED=false}++
1`{CUR="Z.2",DIR=odd,POST="Z.4",LTT=[("3",R)],SL="1",PED=false}++
1`{CUR="Z.3",DIR=odd,POST="Z.5",LTT=[("3",R)],SL="7",PED=false}++
1`{CUR="Z.3",DIR=odd,POST="Z.5",LTT=[("3",L)],SL="7",PED=false}++
1`{CUR="Z.4",DIR=odd,POST="Z.6",LTT=[("3",R)],SL="",PED=false}++
1`{CUR="Z.4",DIR=odd,POST="Z.3",LTT=[("3",L)],SL="",PED=false}++
1`{CUR="Z.5",DIR=odd,POST="Z.9",LTT=[("4",R)],SL="",PED=true}++
1`{CUR="Z.5",DIR=odd,POST="Z.11",LTT=[("4",L)],SL="",PED=true}++
1`{CUR="Z.6",DIR=odd,POST="Z.8",LTT=[],SL="",PED=false}++
1`{CUR="Z.7",DIR=odd,POST="",LTT=[],SL="",PED=false}++
1`{CUR="Z.8",DIR=odd,POST="",LTT=[],SL="",PED=false}++
1`{CUR="Z.9",DIR=odd,POST="",LTT=[],SL="",PED=false}++
1`{CUR="Z.11",DIR=odd,POST="",LTT=[],SL="",PED=false}++

1`{CUR="Z.01",DIR=even,POST="",LTT=[],SL="",PED=false}++
1`{CUR="Z.02",DIR=even,POST="",LTT=[],SL="",PED=false}++
1`{CUR="Z.1",DIR=even,POST="Z.01",LTT=[("1",L)],SL="",PED=false}++
1`{CUR="Z.1",DIR=even,POST="Z.02",LTT=[("1",R)],SL="",PED=false}++
1`{CUR="Z.2",DIR=even,POST="",LTT=[],SL="",PED=false}++
1`{CUR="Z.3",DIR=even,POST="Z.1",LTT=[("2",R),("3",R)],SL="",PED=false}++
1`{CUR="Z.3",DIR=even,POST="Z.4",LTT=[("3",L)],SL="",PED=false}++
1`{CUR="Z.4",DIR=even,POST="Z.2",LTT=[("3",L)],SL="",PED=false}++
1`{CUR="Z.4",DIR=even,POST="Z.2",LTT=[("3",R)],SL="",PED=false}++
1`{CUR="Z.5",DIR=even,POST="Z.3",LTT=[("4",R)],SL="",PED=false}++
1`{CUR="Z.5",DIR=even,POST="Z.3",LTT=[("4",L)],SL="",PED=true}++
1`{CUR="Z.6",DIR=even,POST="Z.4",LTT=[("3",R)],SL="",PED=true}++
1`{CUR="Z.7",DIR=even,POST="Z.1",LTT=[("2",L)],SL="",PED=false}++
1`{CUR="Z.8",DIR=even,POST="Z.6",LTT=[],SL="2",PED=false}++
1`{CUR="Z.9",DIR=even,POST="Z.5",LTT=[("4",R)],SL="",PED=false}++
1`{CUR="Z.11",DIR=even,POST="Z.5",LTT=[("4",L)],SL="6",PED=false}
```

Fig. 7.19 Generalized Petri net model of "DA" route pattern

Table 7.2 Scenario-related elements in general structure

Element	Content	Notation
Train	Train name	NmTr
	Train direction	DirTr
	Route name	NmRt
	Route type (DA,TP,etc)	TpRt
	Train position	PosTr
	Movement authority	MA
Track	Track name	NmTs
	Occupation status	Ocp
Track connections	Current track	CurTs
	Connection direction	DirTs
	Post-track	PostTs
	Points (number varies [0,2]) (with name and its position)	PtTs
	Signal light name [0,1]	NmSl
	Indication of pedal	Ped
Point	Point list (contains name and its position)	LstPt
Signal light	Signal light list (contain name and its color)	LstSl
Destruct Auto	Exiting track (where DA takes place)	TsDa
	Effective direction of pedal	DirDa
	Tracks to be destructed	TsLstDa
	Signal light to release	SlDa
	Points to release	PtDa

The actions that release the formation of the route along with train movement:

- Release tracks of the route behind the train
- Release points of those tracks
- Switch off signal light (if any) after passing

For analysis purposes, we introduce a security guard function that constantly checks the occupation of the front track. The train's movement is safe provided that the front track is clear. Otherwise, if the front track is occupied, there will be a "face to face" or "face to tail" collision.

From what has been mentioned above, the more formal definition of the enabling rules of the DA movement is shown in Table 7.3. With the help of *CPN ML* language, all the conditions above can be embedded into one transition and can be combined into a single model to represent all the DA mode movements.

The study case of Fig. 7.15 is modelled by the generalization concept above. The complete CPN model provides a pattern that could be applied to all the relay-based computer-controlled RIS in the French national context. It can automatically arrange the routes for different trains, block the incompatible routes when a certain route is established, enable the route destruction function after a train passes, and support four types of route modes along with their mixed traffic operations. The

Table 7.3 Conditions and equations of "DA" movement

Condition	Equation

Route type	TpRt = DA
	Ped=TRUE

Route formation	PosTr = CurTs
	DirTr=DirTs
	PtTs $\subseteq$ LstPt

Signal open	(NmSl,green) $\subseteq$ LstSl
Movement authority	PostTs$\in$ MA

DA activated	TsDa = PosTr
	DirDa = DirTr

To release	TsLstDa SlDa PtDa
Security check	Ocp of CurTs = Occupied Ocp of PostTs = Clear

whole model is really large for a demonstration. Only one layer of the model and its results will be introduced. The other parts of the model are built by successive implementation.

Figure 7.19 shows the DA module of the general structure that includes all the necessary elements mentioned before: tokens of train, track segments, track connections, points, signals, and information of automatic destruction. Then, this transition is ordered by the conditions and fulfils the following actions. Train tokens are stored in an "Inside Station" place, with all the trains within this station. All tokens in this module do not really transit. They only "update" the data inside themselves.

Supposing we have the following initial parameters of simulation:

- Train demand route "3/15" : 1'{ NmTr= "TER-0315" , DirTr= odd, NmRt= ("3" ,"15"), TpRt= DA, PosTr = "" , MA = []} ;
- List of all points: 1'[("1" ,R),("2" ,R),("3" ,R),("4" ,R)];
- List of all signal lights: 1'[("1" ,Cv), ("2" ,Cv), ("3" ,Cv), ("5" ,Cv), ("6" ,Cv), ("7" ,Cv)].

The simulation result of CPN tools is shown in Table 7.4. After the establishment of the route "3/15," related points change their position, and related track segments are blocked in memory. After switching on, signal lights change their indication and become blocked. After receiving an MA, the train can start with permission. As the train moves, its MA is gradually reduced and block components are released by the mechanism of automatic destruction. When MA equals zero, the train stops right away and triggers the route destruction. Finally, all the blocked components become free and the train exits the station.

Table 7.4 Result of route “3/15” simulation

Last action	Train token	Signal lights	Points Points	Tracks occupied	Signals occupied
Initial	Canton=Z.01 MA=()	(3,Cv) (7,Cv)	(1,R),(2,R) (3,R),(4,R)		
Route establish	Canton=Z.01, MA=()	(3,Cv) (7,Cv)	(1,L),(2,R) (3,R),(4,L)	Z.01, Z.1, Z.3, Z.5, Z.11	
Open signal lights	Canton=Z.01, MA=()	(3,VL) (7,Et)	(1,L), (2,R) (3,R),(4,L)	Z.01, Z.1, Z.3, Z.5, Z.11	3, 7
Generate MA	Canton=Z.01, MA=(Z.1,Z.3,Z.5,Z.11)	(3,VL) (7,Et)	(1,L), (2,R) (3,R),(4,L)	Z.01, Z.1, Z.3, Z.5, Z.11	3, 7
Z.01→ Z.1	Canton=Z.01, MA=(Z.3,Z.5,Z.11)	(3,Cv) (7,Et)	(1,L), (2,R) (3,R),(4,L)	Z.1, Z.3, Z.5, Z.11	3, 7
Z.1 → Z.3	Canton=Z.01, MA=(Z.5,Z.11)	(3,Cv) (7,Et)	(1,L), (2,R) (3,R),(4,L)	Z.3, Z.5, Z.11	7
Z.3→Z.5	Canton=Z.01, MA=(Z.11)	(3,Cv) (7,Cv)	(1,L), (2,R) (3,R),(4,L)	Z.5, Z.11	7
Z.5 →Z.11	Canton=Z.01, MA=()	(3,Cv) (7,Cv)	(1,L), (2,R) (3,R),(4,L)	Z.11	
Destruction		(3,Cv), (7,Cv)	(1,L), (2,R) (3,R),(4,L)		

Then we use the state space analysis function that is embedded in CPN tools to analyze the space state of this simulation. Its calculation result shows that this “single train” scenario has 26 states and 32 arcs. There is not any deadlock or live lock in the system. Then we perform another two simulations with 2 trains and 3 trains demanding for different interlocking routes. The sizes of the state space are 339 and 2025, and all the states are “safe.”

