Model based system engineering for safety of railway critical systems

Pengfei Sun

N^o d'ordre : 275

École Centrale de Lille

THÈSE

présentée en vue
d'obtenir le grade de

DOCTEUR

en

Spécialité : Automatique Génie Informatique, Traitement du Signal et Images

par

Pengfei SUN

DOCTORAT DELIVRE PAR L'ÉCOLE CENTRALE DE LILLE

Titre de la thèse :

Ingénierie de modèle pour la sécurité des systèmes critiques ferroviaires

Model based system engineering for safety of railway critical systems

Soutenue le 24 juillet 2015 devant le jury d'examen :

Président	CR-HDR, Mohamed GHAZEL	IFSTTAR, Lille
Rapporteur	HDR, Hélène WAESELYNCK	LAAS-CNRS, Toulouse
Rapporteur	Pr, Hassane ALLA	UJF, Grenoble
Membre	Pr, Xiaoyun FENG	SWJTU, Chengdu
Membre	Pr, Jinling ZHU	SWJTU, Chengdu
Directeur de thèse	DR, Simon COLLART-DUTILLEUL	IFSTTAR, Lille
Co-directeur de thèse	CR, Philippe BON	IFSTTAR, Lille

Thèse préparée dans le Laboratoire d'Évaluation des Systèmes de Transports
Automatisés et de leur Sécurité
IFSTTAR, COSYS/ESTAS, Villeneuve d'Ascq

École Doctorale SPI 072 (EC Lille)
PRES Université Lille Nord de France

Je tiens à remercier tous
ceux qui m'ont soutenu et assisté
pendant ces années de thèse

À mes parents adorés

À mes professeurs

Et à mes chèr(e)s ami(e)s

Toute ma reconnaissance au professeur de français

Model based system engineering for safety of railway critical systems

Abstract

Development and application of formal languages are a long-standing challenge within the computer science domain. One particular challenge is the acceptance of industry. Although many successful stories have demonstrated the applicability of formal methods to industrial practice, their use within industry is still limited. This thesis presents some model-based methodologies for modelling and verification of the French railway interlocking systems (RIS), which aims to use formal methods to effectively ensure railway traffic safety.

This thesis mainly addresses two issues. The first one is the modellization of interlocking system by coloured Petri nets (CPNs). Next, a generic and compact modelling framework is introduced, in which the interlocking rules are modelled in a hierarchical structure while the railway layout is modelled in a geographical perspective. Then, a modelling pattern is presented. It is a parameterized model which respects the French national rules. It is a general reusable solution that can be applied in different stations. Then, an event-based concept is brought into the modelling process of low level part of RIS to have better description of internal interactions of low level relay-based interlocking logic.

The second issue is the transformation of coloured Petri nets into *B* machines, which can assist designers on the way from analysis to implementation. A mechanism describing the multi-sets and their behaviours in *B* machines is introduced to allow the following systematic translations. Firstly, a detailed mapping methodology from non-hierarchical CPNs to abstract *B* machine notations is presented. Then the hierarchy and the transition priority of CPNs are successively integrated into the mapping process, in order to enrich the adaptability of the transformation. This transformation is compatible with various types of colour sets and the transformed *B* machines can be automatically proved by Atelier B.

All these works at different levels contribute towards a global safe analysis framework.

Keywords: railway interlocking system, discrete event systems, CPN, modelling methodology

Ingénierie de modèle pour la sécurité des systèmes critiques ferroviaires

Résumé

Le développement et l'application des langages formels sont un défi à long terme pour la science informatique. Un enjeu particulier est l'acceptation par l'industrie. Malgré des succès industriels avérés, la dissémination des méthodes formelles dans la pratique industrielle ferroviaire est encore limitée. Cette thèse présente une approche pour la modélisation et la vérification des postes d'aiguillage français, qui utilise les méthodes formelles pour assurer efficacement la sécurité ferroviaire.

Cette thèse se concentre sur deux questions. La première est la modélisation du système d'enclenchement par les réseaux de Petri colorés (RdPC). Un cadre de modélisation générique et compact est introduit, dans lequel les règles d'enclenchement sont modélisées dans une structure hiérarchique, tandis que les installations sont modélisées dans une perspective géographique. Ensuite, un patron de modèle est présenté. C'est un modèle paramétré qui intègre les règles nationales françaises. C'est une solution générale et réutilisable qui peut être appliquée pour différentes gares. Puis, un concept basé sur l'événement est présenté dans le processus de modélisation des parties basses des postes d'aiguillage. Ce dernier fournit une meilleure description des interactions internes de la logique d'enclenchement à relais.

La deuxième question est la transformation des RdPCs en machines B, qui va aider les concepteurs sur la route de l'analyse à application. Des mécanismes décrivant les multi-ensembles et leurs comportements dans les machines *B* sont introduits. Il permettent la transformation systématique. Tout d'abord, une méthodologie détaillée, s'appuyant sur une table de correspondance, du RdPCs non-hiérarchiques vers les notations *B* est présentée. Ensuite, la hiérarchie et la priorité des transitions du RdPC sont successivement intégrées dans le processus de mapping, afin d'enrichir les possibilités de types de modèles en entrées de la transformation. Cette transformation est compatible avec différents types d'ensemble de couleurs, et la structure des machines *B* produites par la transformation permet la preuve automatique intégrale par l'Atelier B.

L'ensemble de ces travaux , chacun à leur niveau, contribuent à renforcer l'efficacité d'un cadre global d'analyse sécuritaire.

Mots clés : système d'enclenchement ferroviaire, systèmes à événements discrets, RdPC, méthode de modélisation

IFSTTAR, COSYS/ESTAS
20, rue Élisée Reclus – BP 70317 – F-59666 Villeneuve d'Ascq Cedex – France

Remerciements

Ces travaux ont été réalisés au sein de l'unité de recherche « Évaluation des Systèmes de Transports Automatisés et de leur Sécurité » dans le département « Composants et Systèmes » de l'« Institut Français des Sciences et Technologies des Transports, de l'Aménagement et des Réseaux » (IFSTTAR/COSYS-ESTAS) à Villeneuve d'Ascq, de novembre 2012 à juillet 2015. Je suis extrêmement reconnaissant à China Scholarship Council (CSC) qui m'a financé ce travail.

En premier lieu, je souhaite remercier M. Simon Collart-Dutilleul, directeur de recherche à ESTAS. En tant que directeur de thèse, il m'a accueilli au sein de son équipe et m'a orienté, aidé et conseillé. Je suis également reconnaissant de ses qualités pédagogiques et scientifiques, sa franchise et sa sympathie. J'ai beaucoup appris à ses côtés, et je le remercie pour la confiance dont il a su faire preuve à mon égard, notamment pour les recherches sur la méthode B. De plus, Il m'a également beaucoup aidé pour mes démarches personnelles pendant mon séjour en France. Pour toutes ses raisons, je lui adresse mes sincères remerciements.

Je souhaite aussi remercier vivement mon co-encadrant de thèse, M. Philipe BON, chargé de recherche à ESTAS, pour sa gentillesse et son soutien. Mes recherches sur la transformation de la méthode B sont basées sur sa thèse. Il m'a donné des conseils avisés et son écoute a été importante pour la bonne réussite de cette thèse. Son énergie et sa confiance ont été des éléments moteurs pour moi. J'ai pris un grand plaisir à travailler avec lui.

Je voudrais ensuite exprimer mes remerciements à mon ancien professeur Xiaoyun FENG, qui fut le premier à me faire découvrir le monde des chemins de fer, qui a soutenu mon étude de master avec des suggestions utiles, m'a impliqués dans ses projets de recherche et m'a aidé à saisir cette opportunité d'étudier en France. J'ai beaucoup appris de son attitude optimiste sur la vie et l'attitude rigoureuse lors de ses recherches.

Je voudrais également remercier les rapporteurs de cette thèse Mme. Hélène WAESELYNCK, habilitation à diriger des recherches au LAAS-CNRS de Toulouse, et M. Hassane ALLA, Professeur à l'Université Joseph Fourier, chercheur au laboratoire GIPSA-Lab de Grenoble, pour le temps passé et leurs remarques pertinentes. Je remercie aussi tous les

autres membres du jury pour avoir bien voulu participer à cette soutenance : M. Mohamed GHAZEL, directeur de recherche à ESTAS, et M. Jinling ZHU, professeur de l'Université Jiaotong du Sud-ouest en Chine.

Je remercie, pour la gentillesse et la convivialité dont ils ont fait preuve, toutes les personnes avec qui j'ai partagé mes études et notamment ces années de thèse au centre IFSTTAR de Villeneuve d'Ascq et qui ont rendu mon séjour très agréable. Je tiens à remercier particulièrement tous les doctorants et post-doctorants de l'IFSTTAR de Villeneuve d'Ascq : Liuliu, Baisi, Cyril, Antoine, Ciliang ainsi que l'équipe d'ESTAS : Sonia, Julie, Grégory, El-Miloudi, Nathalie, Valérie, pour leur soutien et toute l'aide qu'ils m'ont apporté lors de ma thèse. Je tiens à remercier particulièrement Rahma, qui m'a toujours gentiment donné les explications détaillées avec patience lorsque j'ai commencé à apprendre le langage formel B, ansi que ma camarade Liuliu, à qui je souhaite de finaliser le plus rapidement possible et dans les meilleures conditions ses travaux de thèse. Je lui souhaite une bonne continuation tout au long du chemin qu'elle chosira d'emprunter.

Toute ma gratitude va également à Mlle. Hélène CATSIAPIS, mon professeur de français à l'École Centrale de Lille, qui m'a montré la beauté de la langue et de la culture françaises. Elle a organisé des voyages intéressants et inoubliables qui ont inspiré mon intérêt pour la culture française et ouvert l'appétit pour l'histoire, l'art, la cuisine, et le vin. Tout cela a fortement contribué à mon bien être en France.

Dans ces moments importants, je pense tout particulièrement à ma famille. Tout ça n'aurait jamais été possible sans le soutien inconditionnel de mes parents qui ont toujours cru en moi. Merci pour avoir fait de moi ce que je suis à présent. Je les remercie pour leur soutien durant mon séjour à Lille et d'avoir accepter mon absence pendant trois ans. Je voudrais ensuite remercier ma compagne Huixin HAN dont l'amour et la présence en France m'ont permis de conclure ce voyage avec des sourires et des souvenirs colorés. Sa présence indéfectible auprès de moi dans les bons comme dans les mauvais jour m'a permis de garder un moral stable.

Enfin, merci à toutes les personnes, qui m'ont toujours soutenu et encouragé au cours de cette thèse, que je n'ai malheureusement pas citées ici mais qui se reconnaîtront à travers ces quelques lignes.

Scientific Acronyms

AMN

abstract machine notation. 7, 8, 102, 105, 110, 113–115, 142, 150, 155, 157

B

B method. 6, 7, 15, 16, 18, 19, 22, 23, 26, 27, 101–105, 109, 110, 112–121, 123, 125, 126, 129–131, 134–137, 145, 146, 149, 150, 153–155

CPN

coloured Petri net. 6–8, 19, 21, 24–26, 52, 57, 58, 61, 63–65, 68–70, 73, 74, 76, 80, 85–87, 97–99, 101, 102, 109, 110, 112, 114, 115, 117–119, 121, 123, 125, 127, 129, 130, 134–137, 141–143, 145, 146, 149, 150, 153–157, 187–190, 192

CtlPN

controlled Petri net. 87

DSL

domain specific language. 20, 21, 25, 26

Event-*B*

Event-*B*. 15, 22

FM

formal method. 7, 11, 15–24, 27

GRAFCET

GRAphe Fonctionnel de Commande Étape/Transition. 7, 53–57, 160

HCPN

hierarchical coloured Petri net. 5, 7, 8, 51, 57, 58, 61, 63, 67, 68, 76, 85, 102, 129–131, 137, 150, 155, 156, 187, 190–192

MDE

model-driven engineering. 1, 17, 18, 25, 26

ML

Meta language. 57

PN

Petri net. 5–7, 18, 21, 24–27, 52, 53, 55–58, 63, 64, 67, 73, 77, 87, 92, 101, 102, 109, 110, 146, 153, 154, 192

RdP

réseau de Petri. 160, 161, 163, 167

RdPC

réseau de Petri coloré. 159–167

RdPCH

réseau de Petri coloré hiérarchie. 162, 165, 167

RE

requirements engineering. 1

RG

reachability graph. 60

SCS

safety-critical system. 3, 4, 6, 7, 11, 13, 14, 16–18, 27

SE

systems engineering. 7, 12, 13

SSG

state space graph. 60, 62, 65

TL

transformation language. 25, 26

V&V

verification and validation. 3

Railway Terms

ATC

automatic train control. 19

ATO

automatic train operation. 19, 22

ATP

automatic train protection. 19, 30

CAg

point control relay (french: le relais de **C**ommande des **Aig**uilles). 45

CIt

route control relay (french: le relais de **C**ommande de l'**It**inéraire). 45

EIt

entrance signal relay of interlocked route (french: le relais d'**E**nclenchement d'**It**inéraire du signal d'entrée). 45

ERTMS

European Rail Traffic Management System. 2–5, 51, 52, 159

ETCS

European Train Control System. 19, 20, 25, 68

KAg

point checking relay (french: le **Contrôle** impératif des **Aig**uilles). 45

MA

movement authority. 60–62

PRCI

relay-based computer-controlled control (french: **P**oste d'aiguillage à **R**elais à **C**ommande **I**nformatique). 45, 52, 160, 166

PRG

relay-based geographic cable signalling control (french: **P**oste d'aiguillage tout **R**elais à câblage **G**éographique). 45

PRS

relay-based flexible signalling control (french: **P**oste d'aiguillage tout **R**elais à transit **S**ouple). 45

RIS

railway interlocking system. 3–5, 7, 19–21, 23, 25, 42–44, 51, 52, 66–68, 73, 76, 77, 80, 83, 86, 92–95, 98, 153, 154

RIt

route repeater relay (French: le relais **R**épétiteur d'**It**inéraire). 45

RRV

track circuit relay (french: **R**épétiteur du **R**elais de **V**oie). 45

SEF

systèmes d'enclenchement ferroviaires. 159–162, 166

SIL

safety integrity level. 15, 19, 23, 33, 44, 45

TER

regional express transport (french: **T**ransport **E**xpress **R**égional). 63

TGV

high-speed train (French: **T**rain à **G**rande **V**itesse). 63

TMS

train management system. 19

Tr.I

odd transit relay (french: le relais de **Tr**ansit **I**mpair). 45

Tr.P

even transit relay (french: le relais de **Tr**ansit **P**air). 45

UIC

International Union of Railways (french: **U**nion **I**nternationale des **C**hemins de fer). 3, 20

Table of contents

List of Tables

List of Figures

List of Listings

List of Publications

Following are the list of publications that produced during the course of my PhD research.