7.4.5 *An Event-Based Approach for Relay-Based Logic*

In the previous two sections, we mainly focus on the high-level parts of the RIS. More precisely, we study and model the computer-controlled parts of the RIS. In this section, we analyze the low-level parts of RIS. That is the modelling methodology of the relay-based systems.

7.4.5.1 Background of Relay-Based Logic

All the controls and commands that come from the high-level part of RIS are implemented by a set of relays. They achieve the control procedures by changing their states. Most relays have two states, activated and deactivated, sometimes may be left and right. Because of different functional purposes, the relay circuit diagrams can be divided into separate diagrams. For example, according to the book (Rétiveau 1987), the functional phases of the route establishment of the PRCI type have four stages:

- Route formation: receiving the route command from the dispatcher, and setting point to the right position by point machines.
- Formation verification for interlocking: verifying the positions of the points relay. If all the relays are properly positioned, the formation will be interlocked.
- Route verification: verifying the real point positions; if they are well positioned, then send a command to signal light control logic.
- Signal light control: switching on the lights and displaying different colors depending on the interlocking route itself.

For a better understanding, we create a small scenario with only one point. This example is designed on the basis of the control logic and the circuit diagram in Fig. 15.23, Fig. 15.27, Fig. 15.29, Fig. 15.39, Fig. 15.40, Fig. 15.46 in Rétiveau (1987), and it is shown in Fig 7.20. The example contains the main components for route establishment. It is realized by a set of relays and switches that are located

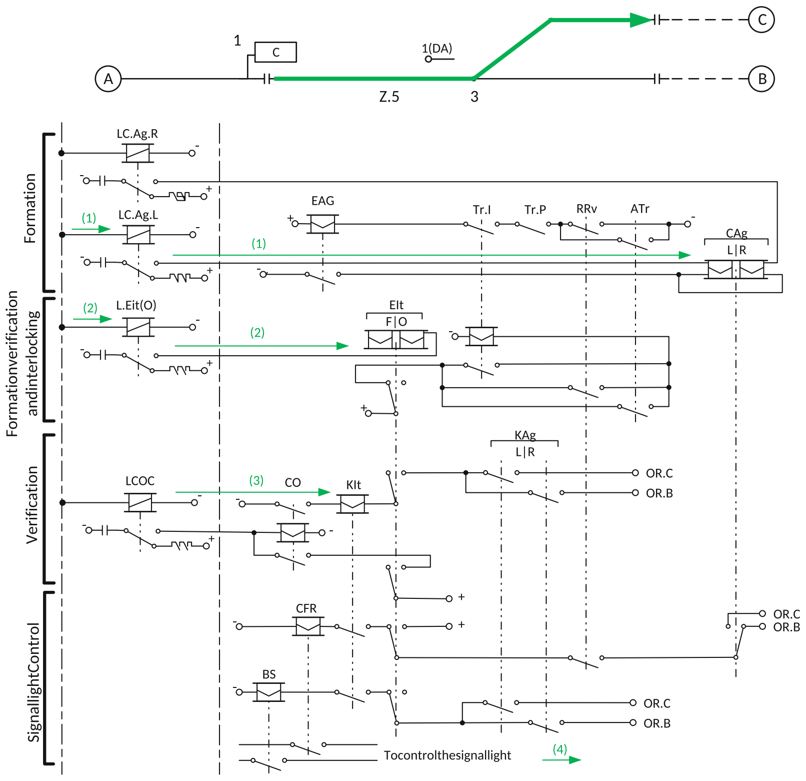


Fig. 7.20 An example of PRCI type system of a single point

in different layers (circuits diagram). However, as shown in Fig. 7.20, the dash-dotted line connected elements, in nature, are the same element. They are physically connected together, changing their states at the same time, but located in different circuits. The established procedures of this example are explained as follows:

- After receiving the formation command (*LC.Ag.L* => left or *LC.Ag.L* => right), the control relay *CAG* is going to change for the preparation of the route.
- After the point is well positioned, interlocking command *L.EIt(O)* is sent to interlock the enable relay *EAG* by locking its transit with *Tr.I* or *Tr.P*.
- When command *LCOC* is received, if the point is in the right position and well-locked, a further command will be sent to control the signal light.
- Switching on the signal light according to the relays *CFR* and *BS*.

From Fig. 7.20 and its procedures, we know that relays can be activated or deactivated in different layers by commands from the signaling center, occupation changes of the track segments, or the internal relay state changes. Moreover, each switches affiliated with these relays will be changed at the same time. Consequently, once a relay changes its values, all the related circuits will be refreshed simultaneously. However, this kind of concurrence is quite different from the rules in CPN. It has brought some problems in our early attempts. Nonetheless, all these problems are caused by the HCPN models that consist of several subnets. If all the logic connections are modelled in a single net, we can combine all the linked elements into one element (place), and there will be no further problem of concurrence. But, in that way, we will obviously lose the readability of the model and lose the description of the system's structure. So all the following problems, discussions, and their solutions are based on the model with multiple nets.

The following part begins with two simple examples to illustrate the problems. Then, we apply the event-driven concept to solve these problems.

Modelling Problem I: Synchronous Firing

In the envisioned model with the hierarchical structure, relays and switches are located in different nets. So if a relay changes its state, the related transitions cannot fire at the same time. As the states of the relays are closely coupled to each other, the dissynchronization of firing transitions fails to refresh the system simultaneously, and it may lead the system to uncertain states, such as standstill, livelock, deadlock, or even an unreasonable state. Such an example can be found in Fig. 7.21.

This example describes two logical processes that are controlled by relay A and relay C. Processes are placed in different nets, and each one has two transitions. Assuming the initial state is $S_{init} = [m, n|A, B, C] = [1, 1|1, 1, 1]$, the expected firing sequence is: $T_{1n}, T_{1m} \longrightarrow T_{2m}, T_{2n}$, and the expected final state is $S_{init} = [3, 3|1, 0, 1]$. But if the transition T_{2n} fires before T_{1m}, the result state is $[m, n|A, B, C] = [1, 3|1, 0, 1]$. This state does not exist in a real system, and it may cause unknown problems. This issue demands a transition management mechanism that could organize all the marking-enabled transitions to be fired in the right orders,

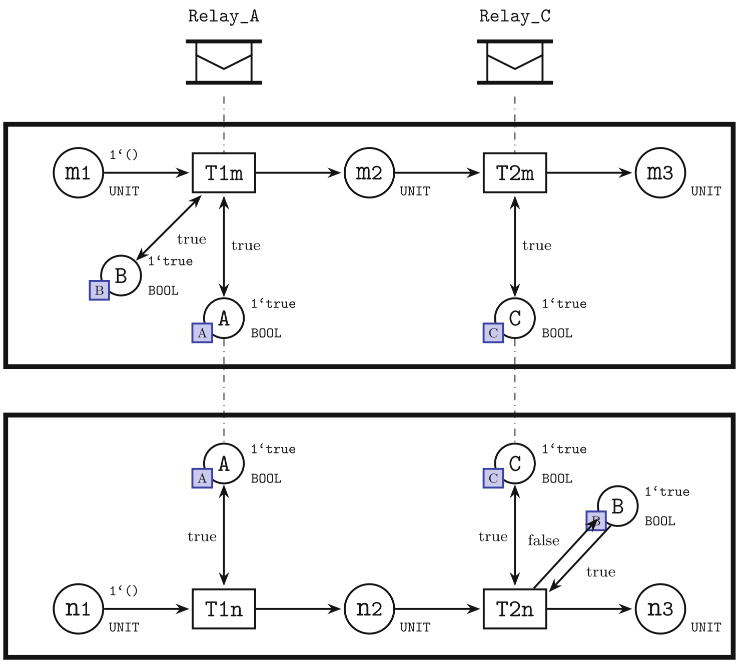


Fig. 7.21 Modelling problem I: synchronous firing

as they do in the real system. Moreover, considering the compatibility, the proposed solution should be achieved under the framework of CPN.

Modelling Problem II: Firing Conditions

Generally, a relay’s status is controlled by several circuit elements, including electrical sources and switches. These elements can be considered as constant variables. If a relay is controlled by such constant variables, no matter the order, when all the elements meet the required conditions, the relay is activated. However, there is another “temporary” type of conditions. They are pulse signals that are a kind of instant variables. A relay connected to such pulse signals will only be activated at the “pulse” moment. For such a relay, we need to pay more attention to its activating condition order. The example is shown in Fig. 7.22a.

The *Cmd*_E is a command from signaling control and the *SW_F* is a controlled switch. Their states affect the value of *Relay_G*. If we have a corresponding model, as shown in Fig. 7.22b, we will encounter the unreasonable firing sequence: $E = true \rightarrow F = true \rightarrow t1$. In order to solve this problem, a reset mechanism (non-timed CPN approach) can be applied or time concept (timed CPN approach) can be introduced. Considering that an RIS is more like a continuous sequence event

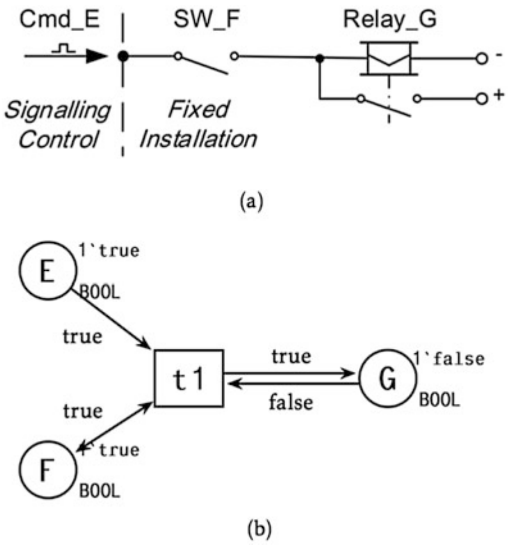


Fig. 7.22 Modelling problem II: firing conditions. (**a**) Example of different conditions. (**b**) Corresponding model

system, it is not necessary to add time factors into our models. The rest solution would be a new mechanism to differentiate two kinds of condition types with good readability.

7.4.5.2 Event-Driven Concept

In relay-based systems, every circuit state change is driven by an event, such as external commands or internal switch actions. Such a mechanism reminds us of a special PN—the controlled Petri net (CtlPN). It is a class of Petri nets with external enabling conditions called control places that allow an external controller to influence the progression of tokens in the net (Holloway and Krogh 1994; Holloway et al. 1997). Figure 7.23 illustrates a controlled Petri net, where the squares (*c1, c2, c3*) indicate the external control place.

As with the ordinary Petri nets, the state of a CtlPN is given by its marking, which is the distribution of tokens in places. A controlled transition can only be fired when this transition is marking-enabled and the connected control places are "TRUE."

Inspired by this occurrence rule, we design a similar mechanism to solve our previous problems under the framework of CPN without breaking any existing rules of CPN. This mechanism is achieved by introducing event-based enabling rules and an *event place* into ordinary CPN models. An event-driven model is a class of Petri nets with event conditions stored in the event place (fusion type place), which makes the connected transitions event-driven, in order to allow internal/external event to influence the progression of tokens. The event place contains an FIFO list that stores all the events in progress in their order of occurrence. This FIFO list has the following properties:

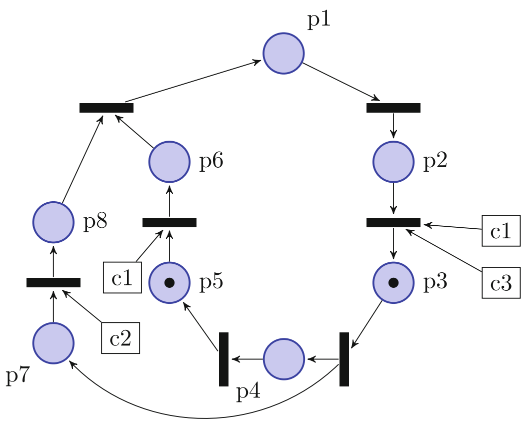


Fig. 7.23 An example of controlled Petri net

- Only the head (first element) of the list is referred as the current *activated* event and *tt* will activate its corresponding transitions.
- The tail (exception of the first element) of the list is considered as *deactivated* until the head of the list is removed. The new head will become activated.
- New events that are induced by internal actions are stored at the end of the list.
- Only when the system has no more events in this list, this system can accept an external command.

As in the Petri net literature, it is commonly assumed that only one transition can be fired at a given instant. So, parallel actions become "choices." If one transition introduces new internal events (relay status changes) before the last event is complete, the system status will appear confusing. However, with the help of event places, we can achieve a loose synchronization of firing the transitions. It continues firing all the enabled transitions related to the first event until there are no more enabled transitions. Then an event management function (transition) will be enabled. It removes the "useless" event (the first event), then moves on to the next event, and makes it the new head of the list. In this way, the whole system is gradually progressing forward, event by event, in order to imitate a synchronization system.

The expected event-driven model has 4 transition priorities: $P_{Event} > P_{Clear} > P_{normal} > P_{External}$.

P_{Event} belongs to the event-related transitions that are directly connected to event place.

P_{normal} belongs to the set of transitions that are not directly connected to event place.

P_{Clear} belongs to an event remove mechanism that will remove the "useless" event from the FIFO list if this event cannot fire any transitions.

$P_{External}$ belongs to external inputs for scenario analysis and state space calculation purposes.

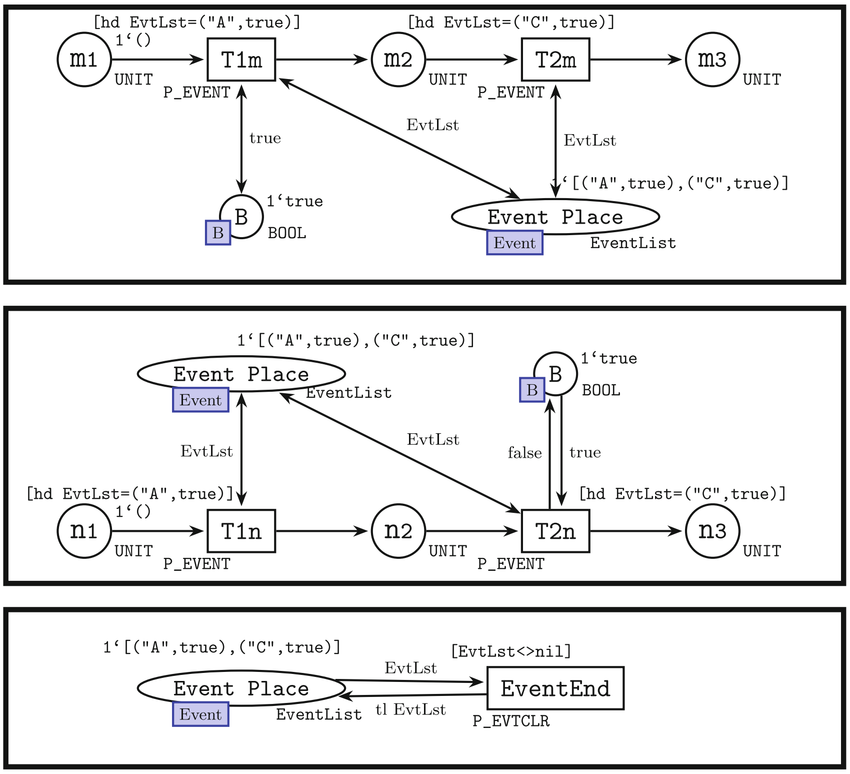


Fig. 7.24 Event-driven colored Petri net model of Fig. 7.21

Now, we can rebuild the example in modelling problem I: “the synchronous firing” with the event-driven concept. In Fig. 7.24, the color set of events is defined as *colset Event= STRINGxBOOL*. It contains the name of the event and its value, for example (“A,” true) means relay “A” is activated and (“B,” false) means relay “B” is inactivated. All the transitions are connected to an “Event Place” that stores the events to be triggered in their order of occurrence. Its color set is *colset EvntList= list Event*. The token in this place is in the form of *list* type. The head of the list (*hd* list, in meta language grammar, is to abstract the first element from the list) represents the event that is currently taking place in the system. The guard function checks the first element of the event list (*hd EvtLst*) and determines whether the transition is event-enabled or not. Any event-enabled transition has the ability to fire, and it can fire if it is also marking-enabled. Moreover, if a transition brings in a new event, then this new event will be stored at the end of the event list in “Event place,” and it can be triggered in later progress. After all the enabled high-priority transitions are fired, the transition with low priority is enabled. It will remove the current activated event from the list (*tl EvtLst* returns a new list with exception of the first element) and the second event becomes activated.