Journals (Published or Submitted)

1. **Pengfei Sun**, Philippe Bon and Simon Collart-Dutilleul, "A Joint Development of Coloured Petri nets and B Method in Critical System". In:*Journal of Universal Computer Science*. (Submitted)

2. **Pengfei Sun**, Simon Collart-Dutilleul and Philippe Bon, "Formal validation of interlocking systems under national railway signalling". In:*European Transport Research Review*. (Submitted)

Conferences (Published or Submitted)

1. **Pengfei Sun**, Simon Collart-Dutilleul and Philippe Bon, "Model transformation from coloured Petri nets with prioritized transitions to B machines". *In:IESM 2015, International conference on Industrial Engineering and Systems Management*, Seville, Spain, October, 2015. (Accepted)

2. **Pengfei Sun**, Simon Collart-Dutilleul and Philippe Bon, "A Model Pattern of Railway Interlocking System by Petri Nets". In:*MT-ITS 2015, Models and Technologies for Intelligent Transportation Systems*, Budapest, Hungary, June, 2015.

3. **Pengfei Sun**, Simon Collart-Dutilleul and Philippe Bon, "Formal modelling methodology of French railway interlocking system via HCPN", In:*COMPRAIL 2014, International conference on Railway Engineering Design and Optimization*, Rome, Italy, June, 2014.

4. Shaobo Song, **Pengfei Sun**, Qingyuan Wang, "Optimization Of The Layout Of Neutral Sections Based On A Differential Evolution Algorithm", In:*COMPRAIL 2014, International conference on Railway Engineering Design and Optimization*, Rome, Italy, June, 2014.

5. **Pengfei Sun**, Simon Collart-Dutilleul and Philippe Bon, "Formal modelling methodology of French railway interlocking system via Hierarchical coloured Petri net". In:*TRA 2014, Transport Research Arena*, Paris, France, April, 2014. (poster)

6. Philippe Bon, Simon Collart-Dutilleul and **Pengfei Sun**, "Study of implementation of ERTMS with respect to french national rules using a B centred methodology". In:*IESM 2013, International conference on Industrial Engineering and Systems Management*, Rabat, Morocco, October, 2013.

7. Huixin Han, **Pengfei Sun** and Jie Wu, "Research on Optimizing Scheme of Neutral Section based on Genetic Algorithm". In:*CCC 2013*, *China Control Conference*, P3327-3333, Xi'an, China, July, 2013.

8. Jianwei Qu, **Pengfei Sun**, Qingyuan Wang and Xiaoyun Feng, "Optimization for High-speed Train Operation Considering Driving Strategies and Timetable". In:*CCC 2013*, *China Control Conference*, P8167-8171, Xi'an, China, July, 2013.

General introduction

Preamble

This doctoral dissertation presents a synthesis of research work, which was carried out as a fruit of my Ph.D. work (2012-2015) in the ESTAS[1] laboratory of the COSYS[2] department in the IFSTTAR[3] (Lille) research institute, under the supervision of Prof. Simon COLLART-DUTILLEUL and Philipe BON. This work was made possible through the co-financing of CSC[4] and ANR[5]. In fact, my Ph.D. work took place in the PERFECT[6] national level project. Moreover, this project is also supported by the French i-Trans[7] cluster that is financed by Université Lille Nord de France, RFF[8], SNCF[9], Alstom.

With the research background of Prof. Simon COLLART-DUTILLEUL in railway safety requirements engineering (RE) and the previous work of Philipe BON in model-driven engineering (MDE), the research presented in this thesis contributes to the model based system engineering for safety of railway critical systems. The intention of this research is to provide some new approaches to effectively ensure railway traffic safety.

Motivation

Single European Railway Area

The organisation of railway traffic, the governance of railway principles and standards use to be a national matter. The signalisations and systems have been validated at the national level where each country has its own "language" for railway and its own requirements for managing trains on its network. With the development of the railway electrification, trains are

[1] Évaluation des Systèmes de Transports Automatisés et de leur Sécurité (Evaluation of Automated Transport Systems and Their Safety Laboratory)

[2] COmposants et SYStèmes (Composants and Systems)

[3] Institut Français des Sciences et Technologies des Transports, de l'aménagement et des Réseaux (French Institute of Science and Technology for Transport, Spatial Planning, Development and Networks)

[4] China Scholarship Council

[5] Agence Nationale de la Recherche (French National Research Agency)

[6] Performing Enhanced Rail Formal Engineering Constraints Traceability

[7] Pôle de compétitivité i-Trans

[8] Réseau Ferré de France (French Rail Network)

[9] Société Nationale des Chemins de fer Français (French National Railway Company)

increasingly dependent on train control system. So far, there are over 20 kinds of signalling and speed control systems throughout the European Union. Every system is autonomous and non-interoperable with the others, so that each train used by a national railway company has to be equipped with at least one on-board system but sometimes more, just to be able to run safely within that one country. Meanwhile, the trains run in several countries have to carry all the on-board systems of these countries, which requests additional integration effort for engineering, increases the total costs for border-crossing traffic, and reduces the performance of rail transport. For example, the "Thalys", a high-speed railway between Paris, Brussels, Cologne and Amsterdam, is equipped with 7 different types of train control systems, which causes considerable costs. All these constrains restrain competition and impede the integration of the railway transport at a European level vis-a-vis road and air transport, which have benefited from the absence of such barriers.

With the frequent economic and cultural exchanges among the European Union, there is a growing demand for international rail traffic. Also, the high speed railways have the quality to shorten the travel time for international journeys. To put an end to such a segmentation situation, and to promote the European rail market for passengers and freight, a packet of solutions has been launched by the European Union to remove the technical barriers that were created historically by the development of the national railways. This innovation relates to the technical and operational harmonisation, called the interoperability fo the high speed network and the conventional network. Meanwhile, the European signalling system ERTMS (European Rail Traffic Management System) has been the subject of a merger of several research projects, in order to create a new generation of signalling and speed control systems. The main objective of these projects is to have a common, harmonised, and standardised management of rail traffic and signalling in Europe in order to have a seamless network at the European level. ERTMS functions are located partly on the ground and partly on board, in this way, the communications between the ground and trains are standardised. This brand new standard is easier to apply in the new lines, where wayside signalling 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 existing lines, there is an alternative "Mixed operation" solution. This is a strategy where the wayside signalling 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 organisational 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.

Every conventional signalling 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 experience of ERTMS, which means it is impossible to evaluate the new system in the traditional way. Meanwhile, the management of railway signalling in ERTMS is based on local rules pertaining to each country and not on global ones, which makes it difficult to evaluate the combined system in terms of safety. In order to ensure the safety of ERTMS deployment, any implementations before being put into use should have detailed verification and validation (V&V), especially the compatibility of ERTMS and conventional signalling standards. This V&V is related to a set of rules, technical and operational conditions which ensure that the essential safety requirements are met. So some new methodologies which are more systematic and formal need to be adopted. Thus, the strong motivation of my research aims at contributing to the validation and implementation of ERTMS.

Railway interlocking system

Normally, locomotives and the rolling stock weigh tons, and are hundreds of meters long, in the distributed environment, under multiple control. These facts mean the trains are always at risk of collision. Because they are running on fixed track, trains are not capable of steering away from another conflicting train as road vehicle do to avoid collision. Another reason is the train needs a quiet long distance for to fully stop, even further than the driver's sight distance. So, in the case of obstacles on railway track, the train cannot stop in time to avoid collision. To avoid collision and other conflicts there is a mechanism called railway interlocking system (RIS), which is a highly safe and environmentally critical system. Its malfunctioning could cause death or serious injury to people, large scale environmental damage or considerable economical loss. For example, the "Zoufftgen train collision"[BEA-TT and AET 2009], which occurred in October 2006, is an RIS accident that was related to human failure. This accident caused 6 death, 2 serious injuries and 14 minor injuries. The organization of the accident response needed around 100 militias, 150 firefighters, 50 emergency vehicles and 7 vehicle extrication units.

Thanks to the usage of computers in safety-critical systems (SCSs), which has increased the concerns about the safety in railways, and successively reduced the number of accidents between 2006 and 2013 (data source from UIC[10] Safety Database Report 2014 [UIC-ETF 2014]). However many of these concerns have been focused on the software component of the computer based complex systems. Normally, the software itself can non cause an accident, but it becomes safety critical when it is used to control potentially dangerous systems. Formal methods, which are based on mathematical foundation, can be applied for modelling such systems giving an increased confidence. According to the European com-

[10]UIC (International Union of Railways french: **U**nion **I**nternationale des **C**hemins de fer) is an international rail transport industry body.

mittee of Electro technical standardization [EN 50128 2011], the use of formal methods in the development of safety-critical and computer-controlled systems for railway applications is recommended.

Based on the above discussion, it was realized that the specification and analysis of railway RIS is an important part of the deployment of ERTMS. Moreover, the critical and complicated nature of railway safety properties obliged us to adopt the formal method for the RIS more rapidly. Thus, my main research approach and case studies are based on the RIS, using formal methods.

Problematic

This thesis is in the domain of safety requirement engineering in railways and more specifically around the usage of formal methods to assess and validate certain SCSs. It also discusses the approach of specification and modelling of complex systems. Undoubtedly, the increasing complexity of automated systems that are applied in the railway industry introduces in an increasing difficulty in assessing their structural and behavioural properties. We are interested in a large part of the design process, from the formalisation of requirements to integration, validation and verification.

Our research objects can be characterised as following:

- A privileged application domain: the safe and efficient operation of railway transport networks under certain signal standards.
- Scientific and technical foundation: Models and formal methods, taking into account the model transformations.
- A methodological context: the development of methods and tools for assisting the design of complex systems.

The aim of the ERTMS is to create an interoperable railway system within Europe in order to increase its own competitiveness. However, as a signalling system, its most important responsibility is to maintain transportation safety. During the implementation of ERTMS, evaluation of the global consistency is needed. Evaluation verifies 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, the difficulty of this problem is the lack of formal representations of both system, which could enable the validation of different aspects through test scenarios.

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, Collart-Dutilleul, and Sun 2013; Collart-

Dutilleul et al. 2014; Sun, Collart-Dutilleul, and Bon 2014]. My Ph.D. project is 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. The study aims to provide a methodology for a comprehensive assessment of the consistency of the following two aspects: the operating rules of local signalling systems and the additional safety requirements (ERTMS).

With this method, we are able to follow the safety analysis: The safety assessment of new systems, the analysis of given scenarios, 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.

Contribution

As discussed above, the overall aim in this thesis is to show and explore the hypothesis that the formal languages can improve the quality with system design and verification in railway industry. The primary result of this thesis is a systematic design methodology for modelling and safety verification of the French railway interlocking system (RIS). It provides a theoretical framework, including system specification and safety requirement verification by model checking and theorem proving.

The work that has been performed in this thesis can be split into two main results:

Formal modelization of railway interlocking system: The first result of the thesis is to provide a modelling framework allowing one to specify a RIS, formulated as hierarchical coloured Petri net (HCPN) models, for simulating and verification of their behaviours. In this framework, the Petri net's "place-transition" structures delineate the construction of the fixed installations and the signalling rules of interlocking control. The dynamic behaviours, such as signal commands and train movements, are represented by firing transitions and tokens.

On the practical side, each station or yard along a railway line has its own interlocking system, which respects the same national standard but has a different facility formation. In order to effectively accomplish the validation tasks and reduce the error probability, we introduce a modelling pattern based on the framework for different railway layouts. This pattern is a parametrized model that respects the French national rules (in this thesis, we means the signalling rules of relay-based computer-controlled systems). It is a general reusable solution to this kind of problem and can be used in many different given contexts.