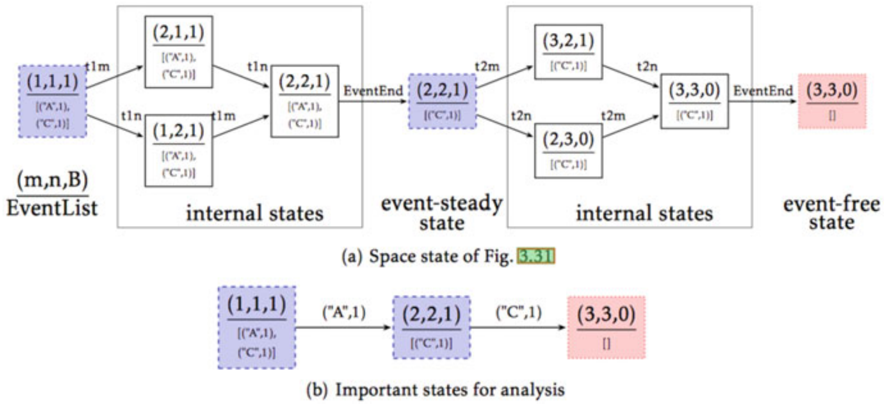


Fig. 7.25 Simplification rules of system space state. (**a**) Space state of Fig. 3.26. (**b**) Important states of analysis

The state space of this model is shown in Fig. 7.25. For a concise indication, in this state space graph, the system state is represented by the marking of the vector (m, n, B). Here, m is a mapping from markings of $(m1, m2, m3)$, and $m \rightarrow \{0, 1, 2\}$ represents the markings of $\{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$. Similarly, $n : (n1,\ n2,\ n3) \rightarrow \{0,\ 1,\ 2\}$. B indicates the marking in "B place " and 1/0 is used to represent "true/false." The inscriptions under the vector are the content of the FIFO event list. The label on the arcs between two states is the fired transition. The initial state of the system is $(m, n, B) = (1, 1, 1)$, $EventList = \left[\left(“A'', 1\right)\left(“C'', 1\right)\right]$. Each time the transition EventEnd is fired, an event will be removed from the event list.

The state in blue is called "event-steady" state. This means that a previous event is finished and begins to activate a new event. The state in red is an "event free" state. This means there are no more events and the system state is preserved until there is an external input event. The state in white is the internal state, or instantaneous state. Between two successive system-steady states, there may be more than one path, and the number of combinations of the path depends on the number of parallel transitions, which could result in a large number of system states. But no matter how the state changes, it will eventually be stabilized and finally reach the next steady state.

When we analyze this space state graph, we will find that not every state has equal importance. The event-steady and event-free states are more concise to describe the safety reachability of a system. Hence, an abstraction method to minimize the size of the system state will be demonstrated in Fig. 7.25b. From the perspective of analysis, the internal states are not useful because they have less value than the steady ones. Each internal state is a tiny change inside the fixed installations, only when the system finishes all the changes in a space state path, which means a complete response to the external input. While, from the modelling point of view, all the states and changes between two steady states should not exist in the real system, because they are parallel at the same time, as in the modelling result, these states can be considered as transient states.

Therefore, the original state space in Fig. 7.25a can evolve into a quite simple one in Fig. 7.25b. The new state space has an initial state (1, 1, 1) and two external input events [(“A”, 1) , (“C”, 1)] , and each event allows the system to advance into a new state. This method will effectively reduce the state space complexity caused by the relay-based components that act simultaneously in different layers.

Also the modelling problem II: the “firing condition” can be solved by the event-driven model in Fig. 7.26. The original pulse signal Cmd_E was replaced by a single event in the “Event Place,” in order to achieve a similar instantaneous effect. From the simulation scenarios and results on the right side, it is clear that this model will fire transition “t1” only in the right action sequence “F=true → E=true → t1 fire.”

From the above examples, we can have a general idea of an even-driven transition. It relies on both the condition places and the event place. However, in real systems, condition changes may call a new event. Moreover, an action could be either new condition change or new event. So the property (event, condition, action) of different system processes should be clearly defined. All the possible types we may use in fixed installations are summarized in Table 7.5.

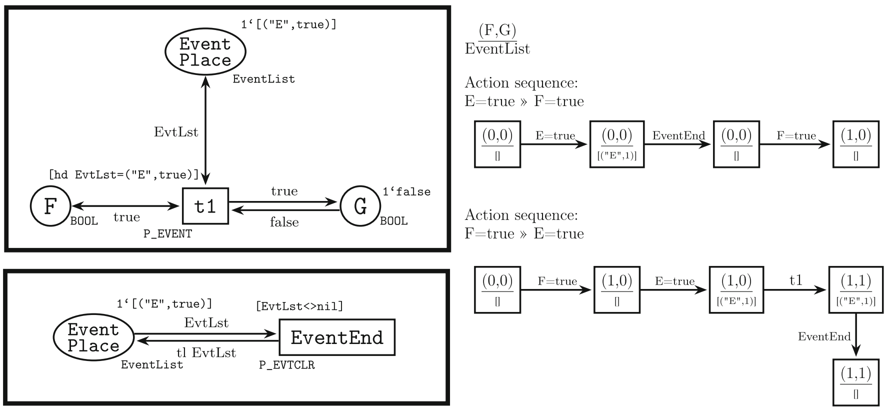


Fig. 7.26 Event-driven colored Petri net model of Fig. 7.22

Table 7.5 Type of logical variables ant its properties

Type	Description	Event	Condition	Action
Control	The status of relay (or switch)	X	X	
Command	The output command of relay	X		X
Indicator	Internal variable	X	X	X
Message	Command send by controlling center	X		
Action	Command or data send back to controlling center			X

7.4.5.3 System Validation of Event-Based Model

The final aim is to verify whether the system specification will hold the safety properties. Standard model checking algorithms are based on an exhaustive visit to all the reachable states of the specification. In our study, we chose CPN tools that integrate a powerful state space tool. It could generate the full state space of the PNs mode, and it could analyze the state space by means of a CTL-like temporal logic that allows user-defined searches and queries.

Model checking relies on the simulation environment. It determines which scenarios are going to be simulated and how each of the scenarios will be simulated. In each case study, we consider the original system to be a multi-input multi-output module. To be able to check its entire property, a test layer is added to provide external input events and variables and allows them to vary freely. In the system priority aspect, the test layer has the lowest priority. The test layer can give a new external input, but only when the original system reaches a new steady state. This assumption is also consistent with real practice, where RIS is a relay-based computer-controlled system. It has a faster processing cycle than its external inputs, such as human instruction or train movements. So it is reasonable to have a test layer with the lowest priority to simulate external input.

Safety performance of the system specification is “Safety property holds in every reachable state” or “danger case never happens.” During the state exploring, if we meet an unsafe state, there is no need to exploit its successive states, because all post-states are potentially unsafe. With this selective exploring method, we can reduce the state space without loss of reliability of safety analysis. So, before starting the state space calculation, we use the safety properties to specify that, under certain circumstances (system not safe), the CPN tools do not need to calculate all the successors of a state.

Normally, after a state exploring, we will get a large number of states and their marking information. A lot of them are internal states caused by subsystems. From the perspective of the safety analysis, we are more interested in a concise state space and system counterexamples. So, we make our own queries (ML functions) to search for all the “event-steady” states and the unsafe states, to generate an event-based state space tree, and to list the event paths of all the counterexamples.

System Modelling

To illustrate a complete practical use, a model of RIS in Fig. 7.20 will be demonstrated. This case study is very simple in that it only contains one point and two interlocking routes. The signaling operations in this model are to send commands to establish or destroy an interlocking route. A reasonable modelling structure and its simulation environment are shown in Fig. 7.27.

It should be noted that in order to better illustrate the analysis capabilities, we need an imperfect system model. So, when modelling the signal system, we

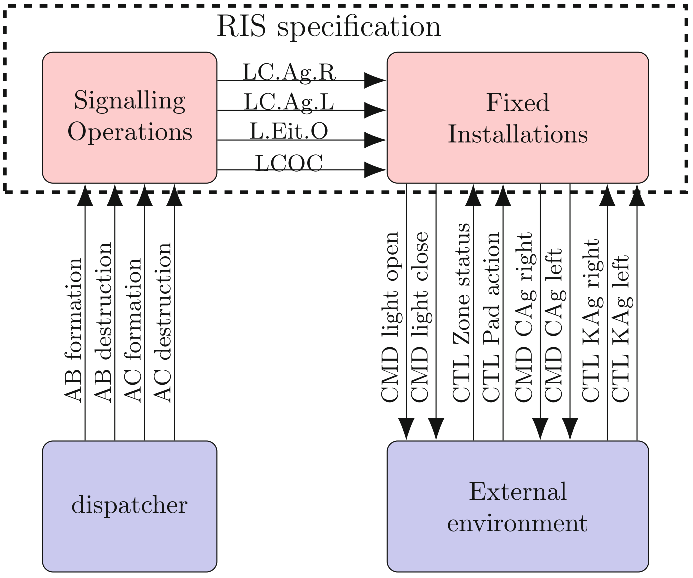


Fig. 7.27 Modelling structure and simulation environment

deliberately ignore a condition that is "System needs to wait 150ms, before sending command to switch on the signal light." Then, we get a potentially unsafe system.

The first part of the RIS model is the signaling operations as we discussed in Sect. 4.3.1. There is a simplified version of it in Fig. 7.28a, which contains different route phases (unformed, permitted, formed, etc.) and the corresponding transitions. Figure 7.28b is also a simplified version of route formation. As the signaling operations have been discussed before, considering the space restrictions, other sub-models of signaling operations are not shown here. The events in this model are defined in the form of (*Event type, Event name, value*), for example, the event to form the route "AB" is (*MSG, "AB " , form*). The event-trigger function is *fun EV :* $Eventlist * Event_x -> BOOL$. It is the guard function of event-related transitions and will return true if the $Event_x$ is at the top of the $Eventlist$.

The point control (in Fig. 7.29) contains two parts: 1. The point layer that could change the point's logical position by route command, interlock, or release point by shared resources and send commands to the point machine to change the rail connection (as shown in dashed line). 2. The transition layer is the necessary condition of route formation in flexible transition mode of the French context. The function "*gEV* " is a multi-event condition for transitions, which means any of the following events will enable this transition.

The final RIS layer is the signal light control (in Fig. 7.30) that could switch on signal lights if a route is established and the front zone is unoccupied. If the route is destroyed or if the front zone is occupied or if the point machine is not well positioned, then the light is switched off.

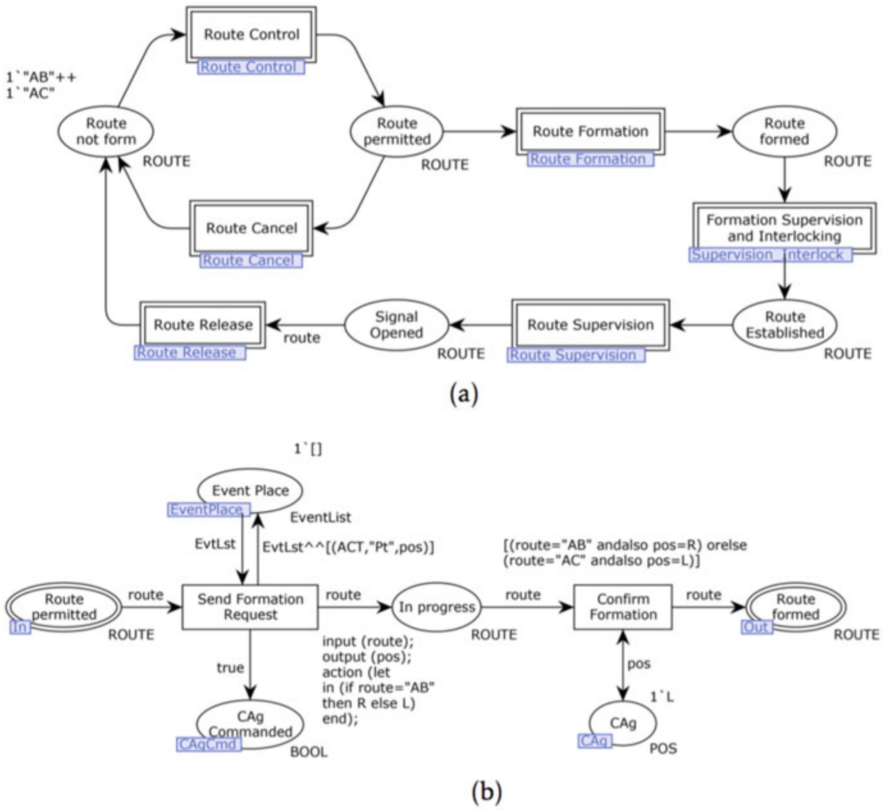


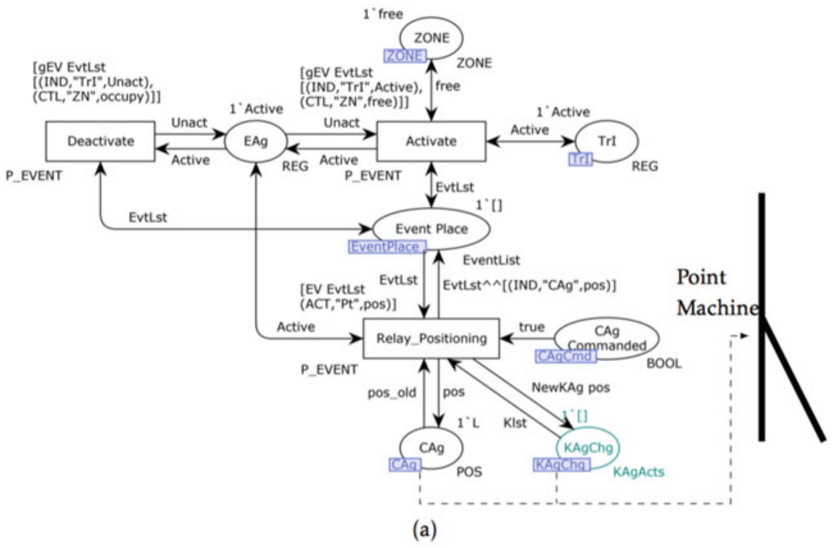
Fig. 7.28 Colored Petri net model of signaling operations. (**a**) Colored Petri net model of signaling operations layer. (**b**) Colored Petri net model of route formation

For model checking purposes, we need to add a test layer to simulate all the external input events in Fig. 7.27, and allow those events to vary freely. In this mode, the considered external inputs are route command (formation/destruction), zone occupation, pedal action, and point machine status *KAg*. (If a point is positioned to the right side, then relay *KAgR=true* else *KAgR=false*.) The outputs are signal light status and point machine command *CAg*. The model of simulation environment is shown in Fig. 7.31.

State Space Analysis

The safety statements of this system are:

φ_1: If any route is formed or zone is occupied, the relay CAg that controls the point cannot change.



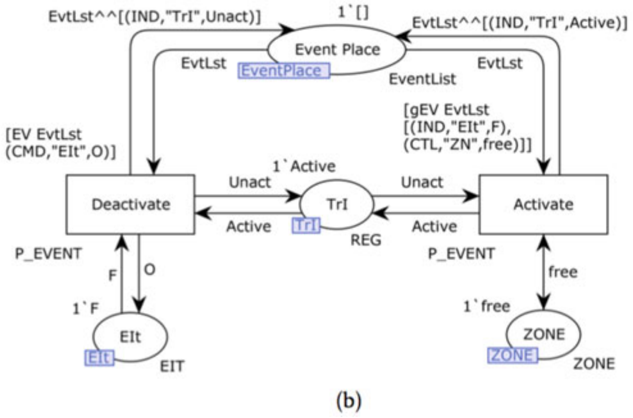


Fig. 7.29 Colored Petri net model of point control. (**a**) Colored Petri net of point layer. (**b**) Colored Petri net of transit layer

φ_2: If no route is formed or zone is occupied, signal light cannot be switched on.
φ_3: If the zone is occupied, the point machine must not act.

The selective branching option for exploiting the state space is designed as $\varphi_1(S) \wedge \varphi_2(S) \wedge \varphi_3(S) \rightarrow BOOL$, if the function returns *false* the state S will be a terminal state. With the result, we can start queries to examine if the state space will break any safety statements. The simulation result is shown in the second column of Table 7.6.