The low-level part of the RIS — the relay-based logic, has a nature of concurrency. It is a mechanism that does not exist in the Petri net language. To solve this problem, we introduce

an event-based mechanism to simulate a relative concurrency, in order to be able to describe the internal interaction between each relay. Comparing to the original system, this mechanism brings "internal" states which will enlarge the system state space. To compensate for this deficiency, we apply a reduction policy, both before and after the state space calculation, to obtain a new compact graph with the same reliability for analysis.

Model transformation from coloured Petri net to *B* machines: 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. The second result of this thesis is to bridge the gap between the specifications and the implementations. As the *B* method offers a formal software development, this thesis introduces a systematic process of mapping coloured 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 this transformation framework, we improved the mechanism of multi-set behaviours based on the work [Bon and Dutilleul 2013], so that the transformed machines can be automatically proved by Atelier B tool. Besides, we propose some mapping rules for different colour sets, in favour of raising the compatibility.

In order to enrich the adaptability of the transformation, 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.

All these above works aims to contribute towards a global safe analysis framework.

Outline

This thesis is divided into 3 parts (6 chapters in total). The brief description of each chapter is as followed:

Part I

Chapter 1. *Background*: This chapter aims to present and locate the scientific background of this thesis. It provides an overview of the related fields which are based on the formal solution of modelling complex safety-critical systems. The first part of this chapter gives

a general introduction on concept and development of four disciplines, including systems engineering, safety-critical system, formal languages. In the rest of this chapter, some previous research and applications about formal method for safety-critical system in the railway domain are reviewed.

Chapter 2. *Some Principles of railway safety*: This chapter describes and formalises some major safety properties and the control logic of railway signalling system. For this purpose, an analysis has been performed step by step, which is organized around the safety nature of the signalling based railway system. Firstly, a particular concept of safety has been explained. Based on that, the background of safety management in France is introduced, and the problematic of the computer-controlled system is discussed. Next, the context and the characteristics of the railway are stated. Then, the composition structure of the railway system is stated. Major components, such as human actors, signalling procedures and installations are introduced. Further, the common risk form signalling system is summarized. Finally, the French railway interlocking system is introduced, including the general architecture and different operating phases of a signal centre.

Part II

Chapter 3. *Formal modelization for railway interlocking system via hierarchical coloured Petri net*: This chapter deals with modelling and analysis of the railway interlocking system. The proposed cases are studied and described by using HCPN. Initially, this chapter starts with the background of Petri nets, and presents a discussion on the GRAFCET and Petri nets, in order to state the reason for choosing Petri nets as formal tools. The graphic notations used in this thesis and the modelling ability of CPN are carried out with a series of demonstrations Then, the structure and the modelling framework of RIS are discussed. After that, an intuitive modelling approach is performed, which formally specifies the constructions of the railway network and the signalling procedures of the interlocking logic. Then, considering each station complies with same interlocking logic rules but with various railway layouts, a general solution is proposed by introducing a modelling pattern, which could be easily adapted to different stations. Finally, the analysis of the low-level system of RIS is stated. An event-based concept is introduced to better describe the internal interactions of relay-based logic.

Chapter 4. *Model transformation: from coloured Petri net to B language*: This chapter introduces the modelling transformation approach. First, it reviews the basics and the outstanding features of the *B* theory, including some notations, concepts and the mathematical syntax of *B* language. After, a systematic transformation framework is introduced, which is from non-hierarchical CPNs to abstract machine notations. In this framework, it improves the multi-set mechanism and its related specifications from [Bon and Dutilleul 2013], so that the transformed *B* machines can be automatically proved by Atelier B without using "interactive prover". Then, we discuss the solutions for hierarchy of abstract *B* machines

and present a new transformation from HCPNs to AMNs based on the previous one. A "multi-system" approach is be studied, and the hierarchical structures are presented by adding accessible pre-defined functions in those high-level *B* machines. Finally, the prioritized transition of CPN is considered, which could increase the capacity of describing the event-based systems. In new transformation, a global priority indicator is added, each operation is assigned with a priority pre-condition, a priority management mechanism is created.

Part III

Conclusions: Finally, we draw the thesis to a close by summarising all our proposed methodologies. We give concluding remarks on applying the methodology and comment on future areas of interest towards extending the methodology.

Part I

Preliminary

Chapter 1

Background

The aim of this chapter is to present and to locate the scientific background of my research. The subject is the modelling of complex systems considering safety-critical scope via formal languages. First, system engineering is presented as the basic concept of complex systems, then, we introduced a general solution of complex system modelling. Second, the definition and categories of safety-critical system are presented, along with the introduction of industry standards and a little analysis of its research domains. Third, the background of formal methods is described and a short discussion about its industry applications. Finally, a literature review is given on scientifically and technologically related topics.

1.1 Modelling of complex system

System modelling is an activity and its aim is to make a particular part or feature of the world easier to understand. It is essential for understanding, simulation and analysis of systems, and it provides a basic prototype for the design process. There are already a lot of techniques to represent a system in a mathematical form. Still, a model is an approximation in a simple format. It has to give an abstraction of reality according to some given established purpose.

But, when a system is getting more complicated, it typically consists of "*a number of elements having diverse, variable behaviours that interact to produce the behaviour of the whole*" [Bar-Yam 2003]. This kind of systems can be referred to as a complex system. Its elements operate according to Its own rules or schedules and are not all entrained by an overall timer. The notion of complexity presents analytical difficulties coming from their behaviours, size and natural properties. However, "*Complexity (ignoring the definitional debates) is subjective because there are many different ways of viewing the parts, describing interactions, and different ways of measuring complexity*" [Berryman and Campbell 2010]. Thus, a complex system cannot be reduced to a single model. Traditional engineering disciplines seem not so well fitted for such systems.

The object of our study is focused on the railway context. The railway systems are large-scale systems typically consisting of many concurrently operating and interacting human and non-human elements. These elements can produce a result or provide a service or set of services. In order to analyse this system, we propose to use the technique of systems engineering.

First, let us look at what is meant by systems engineering, and then consider how complex system approaches are settled by systems engineering as a field of research. Roughly speaking, "*systems engineering is the engineering of complex systems*" [Kossiakoff et al. 2011, p. 3]. The more rigorous forms are illustrated as the following representative definitions from the Federal Aviation Administration (FAA) and the International Council on Systems Engineering (INCOSE).

> "*Systems Engineering is a discipline that concentrates on the design and application of the whole (system) as distinct from the parts. It involves looking at a problem in its entirety, taking into account all the facets and all the variables and relating the social to the technical aspects.*" [FAA 2014, p. 303]

> "*Systems engineering is an interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem: operations, cost and schedule, performance, training and support, test, manufacturing, and disposal. SE considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs.*" [INCOSE 2012, p. 3]

Some keywords can be figured out from the two definitions: *interdisciplinary*, *iterative* and *wholeness*. It is apparent that such kinds of system description from systems engineering have many similar characteristics as the nature of complex systems. In other words, "*systems engineering is the engineering of complex systems*" [Abbott 2006, p. 1].

The technique of modelling is one of the basic tools of systems engineering, especially in the circumstances that explicit facts and quantitative information are not readily available. In such situations, a complex system can non be established as a single model. Thus, a model of a typical complex system could be constructed in term of constituent parts. Each constituent part explains only a part of the knowledge of the target system and masks other information. It is only a "sub-system" representation of a "super-system" in its environment of use. This formation may be called "system of system (SoS)". The purpose of this formation is to define a relatively concise and easily understandable system framework, which can serve as a point of reference for discussing the process of developing a new system and the role of systems engineering through the process.

By their nature, complex systems have a hierarchical structure in their constitution, which integrates a number of major interacting elements. These elements are generally called subsystem. Each subsystem is composed of more simple functional entities, and so on down to primitive elements. The definition and the division of architectural system Levels are confined to the generic system and its subsystem. We will discuss it later in this thesis based on a typical railway system.

After all, systems engineering is a large interdisciplinary field of engineering, and we are only interested in the modelling and analysis tools of the systems engineering. To be more precise, our target field is Safety Engineering, which aims to help non-specialist engineers in designing complex systems to minimize the probability of safety-critical failures.

1.2 Safety-critical system

"*Today, we live in a world in which our safety is more and more dependent on software-intensive systems*" [Larrucea, Combelles, and Favaro 2013]. This is the case for the aeronautic, automotive, medical, nuclear, and railway sectors, as well as many more. The shareholders of such systems are always seeking more economical way to deal with the enormous increase in size and complexity, while necessarily regarding the need to ensure system safety. Consequently, in the engineering domain, the safety of a system is always one of the important goals and core needs. Effort was spent on how to manage risk, eliminate or reduce it to give acceptable levels of safety.

An informal description of safety could be "Nothing bad happens". There is an ideal definition–"*Freedom from accidents or losses*" [Leveson 1995]. But it is so ideal that no system can be perfectly safe in an absolute sense. In order to be able to integrate the safety analysis into a brand new system, designers focus more on removing hazards than accident analysis, so there comes the modern concept of safety: "*System will not endanger human life or the environment*" [Storey 1996].

Safety-critical system (SCS), or so called life-critical system is a system which controls (or as part of the control) hazardous or safety-critical hardware or software. If such a system fails that could result in at least one of the following outcomes [NASA 2004; Wikipedia 2003]:

- Death or serious injury to people
- Loss or severe damage to equipment/property
- Environmental harm

Although we may find some other similar definitions of life-critical systems, sometimes an intuitive criteria will give us a better description. Both formal definition and intuitive description describe the SCS from the failure consequences. "*If the failure of a system could lead to consequences that are unacceptable, then the system is safety-critical*" [Knight 2002].

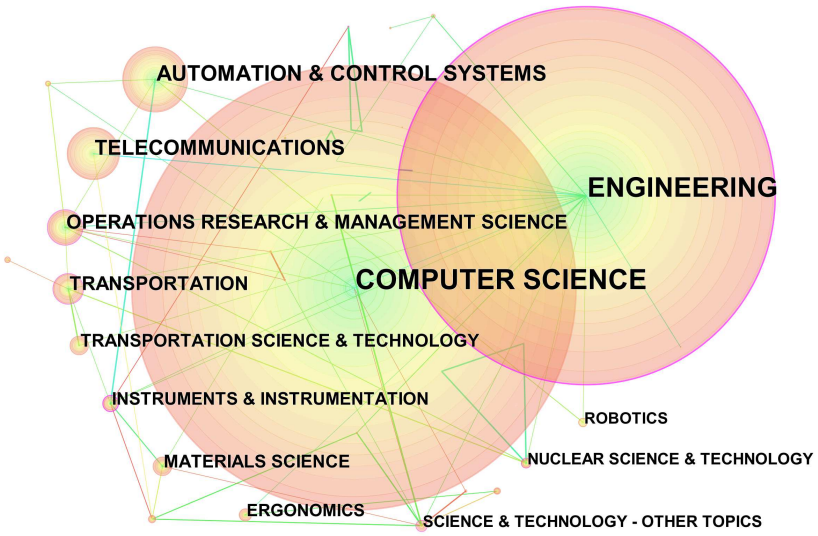


Figure 1.1 – Research domains[1] of safety-critical system

The common SCSs can be categorised into several domains as showed in Fig. 1.1. This figure is the analysis result of CiteSpace[2] [Chen 2006; Chen, Hu, et al. 2012]. The original data is 4,048 paper entries under the topic SCS from the Web of Science (WoS). Although our data source is not exhaustive, and only contains English publications from the WoS database, its analysis can still be used as a reference. There are 45 nodes (some are too small to observe), which represent different research categories. The size of the node depends on the number of papers and the cited index.

In Fig. 1.1, it is clear that there is a large amount of research that has been done on general computer science and general engineering. However, when it comes to domain-based fields, such as telecommunication and transportation, the number of studies significantly reduce. This figure may indicate that the basic theory and methodology of SCS applications in the general areas are mature, while the major difficulty to be applied to typical domains is lacking in industry knowledge. In other words, the development of domain-based SCS has great application prospects.

Some widely accepted industry standards can be found in these domains above, such as IEC 61508 [IEC 61508 2010], EN 50128 [EN 50128 2011], DO178B [DO 178B 2010]. By regulating or restricting the development processes, via verification and validation, these standards are intended to reduce the number of risks introduced by the process (e.g. using rigorous specifications), and to increase the number of risks revealed by the process (e.g.

[2]CiteSpace is a freely available Java application for visualizing and analysing trends and patterns in scientific literature.

applying rigorous verifications) in order that such risks can subsequently be removed.

Standards prescribe the degree of rigour required in development and assurance, according to the criticality of system application. The degree of rigour is typically expressed in terms of safety integrity levels (SILs), which is to determine the acceptable failure rate of the system. For example, in EN 50128 the requirement for SIL 4 is defined as: for continuously operated system, the equivalent failure probability rate range is $10^{-10}/h \sim 10^{-9}/h$ (≈ 1 failure/114150 years).

Besides the degrees, the standards also define the recommended techniques and processes for the development process and system assurance. For example, the guidance EN 50128 states that "Informal" requirements and design specification are considered acceptable for the lower integrity levels, i.e. SIL 1 and SIL 2. "Semi-formal" techniques are admissible for SIL 3, and "Formal (Mathematical)" specification techniques are expected for the highest level of integrity, i.e. SIL 4.