Although the size of the system state space has been simplified, it is still not readable. Moreover, too much information on CPN marking makes it difficult for

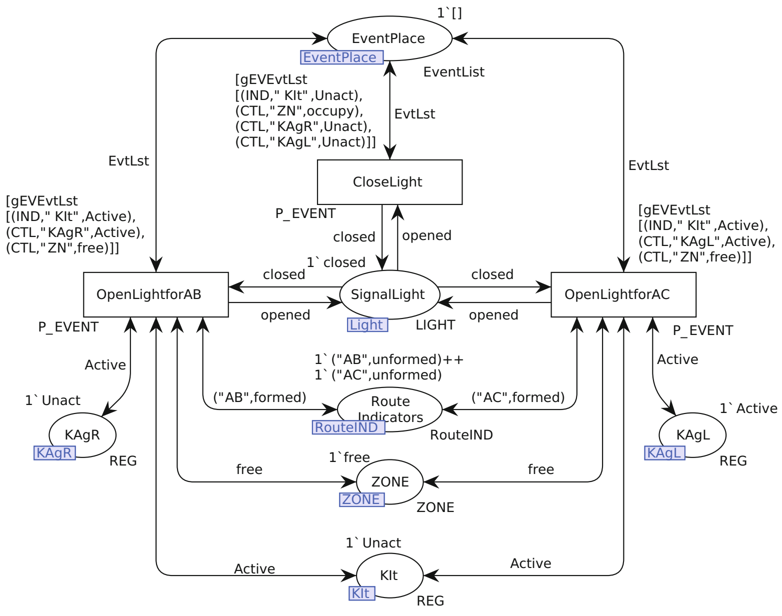

EventPlace
1`[]
EventPlace
EventList
EvtLst
[gEVEvtLst [(IND," KIt" ,Unact), (CTL,"ZN",occupy), (CTL,"KAgR",Unact), (CTL,"KAgL",Unact)]]
CloseLight
EvtLst
EvtLst
[gEVEvtLst [(IND," KIt" ,Active), (CTL,"KAgR",Active), (CTL,"ZN",free)]]
[gEVEvtLst [(IND," KIt" ,Active), (CTL,"KAgL",Active), (CTL,"ZN",free)]]
P_EVENT
closed
opened
1` closed
OpenLightforAB
SignalLight
OpenLightforAC
Light
LIGHT
P_EVENT
Active
1` ("AB",unformed)++ 1` ("AC",unformed)
("AB",formed)
("AC",formed)
Route Indicators
RouteIND
1` Unact
KAgR
REG
1` Active
KAgL
REG
free
1`free
ZONE
ZONE
1` Unact
KIt
Active
REG

Fig. 7.30 Colored Petri net model of signal light control

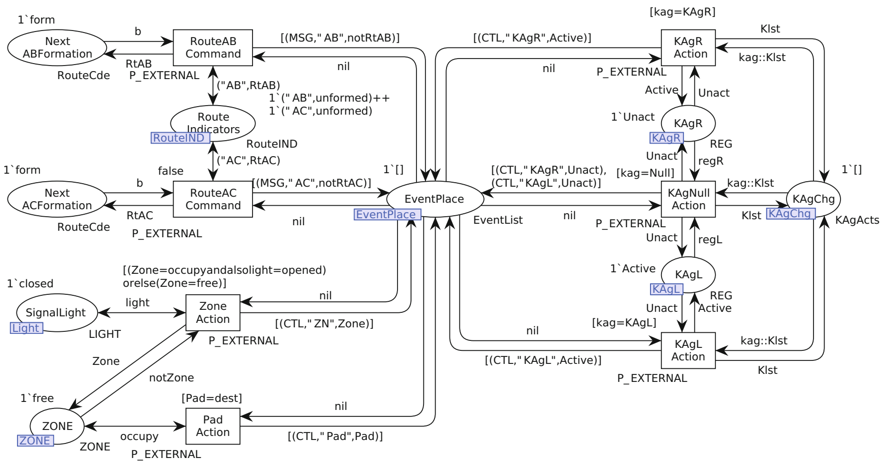


Fig. 7.31 Colored Petri net model of test layer

humans to compare each state. So another query is needed to transform the original state space into a more compatible form. Only event-free states and unsafe states will appear in the new state space. The original paths between each new state will be replaced by an external input event. So the new state space is an event-state graph, where each input event leads the system to a new state. Each of the new states

Table 7.6 State space calculation result

Exploring type	Default	Selective	Vectorization
State space size	366	301	76
Statement φ_1	Holds	Holds	Holds
Statement φ_2	Holds	Holds	Holds
Statement φ_3	Not holds (48 states)	Not holds (48 states)	Not holds (48 states)

is represented by a vector, $S_i = [A, B, C, D, E/F, G]$, each variable represents either a layer status or a relay value, and here, S_i *.= [Route progress, CAg, EAg, Tri, light / Zone, KAgR]*. The new state space graph has a total number of 76 states, where 29 are duplicates and 6 are danger states (third column of Table 7.6). Part of the graph is shown in Fig. 7.32, where the node in grey dashed style is the state already visited (duplicates) and the red node is the danger state.

The counterexamples of the verification are generated by giving the paths from initial state to each danger state. There are six paths in this example:

- Init → rAB=1 → LcAgR=1 → L.Eit=1 → L.Kit=1 → KAgL=0 → rAB=0 → KAgR=1 → rAB=1 → LcAgR=1 → L.Eit=1 → L.Kit=1 → Zon=0 → KAgR=0
- ... → L.Kit=1 → Zon=0 → rAB=0 → KAgR=0
- ... L.Kit=1 → Zon=0 → Pad=1 → KAgR=0
- ... → L.Kit=1 → Zon=0 → Zon=1 → Zon=0 →KAgR=0
- ... → L.Kit=1 → Zon=0 → Zon=1 → Zon=0 → rAB=0 → KAgR=0
- ... → L.Kit=1 → Zon=0 → Zon=1 → Zon=0 → Pad=1 → KAgR=0.

All of the counterexamples violate the statement φ_3. The reason for this danger situation is that when a new command is sent from RIS to point machine, its feedback *KAg* will take some time. If the RIS does not wait for new *KAg* data and continue to perform subsequent processing programs, then the old *KAg* data may lead the RIS to switch on the signal light and allow the train to enter while the point machine is going to change the point's position. So we get a dangerous state. The point position is changing, but there is a train in this zone and this will probably cause derailment.

System Specification Improvement

The solution to this fault is to add a time constraint to the RIS route establishing process. When the logical position of point *CAg* is changed and the front light is not yet switched on, the RIS waits for a moment, which is longer than the operation cycle of a point machine, thereby ensuring that all the actions of the point machine will be accomplished before the light switches on.

After we applied this new constraint to the model and analyzed its safety property, it turns out that the new system holds all the safety statements for every state. The new model has 259 original states in CPN tools' state space calculation,

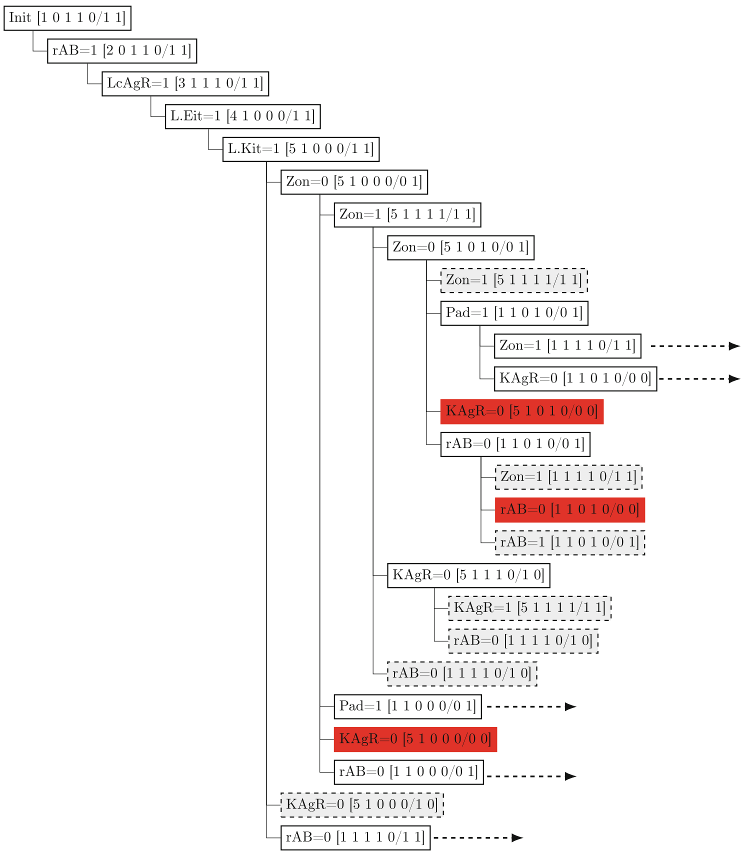


Fig. 7.32 Part of the state space tree

while after state abstraction, it has 65 states where 25 are duplicates, no danger state and no counterexample.