Although there are many standards and techniques to ensure safety, we still have to accept some of the failure, considering the limitation of cost, time and other resources. It is also a wisdom in system risk management — "fallacious expectation of 0 risk". There is no system that can be absolutely safe, and no system can guarantee zero risk.

1.3 Formal methods

In computer science, especially software engineering and hardware engineering, *"formal methods have existed at least as long as the term 'software engineering' itself"* [Gruner et al. 2013]. *"formal methods are mathematically based techniques and tools for the synthesis (i.e. development) and analysis of software systems"* [Haxthausen 2010]. For example, as a tool, the method should include a specification language. Then such a language has a formal syntax, a formal semantics and a formal proof system. *"The techniques of a formal method help to construct a specification, and/or analyse a specification, and/or transform (refine) one (or more) specification(s) into a program"* [Bjørner and Havelund 2014]. In an engineering context, using formal methods (FMs) means applying the design of technology based on scientific insight, and to assess its properties using some dedicated technology for the analyse phase.

After more than 40 years of research, now we have a wide range of FMs that are applied in different areas [Bjørner and Havelund 2014; Haxthausen 2010; You, Li, and Xia 2012]. The foremost ones can be classified as follows, according to their expression abilities.

Model-oriented specification languages that distinctly define states and operations definitely, such as VDM[3] [Bjørner 1978], Z[4] [Woodcock and Davies 1999], *B* method [Abrial 1996] and Event-*B* [Abrial 2010]. Their main focus is on the development of specifications.

[3] VDM: Vienna Development Method.
[4] Z: Zermelo.

Property-oriented specification languages, such as CASL[5] [Bidoit and Mosses 2004], Maude [Clavel et al. 2007], and CafeOBJ [Diaconescu and Futatsugi 1998].

Analysis methods that use analysis techniques, such as Model-Checking, for example SMV[6] [Clarke Jr, Grumberg, and Peled 1999], SPIN[7] [Holzmann 2003].

Logic based methods that explore system by chosen logic, such as ITL[8] [Umamageswaran, Pandey, and Wilsey 1999], DC[9] [Zhou and Hansen 2004] and RTTL[10] [Ostroff 1989].

Graphical methods that are in a graphic notation, such as Petri nets [Petri 1966], Statecharts and MSC[11] [ITU-T 2011]. They are easy to understand and use for some lay people.

Related formal notations that are a set of languages called "process algebras", such as CSP[12] [Hoare 1985; Schneider, Treharne, and Wehrheim 2012] and CCS[13] [Milner 1980].

Despite all these methods, and publications, right now there are still few FMs that have become a success. In our context, success means being widely accepted by industry or integrated into production processes. However, from the safety needs of industry, the FMs are worthwhile. Moreover, they will probably be indispensable in the professional pursuit of the development of SCSs. Therefore, the situation where FMs are not used, while fitted scientific methods are available, indicates the immaturity of the adaptation of this FM to the industrial.

Generally, there are two main technical obstacles of the practice of FMs. First, there is still not enough research and teaching of FMs, which restrict FMs within a scientific level. This leads to a lack of belief that FMs are worthwhile for the industry. The second difficulty is the supporting tools of FM are not sufficient for large scale use in the industry context. It is the fact that "*industry will not use an* FM *unless it is standardized and 'supported' by extensive tools*" [Bjørner and Havelund 2014]. These two obstacles can be considered as "maturity of engineering" nature. It may take years for the science of FM to precede industry-scale engineering.

Fortunately, in the French rail transport area, the FM has reached a level of maturity. For example, the *B* method has been used successfully to verify the most relevant parts of a metro system in Paris [Behm, Benoit, et al. 1999]. The related industry keeps an open mind on FMs, and they are willing to use them. This will definitely accelerate the process of engineering maturity of FMs. Therefore, my research is to devote an effort to this process, concerning the application of FMs for SCSs in the Railway industry.

[5]CASL: Common Algebraic Specification Language.
[6]SMV: Symbolic model verification
[7]SPIN: Simple Promela Interpreter
[8]ITL: Interval Time Logic
[9]DC: Duration Calculus
[10]RTTL: Real-time temporal Logic
[11]MSC: Message Sequence Charts
[12]CSP: Communicating Sequential Processes.
[13]CCS: Calculus of Communicating Systems.

1.4 Model-driven engineering

To increase the formal methods productivity, one obvious approach is to increase the level of automation all along the development process. This is one of the important features of MDE [Santiago et al. 2012; Schmidt 2006], which refers to the systematic use of models as the primary engineering artefacts [Seidewitz 2003] in a formal and precise way [Lukman and Mernik 2008]. MDE can be applied to software, system, and data engineering. *"The term* MDE *is typically used to describe software development approaches in which abstract models of software systems are created and systematically transformed to concrete implementations"* [France and Rumpe 2007]. It offers an environment which ensures the systematic and disciplined use of methods (including FMs) throughout the development process of the system, especially the software systems.

Douglas C Schmidt summarized that the promising approach to address system complexity is to develop FM technologies that combine the following[Schmidt 2006]:

- *Domain-specific modelling languages* which formalize the application structure, behaviour, and requirements within particular domains. Such languages are described using meta-models, which define the relationships among concepts in a domain and precisely specify the key semantics and constraints associated with these domain concepts.
- *Transformation engines and generators* that analyse certain aspects of models and then synthesize various types of artefacts. Such automated transformation processes are often referred to as "correct-by-construction," as opposed to conventional hand-crafted "construct-by-correction" processes that are tedious and error prone.

As we have discussed in the previous Section 1.3, the significant factor behind the obstacles of applying FMs to industry is the conceptual gap between scientific technology and the implementation domain. To bridge the gap, an extensive hand-crafting of implementations is required, which will make the development of SCSsdifficult and costly, and may lead to accidental complexities. *"MDE is primarily concerned with bridging the gap between problem and software implementation domains"* [Ahmad 2013] through using technologies that support systematic transformation of formal abstractions to final implementations. The difficulties of bridging the gap can be addressed through the concept we have discussed in Section 1.1, which describes the complex system in a hierarchical way with multiple levels of abstraction and from a variety of perspectives, then with automated support for processing and analysis the models, finally to automatically transform into implementation results.

A brief conclusion, in the MDE-aided system development, *"models are the primary artefacts of development and developers rely on computer-based technologies to transform models to running systems"* [France 2010]. Our envisioned solution is to put FM into MDE develop-

ment environment to solve the problem in modelling SCSsin the railway domain.

1.5 Literature review

After forty years of development, FMs are considered as the promising solution for the safety certification of railway system designers. This chapter is dedicated to the presentation of the application of FMs in the railway domain. As mentioned in the previous sections, FMs are useful in the research of SCSsin this domain. After a recall of some remarkable FMs, we have chosen two methods, Petri net and *B* method, which have already been accepted by the French railway industry to aid with the development of railway safety-critical systems (SCSs). Thus, in the literature reviews, previous research concerns FMs for SCSs in the railway domain, Petri net and *B* applications in France. Moreover, model transformation of Model-driven engineering (MDE) will be introduced in detail.

1.5.1 Formal method for safety-critical system in railway industry

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 system 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 behaviours 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 2012b; Fantechi, Fokkink, and Morzenti 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.

Traditionally, "*Railway signalling has been considered as one of the most fruitful areas of intervention by formal method*" [Fantechi, Fokkink, and Morzenti 2012]. It is because railway signalling 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 survey and reviews [Bacherini et al. 2006; Fantechi 2012b, 2014b; Fantechi, Flammini, and Gnesi 2014; Fantechi, Fokkink, and Morzenti 2012] focus on the advances in both formal method approaches and railway signalling applications. Still, lots of related work that has been performed by railway companies is not published because of confidentiality considerations.

For a long time, there have been many formal notations, methods and tools (most are prototype) that have been developed for academical propose. They lack industrial robust-

ness in terms of stable tools, detailed documentation and user support. But in the early nineties, a series of railway signalling products took advantage of the formal *B* method in the design phase. "*The success of B has had a major impact in the sector of railway signalling by influencing the definition of the [EN 50128 2001] guidelines*" [Fantechi 2014b], which is issued by the European Committee for Electrotechnical Standardization (CENELEC).

The standard EN50128 is the guideline for "software for railway control and protection systems", and its latest version is [EN 50128 2011]. In this standard, it does not indicate a specified development method, but classifies a wide range of methods (both traditional technologies and new ones), respecting the safety integrity level, in which, FM is ranked as "Highly Recommended" for *software requirements specification*, *software architecture*, *software design & implementation* and *modelling*. Moreover, the formal proof is highly recommended for *verification & testing* and *data preparation techniques*. However, the guideline gives a list(not exhaustive) of example of FMs, including CSP, CCS, HOL, LOTOS, OBJ, Temporal Logic, VDM, Z Method, *B* method and Model Checking.

Generally, railway signalling system can be classified into two main categories: train control system and railway interlocking system. Train control system contains automatic train protection (ATP), automatic train control (ATC), automatic train operation (ATO) and train management system (TMS), while railway interlocking system establishes safe routes for train movements.

Train control system

With the introduction of ETCS (European Train Control System), formal language studies have been conducted regarding its context. In early research, CPN has been chosen to handle with the modelling of ETCS [Jansen, Meyer Zu Hörste, and Schnieder 1998], a model was given by [Hörste and Schnieder 1999]. Later, to formalise the ETCS natural language and to meet the interest of the railway industry, European Railway Agency (ERA) has launched an "EuRailCheck" project [Cavada et al. 2009], which formalize the system requirements into a subset of UML. One of its outcomes can be found in [Cimatti, Roveri, and Susi 2008], which partly relies on the NuSMV model checker. Some recent advances of this aspect are using SMV [Ghazel 2014], CPN [Lijie et al. 2012] and domain language [Iliasov, Palacin, and Romanovsky 2014]. Moreover, an open source project "OpenETCS" [Feuser and Peleska 2014; Hase 2011] is now working at developing an integrated modelling, development, validation framework of ETCS and lowering the costs.

Moreover, railway companies begin to adopt FM in their development cycle. For example, Ansaldo STS has modelled the Radio Block Center with SDL and Message Sequence Charts [DaSilva, Dehbonei, and Mejia 1992], then it has developed a formal specification of Euroradio protocol [Rosaria et al. 2003] which is provided by UML state Diagrams. In the UK, the Invensys Rail (now part of Siemens' Rail Automation) took part in the "SafeCap" project[Iliasov, Lopatkin, and Romanovsky 2013; Iliasov and Romanovsky 2012], which

has developed a domain specific language (DSL), which is a graphical editor adapted to Event-B analysis. It allows signalling engineers to model railway nodes, verify their safety and improve node capacity. The railway signalling division of General Electric Transportation System (GETS) mentioned their model-based development, which aims to enforce the production safety. The company employed system modelling first for the prototype development [Bacherini et al. 2006], then used those models for requirements formalization and automatic code [Ferrari, Fantechi, et al. 2009].

Railway interlocking system

In the past, safety of relay-based railway interlocking systems was based on the fail-safe concept which relies on the gravity. As the systems are being transited into computer-controlled ones, their safety is more difficult to predict. So railway companies and societies always have a great need to find a suitable and feasible way to maintain RIS safety.

The first approaches of manufacturers could be found in [Bernardeschi et al. 1998; Cimatti, Giunchiglia, et al. 1998; Gnesi et al. 2000; Groote, Vlijmen, and Koorn 1995]. But those approaches are adherent to old relay-based technology. To have an innovative solution, UIC launched a Euro-Interlocking project between 1999 and 2006. This project collaborated with the main European infrastructure companies, with the aim of developing a European level interlocking standard, which could reduce life-cycle costs and optimal compliance with ETCS. One of the outcomes of this project is EIFFRA (Euro-Interlocking formalized functional requirements approach) activity [König and Einer 2003]. Another experience is made by SNCF-RFF, which has modelled the French interlocking logic principles using Statecharts and Statemate [Le Bouar 2003].

Later, UIC launched another project called INESS (INtegrated European Signalling System) from 2009 to 2012. This project aims to develop formal specifications of a future interlocking system, which could also extend and enhance the interface of ERTMS standardisation. Another contribution of this project is to provide business testing tools and methodologies to carry out the testing for signalling applications.

Moreover, there are some DSLs that have been proposed to formalise railway RISs [Haxthausen, Peleska, and Haxthausen 2003; Morley 1994]. The most remarkable one is a geographic approach called EURIS [Berger, Middelraad, and Smith 1992; Dijk et al. 1998] which was developed in the Netherlands and later adopted by Siemens, for their GRACE toolset [Fantechi 2014a; Jung 2000].

When referring to the relevant scientific work and research papers, we found that RIS has been studied by a wide variety of formal methods. [Chen, He, and Huang 2011] develops the RIS logic based on *Statecharts*. [Zafar 2009; Zafar, Khan, and Araki 2012] specifies and validates the RIS using *Z notations*. [Rastocny, Janota, and Zahradnik 2004] specifies the functional safety requirements with *UML*. [Hartonas-Garmhausen et al. 2000] verifies the RIS based on *Symbolic Model Checking*. [Bonacchi, Fantechi, et al. 2014; James, Lawrence, et

al. 2014] introduces different formal verifications via *SAT-based Model Checking*. [Basagiannis, Katsaros, and Pombortsis 2006] carries out analysis and verification with the *SPIN model checker*. [Eriş and Mutlu 2010] using *Petri net* to design a RIS structure. [Grégory, Olaf, et al. 2013] presents a method to assess the dependability performances based on *coloured Petri net*. In [Dincel, Eriş, and Kurtulan 2013], an *Automata*-based RIS is introduced and implemented. But most of the work is not suitable for RIS design in large stations, although some optimization of the specification could help [Winter, Johnston, et al. 2006; Winter and Robinson 2003].

Naturally, during the research, we should always be aware of the trends of this domain, which is usually led by the experts.

Alessandro Fantechi, the professor from the University of Florence, thinks that model checking is of particular interest by many railway signalling industries, because it is the most lightweight method from the process point of view. He did research on exploiting the use of model checking [Fantechi 2012a; Ferrari, Grasso, et al. 2010; Ferrari, Magnani, et al. 2011]. Also in his recent reviews, he has pointed out that "*The 'SAT-based' model checking appears to be more promising in this respect*" [Fantechi 2014a], because in some more recent work [Bonacchi and Fantechi 2014; Bonacchi, Fantechi, et al. 2014; Haxthausen, Peleska, and Pinger 2014; James, Moller, Nguyen, Roggenbach, Schneider, Treharne, et al. 2014; James, Lawrence, et al. 2014], they have a common interest in using "SAT-based" model checking for system verifications.

Anne Elisabeth Haxthausen, the professor from Technical University of Denmark, has been committed to the research of RAISE[14] Specified Language (RSL) in railway applications. The RAISE/RSL [George 1991; The RAISE Language Group 1992, 1995] is a powerful specification and design language, with 20 years' research experience in systematic software development. Her previous works mainly focus on relay-based RISs [Aanæs and Thai 2012; Haxthausen 2011; Haxthausen, Kjær, and Le Bliguet 2011; Haxthausen, Le Bliguet, and Kjær 2010] and interlocking table [Haxthausen 2014; Hede 2012].

Phillip James, a lecturer at Swansea University, is also an advocate of DSL. He has researched DSL at verification using CSP||B [James, Moller, Nguyen, Roggenbach, Schneider, Treharne, et al. 2014; James 2014; James, Moller, Nguyen, Roggenbach, Schneider, and Treharne 2014a,b,c] and SAT-Solving[James, Moller, Nguyen, Roggenbach, Schneider, Treharne, et al. 2014; James, Lawrence, et al. 2014; James and Roggenbach 2011].

Last but not least, we should take a look at the efforts of **Dines Bjørner**, an expert on domain engineering, requirements engineering and formal methods. He believes that the two promising directions of future development for FMs are: verification support for programming Languages and the point of singularity for FM. The point of singularity means that "*specification, programming and verification are performed in an integrated manner, within the same language framework, additionally supported by visualization and meta-programming*"

[14] Rigorous Approach to Industrial Software Engineering

[Bjørner 2012a,b, 2014; Bjørner and Havelund 2014].

1.5.2 The *B* method in railway

The purpose of the *B* method is to formally model a system or software and to prove that all the system or software behaviours respect the properties [Abrial 1996]. Later, there is another wider used version of the *B* language called Event-*B* [Abrial 2010], which aims "*to analyse, study and specify, not only software, but also whole systems*" [Lecomte, Servat, and Pouzancre 2007].

SACEM system

The first industry use of the *B* method took place in SACEM system of RER line A in Paris. In 1988, GEC Alsthom, Matra Transport International (now Siemens Transportation Systems) and CSEE Transport (now part of Ansaldo) started to develop the SACEM system for RATP in cooperation with SNCF [Bowen and Stavridou 1993]. The SACEM is an automatic train protection system, which continuously governs the speed of all trains on the line. It is expected to increase the network traffic by 25%, while maintaining the existing safety level. The software of SACEM was completed with Modula 2 programming language, of which "*63% is regarded as safety-critical and has been subjected to formal specification and verification*" [Guiho and Hennebert 1990; Hennebert and Guiho 1993]. A formal specification of the functional requirement was made in the *B* language and "*the proofs were done interactively using automatically generated verification conditions for the code*" [Woodcock, Larsen, et al. 2009].

The FM in this project helped to find 12 differences between the implementation and the natural language specification [Guiho and Hennebert 1990]. The successful application of FM in the SACEM system convinces RATP to continue the use of FM in their following projects.

MÉTÉOR

Météor [Behm, Benoit, et al. 1999; Behm, Desforges, and Meynadier 1998], the new metro line number 14, is the first driverless metro line in Paris, and its ATO system is also the first real success of industrial use of formal methods, which has been in full operation since 1998. This project is supported by RATP, and promoted by the *B* method and Atelier B, which is the implementation tool. Matra Transport International developed the system using *B* formal method to develop and validate the self-critical parts of the ATO system. This automatic guide-way transportation system allows managing automatic driverless trains together with non-equipped manually driven trains, in which, "*the safety-critical software controls the running and stopping of every train, and controls opening and closing of doors located in trains and platforms*" [Behm, Benoit, et al. 1999].

One of the conditions for success should be thanks to the industrialisation of FM. The *B* method and its associated tool kit Atelier B were industrialized by RATP, Matra Transport International and Stéria Méditérranée. The formal development started by specifying the nature requirement in the abstract *B* machine. Then, these models/machines are refined step-by-step into more concrete model. Finally, they are transformed into the programming language Ada. Errors were found during the proofs. As a result, no bugs are detected during the system testing, either at function validation on host computer, integration validation on target computer or on-site tests. Furthermore, there is no bugs have been found since the system was put into operation [Haxthausen 2010; Lecomte, Servat, and Pouzancre 2007]. The formal development was cost-effective. In particular, initial budgets could be kept to.

Other projects

The VAL system [Ferbeck 1981] is the technical predecessor of MÉTÉOR. It is the first fully automatic driverless metro in the world, and is controlled by digital monolithic integrated circuits and was applied in Lille in 1983. The "VAL de Roissy" project re-developed the VAL system by *B* method. This project is a driveless light train system for the Roissy Charles de Gaulle airport shuttle [Badeau and Amelot 2005]. The safety software for the automatisms was developed by *B* method and has been in full operation since 2007. Siemens Transportation Systems (formerly Matra Transport) took the responsibility of developing this shuttle system and subcontracted the safety-critical parts to ClearSy Company using *B* language. After this project, new VALs are now operating in Taipei, Toulouse, Rennes and Turin [Boulanger 2014].

Another recent *B* method engineering application is the COPPILOT system [Lecomte 2008; Patin, Pouzancre, and Servat 2006], It is a platform screen door controller, which controls the opening and closing of the doors at a metro station platform. These doors are installed to keep traffic flowing safely. This system was called upon by RATP and developed by ClearSy company via formal method. Its development complies with SIL 3 safety software. Now, it has been installed in the Paris Metro and the São Paulo Metro.

Related works

In railway signalling related studies, the prominent ones are [Leuschel et al. 2011; Sabatier et al. 2012], which are experience of large scale work. [Cimatti, Corvino, et al. 2012; Ferrari, Magnani, et al. 2011; James and Roggenbach 2011; Kanso, Moller, and Setzer 2009] verify the safety of RIS with logical approaches. Besides, some researchers use CSP||B to address RIS verification, such as [Moller et al. 2013] and Phillip James, whom we have already mentioned.

1.5.3 Petri net in railway

The concept of Petri net (PN) was first introduced by C.A. Petri in his Ph.D. thesis [Petri 1962]. Having the advantages of expressing discrete event control systems [Holloway, Krogh, and Giua 1997a; Murata 1989], PN has been extensively researched for specification, design, verification, performance evaluation, and simulation.

Studies of PN in railway can be traced back almost 20 years ago [Decknatel and Schnieder 1998]. Afterwards, many researchers and teams devoted their work to contributing to modelling and analysing railway system using Petri Nets, such as [Decknatel 1999; Hielscher et al. 1998; Hörste and Schnieder 1999; Lei, Tanglong, and Jian 2000]. But these early approaches are mainly concerned with isolated phenomena. There are some prominent papers that study the scheduling of railway stations [Aalst and Odijk 1995], and optimal behaviour on a point [Ren and Zhou 1995]. However, the literature of railway modelling and analysis using classical Petri net is not extensive in this paper. Such a survey could be found inside a Ph.D. thesis [BAppSc 1998].

Because 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 coloured Petri net (CPN) [Jensen 1981, 1987] and Predicate/Transition Nets [Genrich 1987], later be revisited in [Billington 1991].

With the help of such high-level PNs, there comes a large scale application—Oslo Subway [Bjørk 2006; Hagalisletto et al. 2007; Moen and Yu 2004; Yu 2004], which integrates Petri net into the system development to simulate the Oslo subway and analyse schedules of trains. This project developed a specification tool for specifying and automatically constructing large CPN models of railroads. But due to the immature nature of CPN support tools, "*neither Design/CPN nor CPN Tools are able to represent or execute the models, although standard CPN formats were used in the implementation*" [Hagalisletto et al. 2007]. However, another important experience of this project shows that PN 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 PNs nor FMs, they understood the models, and gave improving suggestions.

The specification, analysis and implementation of railway control logic are always a hot research topic. In work [Fanti, Giua, and Seatzu 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] uses a fuzzy PN for railway traffic control. A similar solution can be found in [Kaakai, Hayat, and El Moudni 2007] using a hybrid PN.

The level-crossing (LC) is also a critical crux in both road and rail infrastructures. Stochastic PNs are applied in [Ghazel 2009; Huang, Weng, and Zhou 2010] in order to precisely reflect the system's dynamics. Furthermore, stochastic PNs could be used to evaluate the

real time system in railways, such as data processing [Zimmermann and Hommel 2003], Device-to-Device communication [Lei, Zhang, 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, Ning, and Xu 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 Pole technilogique 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 PNs, 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 signalling rules and European signalling standards, by specification and analysis of CPNs [Buchheit et al. 2011; Grégory, Nicolae, et al. 2010; Lalouette et al. 2010].

1.5.4 Model transformation of model-driven engineering

In MDE, everything is a model. Thus, the model transformations are considered as the "heart and soul"[Lúcio et al. 2014] of model driven engineering, and can be used for a variety of purposes, such as the generation of models on different levels of abstraction, the creation of different views on a system and automation of model evolution [Czarnecki and Helsen 2006].

They are already some good literature reviews [Amrani, Combemale, et al. 2014; Amrani, Lúcio, et al. 2012; Calegari and Szasz 2013; Lúcio et al. 2014] on this domain. In the literature, several issues around MDE have been studied and researched, for example, transformation languages (TLs) (Graph-based TL [Jurack and Taentzer 2010], meta-programming language [Combemale et al. 2009]), mapping techniques [Hammoudi, Vale, and Lopes 2008;

Lopes et al. 2006], model transformation classification [Czarnecki and Helsen 2006; Mens and Van Gorp 2006].

According to their reviews, a transformation has two natures [Bézivin et al. 2006]: transformation model and model transformation. The first propriety emphasizes on the model of computation that is embedded in a transformation, while the second one focuses on particular artefacts manipulated by a transformation. During all the research, meta-models are often defined using UML Class Diagram. However, there are some other specific languages, such as MOF (Meta Objec Facility) [Object Management Group 2003], EMF (Eclipse Modelling Framework) [Steinberg, Budinsky, and Paternostro 2009] and KM3 (Kernel MetaMeta-Model) [Atlas 2005].

In our research, we are more interested in the PN-related applications in MDE. So far, the PNs have been mainly used for formal verification, either verifying the original model or the transformation process. A general introduction can be found in Work [Maximova, Ehrig, and Ermel 2010], which shows the Graph-based TL share a common formal background with Petri net. M. Wimmer developed a TROPIC (TRansformations On Petri Nets In Color) framework[Wimmer, Kappel, and Kusel 2009; Wimmer, Kusel, Reiter, et al. 2009; Wimmer, Kusel, Schoenboeck, Kappel, et al. 2009; Wimmer, Kusel, Schoenboeck, Reiter, et al. 2009] that provides mapping based on a domain specific language (DSL) and expressed as CPNs. Varró introduces a transformation from UML statecharts to Petri nets in [Varró and Pataricza 2003] and later analyses graph transformation systems via PNs [Varró, Gyapay, et al. 2006]. [Combemale et al. 2009] mapping a DSL, xSPEM, into prioritized time Petri net, and formally prove the transformation. [Lara and Vangheluwe 2009] introduces a transformation from Domain-Specific Visual Languages (DSVLs) into Petri net, in order to benefit from its formal verification abilities. [Syriani and Ergin 2012] transformed a UML activity diagram into a Petri net model to detect deadlocks. [Kühne et al. 2009] defines a translation from State automata to Petri net. The translated model is used to check the given propriety of an input sequence.

Anyway there are few researchers working on transforming PNs into other languages. Korečko, in [Korečko, HudáK, and ŠIMOŇÁK 2007; Korečko and Sobota 2014], introduces a mapping method to translate classical Petri nets into classic *B* machines and Event-*B* machine. Attiogbe [Attiogbe 2009] introduces an encoding method from classical PNs to *B* language. The works of Bon [Bon 2000; Bon and Dutilleul 2013] introduce a solution to transform a CPN model into a *B* machine respecting Atelier B syntax.