7.5 Conclusion and Perspectives

7.5.1 Conclusion

This chapter has been devoted to the model-based system engineering for safety of railway interlocking system. It provides a new approach via formal languages that aims to aid designers in effectively ensuring railway safety and improving the quality with system design and verification in railway industry. The study has focused on the formal modelling of French railway interlocking system. The nature and its formal specifications of RIS have been studied. A hierarchical modelling framework is proposed via CPN to specify and verify properties and behaviors of the RISs. The work has been presented as follows.

Due to various reasons, the knowledge of railway is partly written in textual documents and partly unwritten while owned by engineers. So in the system design or development process, we always need the assistance and supervision of expert engineers who have got both the written and unwritten knowledge. Initially, a quick comparison of GRAFCET and CPN is given, which illustrates their similarity and the ability to be seamlessly converted. So CPN has been chosen as our formal specification language. Its hierarchy and color features make it possible to propose a generic and compact structure that contains all high-level functions of RIS. At the same time, PN's rigorous semantics allow us to implement formal proofs.

The RIS is one of the crucial parts of the railway transit safety. In the French railway domain, the computer-controlled relay-based RIS (PRCI type) is the dominant practice. Its complex sequences and consequent actions make it difficult to be verified and validated. For such systems, first we analysis the architecture of RIS and the hierarchical structure of modelling framework. After that, we introduce an intuitive modelling approach that could formally specify the constructions of the fixed installations and the signaling operations of the interlocking logic. As a multi-input multi-output system, the signaling part of RIS is suitable to be modelled in a vertical decomposition way. It should contain different aspects, including components, scenarios, and functions. The fixed installation part is represented by logical objects connected to each other in the form of the track layout. It is natural for us to model it in a geographic way. However, in practice, each station or yard in a single line has its own RIS, which respects the same national standards but has different facility layouts. Normally, to specify all the stations, we have to rebuild models. It has low efficiency and will probably introduce new errors during the rebuilding process. With the modelling power of CPN, a general solution is proposed by introducing a modelling pattern, which could be easily adapted to different stations with PRCI type RIS. It is a general solution in a parameterized form. The "place/transition" structure (unmarked CPN model) represents a set of RIS functional rules. The logical formation of railway layout (configuration) is represented by the information contained in tokens. Besides, models that are less compact can be derived from this generic one in order to validate various aspects while keeping the safety property. Finally, we analysis the low-level part of the RIS that is the relay-based logic circuits. The relay-based circuit components have the nature of concurrency. An event-based concept is introduced to better describe these internal interactions. All the relay-based transitions (actions) are supervised by an "event place," and different transition priorities realize their relative synchronization. Furthermore, this event-based model is compatible with the classic CPN.

7.5.2 Perspectives

7.5.2.1 Transformation from CPN to *B machine*

Since the formalism of Petri nets has the advantage of communicating and their models could be validated by some engineering experts, it is still a long distance from the final implementation. To bridge the gap between the specifications and the implementations, we carry out another study—a model transformation from colored Petri net to ***B*** language, which could help people to quickly shift from a valid design solution to a valid input of ***B*** development process in the design phase. Detailed references can be found in Bon and Collart-Dutilleul (2013), Sun et al. (2015), Sun (2015).

The ***B*** method can offer a formal software development. In the French railway context, the ***B*** method is industry recognized tool and already has some success implementations, such as Météor (Behm et al. 1999), the new metro line number 14 in Paris. These successful engineering stories convince people of the reliability of the ***B*** method because the final implementation code generated from abstract ***B*** machine is considered safe and is proved to be safe. So in the French railway context, ***B*** proved model is accepted as a strong safety proof (Boulanger 2013a,b).

In our study, after mapping colored Petri nets (CPNs) formalism into ***B*** language formalism, the transformed ***B*** machines will be the input of the ***B*** development process and could be automatically refined into the implementable codes. Moreover, considering the limitations of model checking, sometimes we want to apply a theorem-proving for the purpose of verification. As the ***B*** proved models are considered “safe” in French industry, the transformation from Petri net to ***B*** machine is needed by any means necessary.

In the transformation framework, we maintain the mechanism of multi-set behaviors, and the transformed machines can be automatically proved by Atelier ***B*** tool. Besides, we propose some mapping rules for different color sets, in favor of raising the compatibility.

Furthermore, the concept of hierarchy is integrated into the mapping process. A multi-system that is modelled in a hierarchical way can be translated into a set of abstract B machines. The hierarchy is expressed by the composition relations of the machines and the accessible operations. Then, the concept of prioritized transition is introduced into the transformation. It is achieved by giving each operation a priority and adding an operation to the machine for priority management. It maintains the same priority mechanism of as in Petri nets.

7.5.2.2 Transformation from UML to CPN

Nowadays, UML is considered to be the standardized language for object-oriented modelling and analysis. However, UML cannot be used for automatic analyses and simulation. In Kerkouche et al. (2010), they propose an approach for transforming

UML state chart and collaboration diagrams to colored Petri net models. It produces highly structured, graphical, and rigorously analyzable models that facilitate early detection of errors such as deadlock and livelock. This transformation helps to bridge the gap between informal notation (UML diagrams) and more formal notation (colored Petri net models) for analysis purposes.

All the model transformations above along with the formal modelling of RIS aim to contribute toward a global safe analysis framework.

References

Antoni, M. (2009a). Formal validation method and tools for French computerized railway interlocking systems. *International Journal of Railway, 2*(3), 99–106.

Antoni, M. (2009b). Formal validation method for computerized railway interlocking systems. In *International Conference on Computers Industrial Engineering, CIE 2009*, pp. 1532–1541.

Antoni, M. (2009c). Validation d'automatismes ferroviaires de sécurité à base de réseaux de Petri. Ph.D. thesis. Braunschweig, Allemagne: Technischen Universität Carolo-Wilhelmina zu Braunschweig.

Antoni, M. (2012a). Formal validation method and tools for computerized interlocking system. In *FM Industry Day*, pp. 1–44.

Antoni, M. (2012b). Méthode de validation formelle d'un poste d'aiguillage informatique. *Recherche Transports Sécurité, 28*(2), 101–118.

Antoni, M., & Ammad, N. (2007). Feasibility study for the implementation of a formal proof of interpretable specification (for an interlocking system). In *FORMS/FORMAT 2007, Formal Methods for Automation and Safety in Railway and Automotive Systems*, Braunschweig.

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*, pp. 1–10.

Bacherini, S., Fantechi, A., Tempestini, M., & Zingoni, N. (2006). A story about formal methods adoption by a railway signaling manufacturer. *FM 2006, Formal Methods* (pp. 179–189). Berlin, Heidelberg: Springer Berlin Heidelberg.

Banci, M., Fantechi, A., & Gnesi, S. (2004). The role of formal methods in developing a distributed railway interlocking system. In *FORM-S/FORMAT 2004*, pp. 220–230.

Behm, P., Benoit, P., Faivre, A., & Meynadier, J.-M. (1999). Météor: a successful application of B in a large project. *Petri nets: Central models and their properties* (pp. 369–387). Berlin, Heidelberg: Springer Berlin Heidelberg.

Bon, P., & Dutilleul, S.C. (2013). From a solution model to a B model for verification of safety properties. *Journal of Universal Computer Science, 19*(1), 2–24.

Bon, P., Collart-Dutilleul, S., & Sun, P. (2013). Study of implementation of ERTMS with respect to French national rules using a B centred methodology. In *Industrial Engineering and Systems Management (IESM 2013)*, pp. 1–5.

Boulanger, J.-L. (2013a). Formal methods: industrial use from model to the code. *ISTE*. Wiley.

Boulanger, J.-L. (2013b). Industrial use of formal methods: formal verification. *ISTE*. Wiley.

Bjørner, D. (2003). New results and trends in formal techniques & tools for the development of software for transportation systems – a review. In *Formal Methods for Railway Operation and Control Systems (FORMS03)*, pp. 1–20.

Bjørk, J. (2006). Executing large scale colored Petri nets by using Maude. Ph.D. thesis. Oslo, Norway: University of Oslo.

Buchheit, G., Malassé, O., Brinzei, N., Lalouette, J., Walter, M., et al. (2011). évaluation des performances d'un axe ferroviaire en fonction des caractéristiques fiabilistes de ses systèmes de signalisations. In *Qualita 2011, 9ème Congrès international pluridisciplinaire qualité et sûreté de fonctionnement*.