1.6 Conclusion

In this chapter, we first introduce the scientific background of my research, including system engineering, safety-critical system and formal method. Then, we present an extensive literature review on the necessary knowledge based on the survey of the art of the various

tools or techniques. This thesis is dedicated to the modelling of complex systems considering safety-critical scope via formal languages. So, the literature reviews have to be an interdisciplinary document survey. After a general review concerning formal methods for safety-critical system in the rail domain, the *B* method and the Petri nets techniques and their applications in railway are studied. Furthermore, the model transformation of model-driven engineering is discussed.

To pursue the highest possible guarantees is always a key aspect in the adoption of computer-controlled railway applications. From the survey on the state of the art of formal methods for safety-critical systems, we have enumerated numerous references and some projects. All this research make railway signalling a fruitful area of formal method theories and applications.

Considering the French railway context and the industrial robustness, we choose the Petri nets and the *B* method for system modelling. The Petri nets have many great features, such as graphic notations, precise mathematical semantics and hierarchy, which mean these languages are user friendly, formal and can be applied for large complex systems. Most important is that Petri nets can be understood by engineering experts, which means we can always get assistance and suggestions from them. The *B* method aims to analyse, study and specify the software or the whole system. We already have many successful applications of the *B* method, such as SACEM system, MÉTÉOR and the new VAL system.

From the engineering point of view, these two methods have some advantages and disadvantages, and we consider them as complementary in the French railway context. The proposition leads us to take advantage of both methods using the techniques of model driven engineering. So, an interesting solution should be a model transformation: from the PNs to the abstract *B* machines. Such a transformation can build a bridge between critical tasks, from a strong requirement analysis towards a valid implementation on a real system. Last but not least, if we want to validate safety at a system level, this transformation is a strong contribution in order to integrate information from different parts.

Chapter 2

Some principles of railway safety

Any applications of signal centre and signalling related procedures require a full explanation of safety properties. This chapter aims to describe and to formalise some major safety properties and the control logic of critical systems in railway signalling.

Around the safety nature of the signalling based railway system, a step by step analysis has been performed: First, the particular railway safety in France is introduced. Then the composition structures and major components of railway system are stated. Finally, the French railway interlocking system is introduced, including the general architecture and different operating phases of a signal centre.

2.1 Railway safety

When analysing the safety of a particular railway signalling system or device, we have to consider its integration into its outside environment, in which it participates as a component of the whole system.

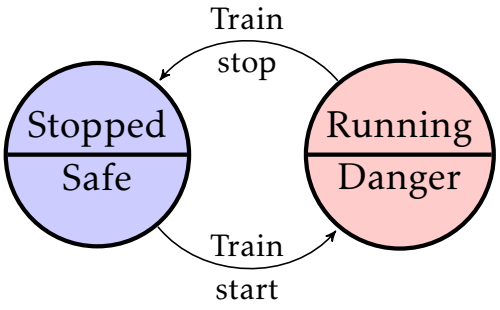


Figure 2.1 – Railway system state

2.1.1 Concept of safety

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 signalling 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. 2.1.

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 collision.

In some engineering standards, they may refer to another similar concept — reliability. There are numerous general definitions of it, such as IEEE, which defines reliability as "*the ability of a system or component to perform its required functions under stated conditions for a specified period of time*" [IEEE 1990]. Still, it is hard to give an unambiguous and quantified definition for railway. However, if we look at their respective consequences, it will be clearer. Reliability means that the system maintains safety, while the dangers may become system failures [Antoni 2009c].

How to evaluate the reliability of a system, and do all the undesirable states have equal importance? This is in the domain of safety requirement engineering, which aims to provide a safer system under certain safety levels. However, Absolute safety is a beautiful dream that can never be achieved, so engineers always have to face the unreliability of the machines. The famous explanation of this fact is Murphy's law: "Anything that can go wrong, will go wrong". This law transforms a probability into a certainty. For a given unsafe event with determinate probability of occurrence, if given enough time the probability of this unsafe event within a limited time is actually very high.

How to ensure safety when the systems suffer from the lack of reliability? Let us review the existing applications and the experience in France. In electromechanical devices, used in most railway signalling system before the introduction of the computer, gravity was used to bring the system to the fail-safe state in any critical event. For example, a switch itself could fail to block a section in case of critical events. Another important point is that the safety of such a system does not primarily depend on the reliability of the devices, but on the reliability of some physical laws and the "oriented" design concepts. As in electromechanical devices, the system states are not considered as equally stable. Therefore, the system is "oriented" designed according to the following principles:

- Some states are more stable than others
- "Spontaneous" subsystem always evolve towards the most stable state changes
- It is possible to match the most stable device state with the safest system state

In this way, railway systems are designed with the techniques above, so that their operating results will always evolve towards stable states which are intended to be safe. However, enhanced system reliability is not enough for the "absolute safety". In the view of safety, besides reliability, there is a need for a rigorous preventive signalling rule.

Signalling rules in the French railways are historically based on *determinism*. Every system state has one or more cause. 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. Consequently, a system is designed in accordance with the following design rules [Antoni 2009c]:

- A single cause can lead to an accident
- potential failures can be recovered by fail-safe technique. detectable failures always go in the direction of safe
- Limit the shares of men and procedures
- A precise definition of the role for each actor
- Independent recovery loops as much as possible between different actors
- Supervision (permanent, human, procedures or tools)
- Continuous improvement based on feedback

Moreover, the reliability of the signalling rule has a direct influence on system safety. The fewer time delays and incidents, the fewer dispatchers and procedures will be needed and the risk will be lower as well. In addition, the fixed installations must also be fast repairable or self-recovery in order to guarantee the required safety level. In Fig. 2.2, there are two paths to the "unsafe" state. So, if a system can quickly recover from the "Failure Detected" state, it will obviously reduce the number of the accidents.

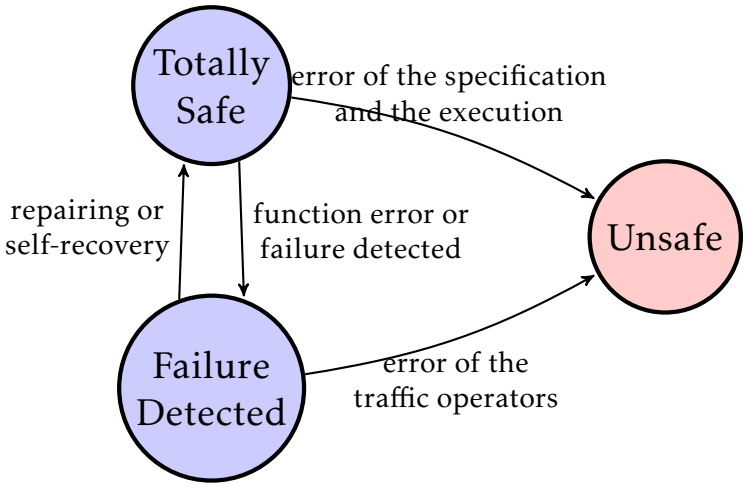


Figure 2.2 – Impact of reliability of the overall safety of a signalling system

Nevertheless, from a technique point of view, humans, who cannot perfectly integrate into the automated systems, are still a weak link in the signalling control loop. However, humans play a main role in the railway system. Human dispatchers and operators are usually at the top level of the whole system, but their reliability does not depend on technique. Generally, the increasing safety level guarantees the system performances. The failure rate of a system fails from *totally safe* state and *failure detected* state is quite low, so that it can "compensate" human operations in the control procedures. But we should always be aware that, when the traffic workload is increased, the error rate of an operator may become unacceptable.

2.1.2 Safety management of French railway system

The deterministic *reasoning* can only be applied to a closed system, because an open system has unexpected interactions with the outside world, and there will be a risk of unforeseen system state. This prerequisite also applies in other industrial safety domains.

Thus, the principle of the organization of the external environment is to limit the number of the 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. Moreover, for any system it must have at least one safe state. It can be found in the system states exploitation, and it should be accessible.

As a matter of fact, national railway systems are managed by transportation service providers who do not develop every system. Rather, they act as system integrators of systems purchased from external suppliers. Therefore, the technical characteristics of the railway require identifying and anticipating potential conflicts, in order to enhance the integration safety level. Overall, the integration of different systems helps to achieve a safe enough system. According to their properties, safety management can be roughly divided into two categories:

- *Operating safety* is the interactions between integrated systems. Any expected change among the interactions could interfere with the traffic. It depends on the operators following the signalling rules and procedures.
- *Technical safety* aims to preserve the functional characteristics of railway traffic, which allow trains to run on an infrastructure. It depends on the device development and maintenance with respect to the operation procedures of operators and drivers.

In such a railway system, any modification of a system component may affect other components, and threaten the existing system safety. It is necessary to have rigorous integrating procedures, which are based on verification and validation, functional and safety assessment, and safety approval.

The French railway practice is originated in a deterministic approach, called the intrinsic safety [Antoni 2009c], which implies an exhaustive knowledge of the component failure models:

- Failure modes are determined by the physical laws (gravity fail-safe components)
- The combination of failures are accessible for maintenance staff
- The (first) failure could be detected by the default system state

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

Another important fact is that computers cannot rely on the gravity. That means it is impossible for these systems to automatically lead to the default state, and it is also impossible to identify and address all the possible causes of the unsafe failures. This is also the cause that has long delayed the acceptance of computer-controlled signalling equipment.

It has been a key aspect for the infrastructure management to find a very stable methodology to ensure the correction of the exhaustive functional applications for the adoption of computer-controlled equipment in railway applications.

2.1.3 Problematic of computer-controlled system

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 behaviours of such automated systems must be deterministic
- Some properties should be specified rigorously
  - safety predicates
  - functional predicates
  - assumptions of interactions with external environment
- For model checking based formal proof, it is only possible when the reachable system states is finite

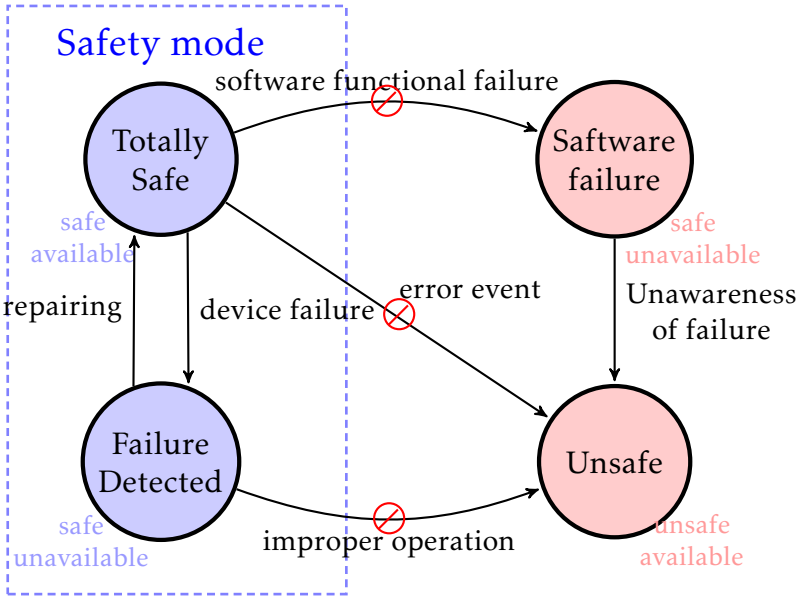


Figure 2.3 – The overall safety of a computer-controlled signalling system

The safety state of a computer-controlled signalling system is shown in Fig. 2.3. Transitions with red forbidden sign are the undesired system changes, which should be identified and reduced through formal specification & verification. In this way, the final system could operate as a "fail safe" system.

2.2 Railway context

Compared to other transport modes, the greatest feature of the railway is that all the vehicles run on the rail. Normally, either the locomotive or the rolling stock weigh tons and

are hundreds of meters long. Unlike other vehicle drivers, a train driver is free from the running direction of the train. Besides the long length and the high speed of the train make it impossible for the driver to observe and judge the situation ahead by his own eyes. The fixed tracks make the whole railway system in an one-dimensional world, which has the following characteristics:

- The railway tracks are made of steel, which can withstand very heavy transits, but the wheel-rail adhesion coefficient is lower than other road transportations.
- The direction of the train can only be changed at particular locations, such as a station or a point.

The above inherent characteristics adds to the difficulty of the train operation. First the braking distance is greatly exceed the visionary scope of a human driver [1]. Second, the rail prevents the driver from changing the route in order to avoid collisions.

Because of these facts, the train driver, in normal operations, cannon drive "on sight"[2], and he have no right to control the direction of the trains. So there needs to be some other staff to take responsibility for the organization of the traffic and the operation of the fixed installations.

The duty of the dispatchers are:

- Organize a well-thought schedule of traffic management, which will reasonably space apart the trains on the same route.
- Distribute movement authority. With the aide of train protection technique, It could inform the train drivers early enough to fully stop a train and maintain a sufficient safe length to ensure a safe transit.

So the mode of railway transportation is in a distributed environment, under multiple control and guides. The "operation of the train" is not by its driver, but a result of multi-party cooperation. In practice the main components of the railway system that contribute to the operation are:

- Fixed infrastructure:
  - trackwork: junction, point, level-crossing, $\cdots$
  - track structure: curve, ramp, tunnel, bridge, $\cdots$
  - track characteristics: degree of curvature, gradient, tunnel diameter, $\cdots$

[1] Compared with road vehicles, the braking distances of train are not optimistic. For example, the full stop distance of a freight trains running at $100km/h$ needs about $1000m$, while for a passenger train at $300km/h$, the distance will be about $3500m$.