Chen, L., Ning, B., & Xu, T. (2007). Research on modeling and simulation of vehicle-on-board automatic train protection subsystem of communication based train control system. In *ICVES 2007, IEEE International Conference on Vehicular Electronics and Safety*, pp. 1–5.

Cheng, Y.-H., & Yang, L.-A. (2009). A fuzzy Petri nets approach for railway traffic control in case of abnormality: evidence from Taiwan railway system. *Expert Systems with Applications, 36*(4), 8040–8048.

Collart-Dutilleul, S., Bon, P., El-Koursi, E., & Lemaire, é. (2014). Study of the implementation of ERTMS with respect to French national on board rules using a collaborative methodology based on formal methods and simulation. In *TRA 2014, 5th Transport Research Arena 2014*, Paris, France.

Fantechi, A. (2012). The role of formal methods in software development for railway applications. In *Railway Safety, Reliability and Security: Technologies and System Engineering* (chapter 12), pp. 282–297.

Fantechi, A. (2014). Twenty-five years of formal methods and railways: what next? *Software engineering and formal methods* (pp. 167–183). Cham: Springer International Publishing.

Fantechi, A., Flammini, F., & Gnesi, S. (2014). Formal methods for railway control systems. *International Journal on Software Tools for Technology Transfer, 16*(6), 643–646.

Fantechi, A., Fokkink, W., & Morzenti, A. (2012). Some trends in formal methods applications to railway signaling. *Formal methods for industrial critical systems* (pp. 61–84). Hoboken, NJ, USA: John Wiley & Sons, Inc.

Fanti, M.P., Giua, A., & Seatzu, C. (2006). Monitor design for colored Petri nets: an application to deadlock prevention in railway networks. *Control Engineering Practice, 14*(10), 1231–1247.

Ghazel, M. (2009). Using stochastic Petri nets for level-crossing collision risk assessment. *IEEE Transactions on Intelligent Transportation Systems, 10*(4), 668–677.

Giua, A., & DiCesare, F. (1993). GRAFCET and Petri nets in manufacturing. *Intelligent manufacturing* (pp. 153–176). London: Springer London.

Giua, A., & Seatzu, C. (2008). Modeling and supervisory control of railway networks using Petri nets. *Automation Science and Engineering, 5*(3), 431–445.

Buchheit, G., Malassé, O., Brinzei, N., & Lalouette, J. (2010). Évaluation des performances d'un axe ferroviaire en fonction des caractéristiques fiabilistes de ses systèmes de signalisations. In *9ème Congrès International Pluridisciplinaire Qualitéet Sûreté de Fonctionnement, Qualita'2011*.

Hagalisletto, A.M., Bjørk, J., Yu, I.C., Yu, I.C., & Enger, P. (2007). Constructing and refining large-scale railway models represented by Petri nets. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, 37*(4), 440–460.

Holloway, L., & Krogh, B. (1994). Controlled Petri nets: A tutorial survey. English. In G. Cohen, & J.-P. Quadrat (Eds.), *11th International Conference on Analysis and Optimization of Systems Discrete Event Systems* (vol. 199). Lecture notes in control and information sciences (pp. 158–168.). Berlin, Heidelberg: Springer.

Holloway, L., Krogh, B., & Giua, A. (1997b). A survey of Petri net methods for controlled discrete event systems. English. *Discrete Event Dynamic Systems, 7*(2), 151–190.

Huang, Y.-S., Weng, Y.-S., & Zhou, M. (2010). Critical scenarios and their identification in parallel railroad level crossing traffic control systems. *IEEE Transactions on Intelligent Transportation Systems, 11*(4), 968–977.

Janhsen, A., Lemmer, K., Meyer zu Hörste, M., & Schnieder, E. (1997). Migration strategy for different level of the European train control system to existing railway environment. In *Proceedings of World Congress of Railway Research, volume C: Power Supply, Signaling, Telecommunications and Non-conventional Systems*, Florence, pp. 101–118.

Jansen, L., Meyer Zu Hörste, M., & Schnieder, E. (1998). Technical issues in modelling the European train control system (ETCS) using coloured Petri nets and the design/CPN tools. In *Workshop on Practical Use of Coloured Petri Nets and Design/CPN* (pp. 103–115). Aarhus, Denmark: Citeseer.

Jensen, K. (1981). Coloured Petri nets and the invariant-method. *Theoretical Computer Science, 14*(3), 317–336.

Jensen, K. (1987). Coloured Petri nets. *Petri nets: central models and their properties* (pp. 248–299). Berlin, Heidelberg: Springer Berlin Heidelberg.

Kaakai, F., Hayat, S., & El Moudni, A. (2007). A hybrid Petri nets-based simulation model for evaluating the design of railway transit stations. *Simulation Modelling Practice and Theory, 15*(8), 935–969.

Kerkouche E, Chaoui, A.A., Bourennane, E.B., et al. (2010). A UML and colored Petri nets integrated modeling and analysis approach using graph transformation. *Journal of Object Technology, 9*(4), 25–43.

Lalouette, J., Caron, R., Scherb, F., Brinzei, N., Aubry, J.-F., Malassé, O., et al. (2010). évaluation des performances du système de signalisation ferroviaire européen superpose au système français, en présence de défaillances. In *Lambda-Mu 2010, 17e Congrès de Maîtrise des Risques et de Sûreté de Fonctionnement*.

Lei, L., Zhang, Y., Shen, X., Lin, C., & Zhong, Z. (2013). Performance analysis of device-to-device communications with dynamic interference using stochastic Petri nets. *IEEE Transactions on Wireless Communications, 12*(12), 6121–6141.

Moen, A., & Yu, I.C. (2004). Large scale construction of railroad models from specifications. In *IEEE International Conference on Systems, Man and Cybernetics*, pp. 6212–6219.

Pachl, J. (2002). *Railway operation and control*. VTSTD Rail Publishing.

Petri, C.A. (1966). *Communication with automata*, technical report RADC-TR-65–377 1 (2nd edn.). New York: Griffiss Air Force Base.

René, D., & Alla, H. (1992). *Petri nets and Grafcet: tools for modelling discrete event systems*. Prentice Hall (cit. on p. 56).

René, D., & Alla, H. (1997). Du grafcet aux réseaux de Petri. In *Ouvrage*. ISBN13: 978-2-86601-325 7.

Rétiveau, R. (1987). La signalisation ferroviaire. Presse de l'école nationale des Ponts et Chaussées.

Sun, P. (2015). Model based system engineering for safety of railway critical systems. Ph.D. thesis. Lille, France: école centrale de lille.

Sun, P., Collart-Dutilleul, S., & Bon, P. (2014). Formal modeling methodology of French railway interlocking system via HCPN. In *COMPRAIL 2014, International Conference on Railway Engineering Design and Optimization*, Rome, Italy.

Sun, P., Bon, P., & Collart-Dutilleul, S. (2015). A joint development of coloured Petri nets and B method in critical system. *Journal of Universal Computer Science, 21*(12), 1654–1683.

Sun, P., Collart-Dutilleul, S., & Bon, P. (2015). A model pattern of railway interlocking system by Petri nets. In *MT-ITS 2015, Models and Technologies for Intelligent Transportation Systems*, Budapest, Hungary.

Wang, F., & Bai, Z. (2010). Research for urban rail transit train regulation based on time Petri nets. In *CCTAE 2010, International Conference on Computer and Communication Technologies in Agriculture Engineering*, Chengdu, China, pp. 461–465.

Wu, N., & Zhou, M. (2004). Modeling and deadlock control of automated guided vehicle systems. *IEEE/ASME Transactions on Mechatronics, 9*(1), 50–57.

Xu, T., & Tang, T. (2007). The modeling and analysis of data communication system (DCS) in communication based train control (CBTC) with colored Petri nets. In *ISADS 2007, 8th International Symposium on Autonomous Decentralized Systems*, Sedona, AZ, pp. 83–92.

Yu, I.C. (2004). A layered approach to automatic construction of large scale Petri nets. Ph.D. thesis. Oslo, Norway: University of Oslo.

Zaytoon, J., & Villermain-Lecolier, G. (1999). Grafcet: methodological and formal issues. *Advances in manufacturing* (pp. 101–114). London: Springer London.

Zhu, L., Yu, F.R., Ning, B., & Tang, T. (2012). Service availability analysis in communication-based train control (CBTC) systems using WLANs. In *ICC 2012, IEEE International Conference on Communications*, Ottawa, ON, pp. 1383–1387.

Zimmermann, A., & Hommel, G. (2003). A train control system case study in model-based real time system design. In *IPDPS 2003, International Parallel and Distributed Processing Symposium*, 8 pp.