[2] on sight driving mode is only accepted as a degrade mode

- Locomotive & rolling stock:
  - functional properties: mass, moment of inertia, track transition curve, top speed, braking ability, ⋯
  - usability: door control system, ⋯
- Signalling
  - It is a series of signalling rules and the control procedures
  - It organizes the "safe" route for trains, and transmit commands to drivers
  - It obviates the risks from the train movements and their intersection with the fixed infrastructures

Since the railway system has such complex components, and the signalling system acts as a link between trains and fixed infrastructure, it is not allowed to have any functional defects or technical gaps in the railway signalling.

2.3 The railway system

Similar to any other industrial operation process, the entire railway system can also be categorised into three parts: human actors, signalling procedures and facilities.

Before clearly specifying the safety requirements, it is necessary to define the framework and major components of the railway system. The human actors include train drivers on board and the dispatchers on the ground. They operate the locomotives and the fixed infrastructure respectively, in accordance with a series of global signalling procedures. The system diagram is presented in Fig. 2.4.

The rail system is therefore composed of four types of actors. Fig. 2.4 illustrates the normal operation mode of the railway system. It is obvious that all the facilities require human actors to operate and to supervise. The train movement represents the interactions between the rolling stocks and the fixed infrastructures. All the components are comply with the unified signalling procedures, which ensure consistency and correctness.

2.3.1 Human actors

Human actors are an indispensable link of the system's safety and reliability. This is based more on human reliability than the technical reliability. In order to enhance the system's safety level, to reduce the workload of staffs and to increase the system's efficiency, automatic and semi-automatic facilities are introduced into the railway system. These devices can effectively reduce the probability of system error. A schematic diagram of such an effect of the automatic devices is illustrated in Fig. 2.5.

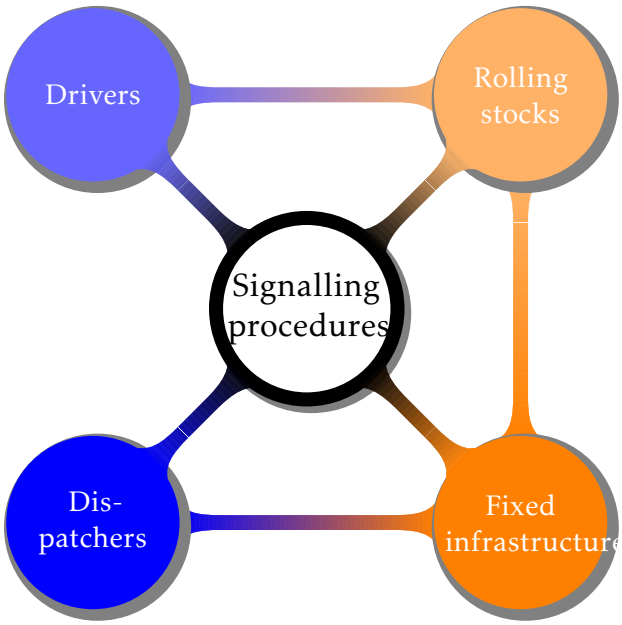


Figure 2.4 – The global vision of railway system

In Fig. 2.5, we assume that every actor and device has an intrinsic error rate (P_1, P_1', P_2 ,P_2'). Generally, it is believed that the error rate of the automatic devices is much lower than human, which means $P_1 >> P_1'$ and $P_2 >> P_2'$. The final traffic safety is a combined result of

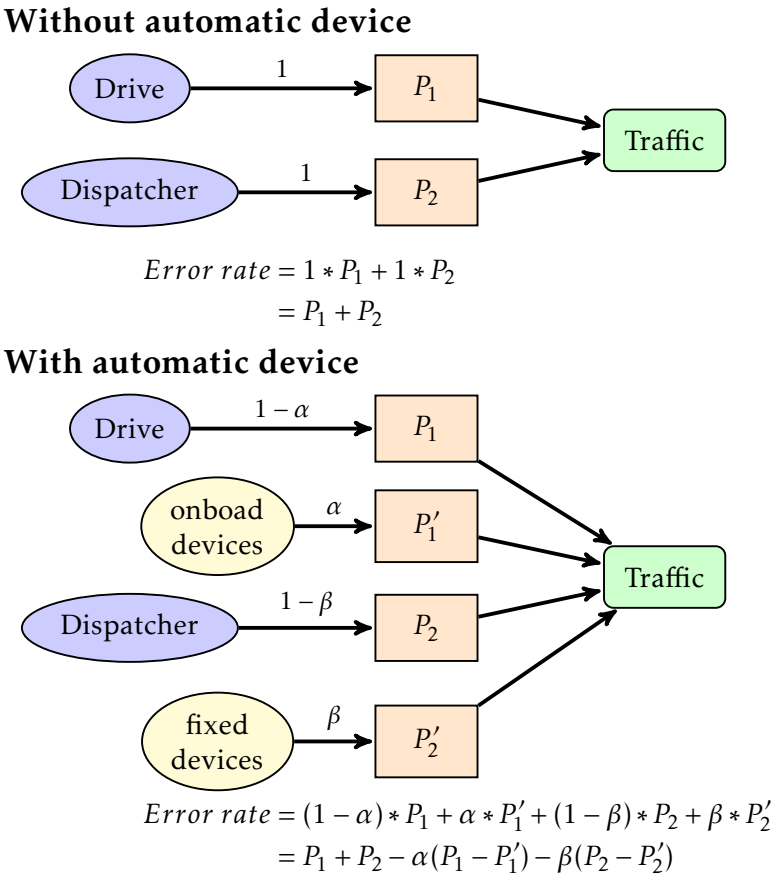


Figure 2.5 – The effect of the automatic devices in system safety

each player. α and β are the percentage of task distributions, where $\alpha, \beta \in [0,1]$. From the figure, it is clear that the system will be safer with automatic devices.

But once the system fails and evidence points to human error, there will always be a controversial paradox: why does an automatic system allow people to take actions that may lead to unexpected situations? Should people or the automatic system take the responsibility for the final security guarantee? From the accident reports, the cause of an accident is always an integration of multiple factors, even some tiny mistakes or coincidence events.

The reliable nature of such automatic systems naturally induces the human actors to have a strong tendency to trust them. Actually, the highly automated system has sophisticated and complex characteristics. Most human actors treat them as black boxes in their daily operations. Because of a lack of knowledge of the internal mechanisms, in some failure situations, it is sometimes difficult to make a quick right response. On the other hand, the failure probability of dangerous dysfunctions is so low that most actors will not encounter this in their lives, not to mention their experience in dealing with it. However, treatment of the failure situations and the dysfunctions still needs human actors as a part of signalling procedures.

Human actors are the only ones who are capable of dealing with all situations, especially in the case of failures and dysfunctions. Human-in-the-loop, plays a major role in the railway system. It is necessary to take human actions into account for any validation of system safety [Antoni 2009c].

2.3.2 Signalling procedures

The signalling procedures are designed to ensure the consistency of all the activities between different actors. It also monitors and executes the operation of all the equipment. The procedures are based on a unified language, the signalling rules, whose terms and vocabularies have been clearly defined and which is known to all the actors.

Different actors have different operation targets such as various types of rolling stock and all possible infrastructures. They need to follow different operation procedures, but all these procedures comply with some basic rules, such as the national signalling standard. In computer-controlled system, the procedures are greatly reduced on account of the highly automated facilities, which could implement some former low-level procedures.

There are some examples of the signalling procedures:

The train driver can only passively obey the external signals, whether through the radio or the cab screen. According to the termination of a signal, the driver must immediately take action to adapt the train to the current signal.

The dispatchers could monitor the occupation status of all the tracks. They can interlock a series of tracks to provide a safe route and open the signal light to inform the train driver. Although the dispatchers are given the opportunity to operate and arrange the sig-

nalling facilities at any time, they have to use their rights only when necessary and under the instruction of superior operating plans.

The various applications of the signalling procedure should be always validated for safety reasons.

2.3.3 Installations

The installation could be divided into two categories: on-board devices and trackside fixed installations. They are composed of highly safe and reliable components, in order to secure human actions and improve system performance. The development trend is to gradually replace manual processes with automated devices, in order to decrease operational errors.

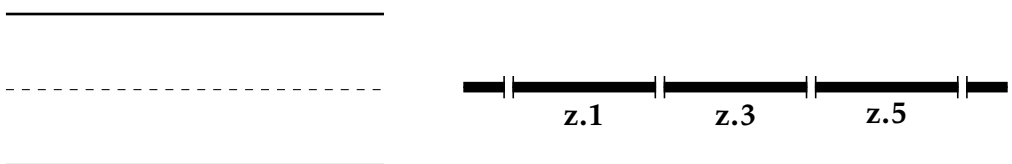
The on-board devices are mainly equipped inside locomotives, including power, signalling and mechanic devices. The safety related devices are: brake system, speed control system, train protection system and cab signalling.

The safety related devices of fix installations are: points, interlocking system, level-crossings and trackside signals

In order to specify the safety problems of the railway system, especially the interlocking system, in the rest of this section, we will describe the component of the critical network components.

Track Segment

The track segment is the main component of a railway network. It is a single piece of railway track with two or three ends. For safety reasons, track segments are always equipped with signalling for train movements. The one with two ends is a simple straight track segment, which can be driven in both directions, as shown in Fig. 2.6(a). It consists of two parallel rails without branches. In the French system schematic diagram, the straight track is drawn as a "ZONE" with insulated ends in Fig. 2.6(b). The track segment is called a "Turnout", if it is connected with three other track segments.



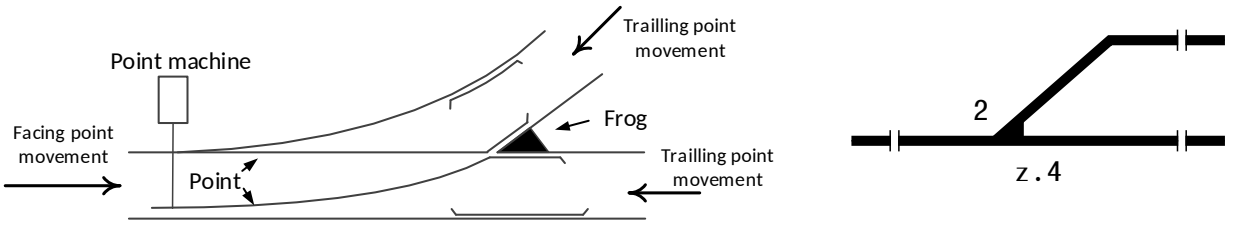
(a) straight track segment (b) schematic diagram

Figure 2.6 – Components of track segments

Turnout/Point

A turnout is an assembly of track segment, movable points, and a frog, which affects the tangential branching of tracks and allows trains to run over one track or another (Fig. 2.7(a)).

In the French system schematic diagram, the turnout is typically represented as a "ZONE" with a point. For example in Fig. 2.7(b), "z.4" is the ID of the current track segment and the number 2 above is the ID of the point.



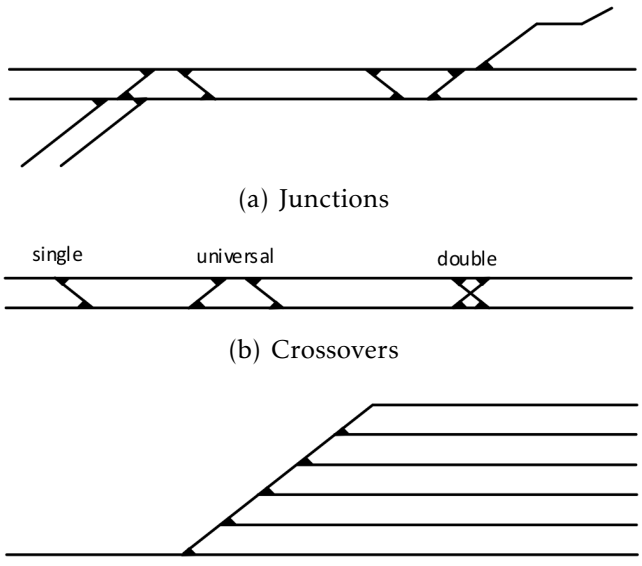
(a) Turnout and its compositions (b) schematic diagram

Figure 2.7 – Components of turnout

In fact, the term turnout[3] is mostly used in civil engineering, while in the railway signalling domain it is usually referred to as "point", although this term in its original meaning only applies to the part of a turnout where the points are located. In our following research, we use "point" to refer to "turnout".

The points may be operated manually or by a point machine. The part where two rails cross is referred to as a frog, which is also sometimes operated by a point machine. In each position change, the point will first disconnect with the original track branch, move to the new branch and build the new connection. The train movement in this area approaching the point is called "facing point movement", whereas the movement approaching the frog is called "trailing point movement".

For a single point, the safety requirement and its control logic are very simple and clear.



(a) Junctions

(b) Crossovers

(c) Ladder

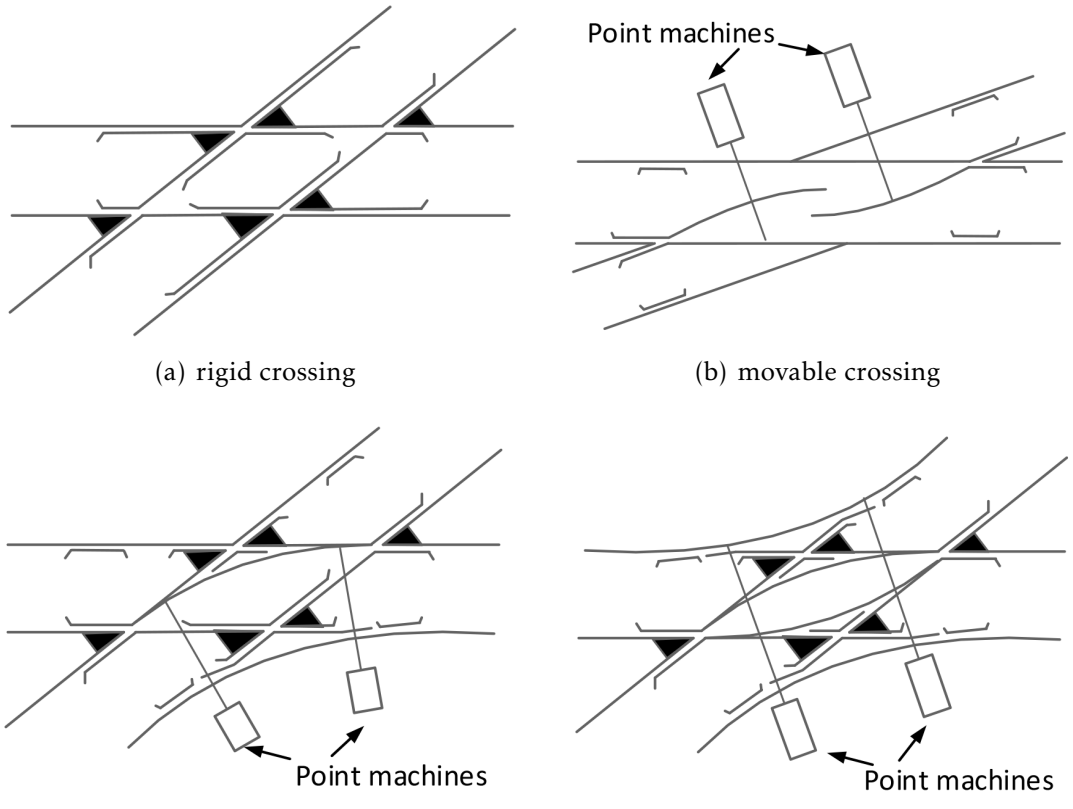
Figure 2.8 – Basic point arrangements

[3]In the American system, a turnout is called a switch, and the point machine is called a switch machine

But in real cases, the railway network is always a complicated combination of elements, which need special interlocking rules to ensure traffic safety. Some common combination, such as junction, crossover and ladder, are shown in Fig. 2.8. A junction is an arrangement where a line is joined by another one. A crossover is a connection between two parallel tracks which enables trains to move from one line to another. A ladder is a series of point and parallel tracks that allow trains to access any parallel tracks. The ladders are normally for yard and terminal layouts.

Crossing

The railway crossing is a critical component of the railway network. It is the intersection of two rail routes, as shown in Fig. 2.9. There are several types of crossing. Some crossings that are equipped with points allow trains to move from one rail route to another. For safety reasons, if a train is moving along on a crossing, then for both rail routes, this crossing needs to be locked to avoid collision.



(a) rigid crossing (b) movable crossing

(c) single slip (d) double slip

Figure 2.9 – Components of crossings

2.4 The risk from signalling system

From the analysis above, the characteristics of the railway system that may be potential safety restrictions are shown in Table 2.1

Table 2.1 – Safety restrictions of railway system

Item	Characteristics	Safety restrictions
①	One-dimensional	unable to overtake and avoid
②	Low adhesion	long braking distance
③	Points & Crossings	traffic convergences
④	Third parties	uncertain factors

From these potential restrictions, there are 3 categories of rail accidents, which contain 10 kinds of dangerous events. In Table 2.2, each event should be monitored and prevented by one or more systems, including rolling stock, fixed installations and interlocking systems.

It should be noted that in the PERFECT project frame, we study the implementation of ERTMS in high speed lines. However, there will be no level-crossings along the high speed lines, so the level-crossing is not one of our research topics.

2.5 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 signalling 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.

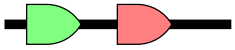
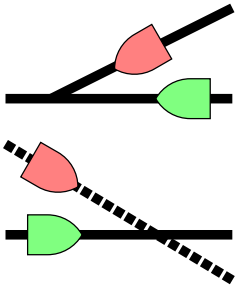
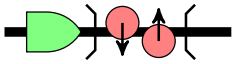
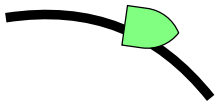
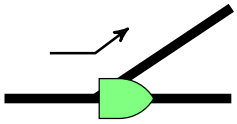
2.5.1 Overview

There is a simple example of an interlocking system, as illustrated in Fig. 2.10. Track segments are represented in a topology structure, and all of them have track circuits, which 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.

In Fig. 2.10, 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;

Table 2.2 – Dangerous events of railway system

	Dangerous events		Causes
	Name	Explanation	
Collisions	Head-on collision		(1) (2)
	Rear collision		(1) (2)
	flank collision		(2) (3)
	Level-crossing Collision		(1) (2) (4)
	Collision with Obstructions		(1) (2) (4)
Derailments	Plain track		(2)
	Curves		(2)
	Junctions		(2) (3)
Other	Fire & explosions		(4)
	Fails from trains, individual on tracks		(4)

- 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 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 authorise the movements of the train. They are handled by the interlocking system, and can be considered as a 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 signalling, the term “interlocking” has two meanings [Pachl 2002]. First, “an interlocking” is an arrangement of signal appliances that prevent conflicting movements through an ar-

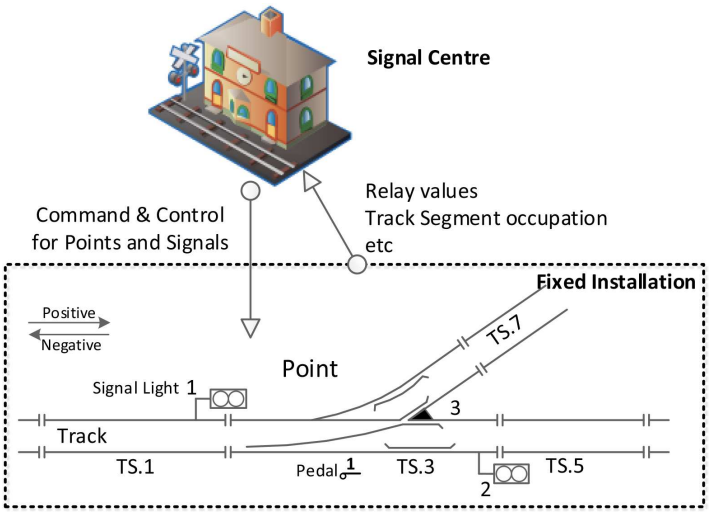


Figure 2.10 – An example of railway interlocking system

rangement 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 signalling operations and the fixed installations.

Signalling 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, ⋯, while human control deals with decision-making, such as route selection, route mode selection, route manual destruction, ⋯, and some non-regular operations, such as shunting operations.

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

2.5.2 The general architecture of the French signal centre

The general architecture of a signal centre is the result of historical evolution. It is broken down into the different levels of Fig. 2.11, which provide different security levels of function.

In Fig. 2.11, Level 0 is human machine interface. Its corresponding functions are command interlocking routes and certain positions of the components. Level 0 is achieved with a dependability of SIL2 [Antoni 2009c].

Level 1 is interlocking system. Its aim is to ensure that the various commands are operated in compliance with all the safety rules. This level is the core of the system safety. It must be realized with the highest level of security, the SIL4.

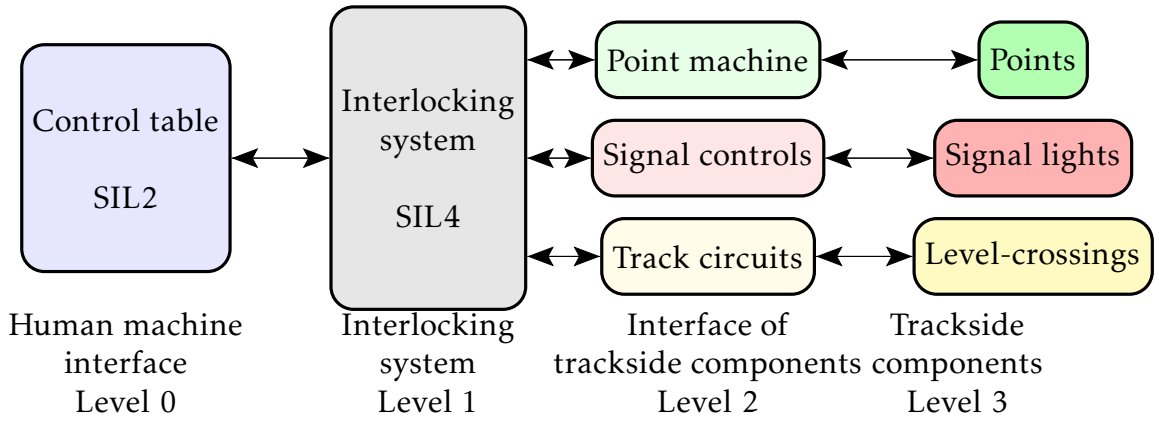


Figure 2.11 – The general architecture of a signal centre

Level 2 is the control interface of the trackside components, such as the point machines, signal controls and track circuits. This level should be achieved with the fail-safe concept, and it must also be realized with SIL4 security level.

Level 3 is the trackside components, such as points, signal lights and level-crossings.

2.5.3 The operating phases of a signal centre

Since the advent of the PRS[12] type signalling control system, the framework of operating phases[Rétiveau 1987] is basically set down, and still in use today. Although with the advances in technology, many new types of signalling control systems (PRG[13], PRCI[14]) have been developed, they all comply with the following basic principles:

- There are a route control interface (buttons or a computer screen) and an optimal interlocking table.
- It is possible to save the following functions:
  - Saving a route before its formation conditions are met
  - Forming a permanent route (for several successive trains)

[4]CIt (route control relay, french: le relais de **C**ommande de l'**It**inéraire)
[5]CAg (point control relay, french: le relais de **C**ommande des **Aig**uilles)
[6]Tr.I (odd transit relay, french: le relais de **Tr**ansit **I**mpair)
[7]Tr.P (even transit relay, french: le relais de **Tr**ansit **P**air)
[8]RRV (track circuit relay, french: **R**épétiteur du **R**elais de **V**oie)
[9]EIt (entrance signal relay of interlocked route, french: le relais d'**E**nclenchement d'**It**inéraire du signal d'entrée)
[10]RIt (route repeater relay, french: le relais **R**épétiteur d'**It**inéraire)
[11]KAg (point checking relay, french: le **Contrôle** impératif des **Aig**uilles)
[12]PRS (relay-based flexible signalling control, french: **P**oste d'aiguillage tout **R**elais à transit **S**ouple)
[13]PRG (relay-based geographic cable signalling control, french: **P**oste d'aiguillage tout **R**elais à câblage **G**éographique)
[14]PRCI (relay-based computer-controlled control, french: **P**oste d'aiguillage à **R**elais à **C**ommande **I**nformatique)

  - The operation is "readable" and fully deterministic

- All the operations are clearly and rigorously specified in a regulation

Generally, the operations are carried out in successive phases that take account of successive behaviours of points and track segments (Table 2.3 and Fig. 2.12).

The safety principle of the interlocking is achieved by the function of transit relays. Normally, each track segment has at least one transit relay, *Tr.I* or/and *Tr.P*(in Fig. 2.12). Each transit relay is responsible for one direction of traffic. Any of the two relays, which are inter-

Table 2.3 – Function phases of establishing a safe route (based on PRS type)

Phases	**Actions**	**Translations in PRS**	**Results**
Control & register (*commande et enregistrement*)	Press down the button	Activate the route control relay (CIt^4), the button start flashing	Interlock the point control relay (CAg^5)
Preparation (*préparation*)	Positioning the point for the route on the condition that this point is not interlocked or occupied by other routes	Only when the related related transits ($Tr.I^6$ and $Tr.P^7$) are activated and the track circuit relay (RRV^8) are deactivated, then the point control relay (*CAg*) can topple over to the right position.	After all the points of this route are placed in the right position, the related relay of the entrance signal of route interlocking (EIt^9) is allowed to be switched to "open" position
Formation interlocking and verification (*Enclenchement et contrôle de formation*)	Interlock this route and prevent the formation of other conflicting routes	Deactivate the transit relay (*Tr.I* or *Tr.P*), which ensures the interlocking of the transit; Active the route repeater relay (RIt^{10}) which ensures continuous control of the position of *CAg* and *EIt*.	The button stop flashing and switch on steady
Verification (*Contrôle*)	verify the establishment of this route and open the entrance signal light	The effective position of the point are represented by the status of relay KAg^{11}; The activation of *KAg* allows us to control the entrance signal light	Switch on the entrance signal light

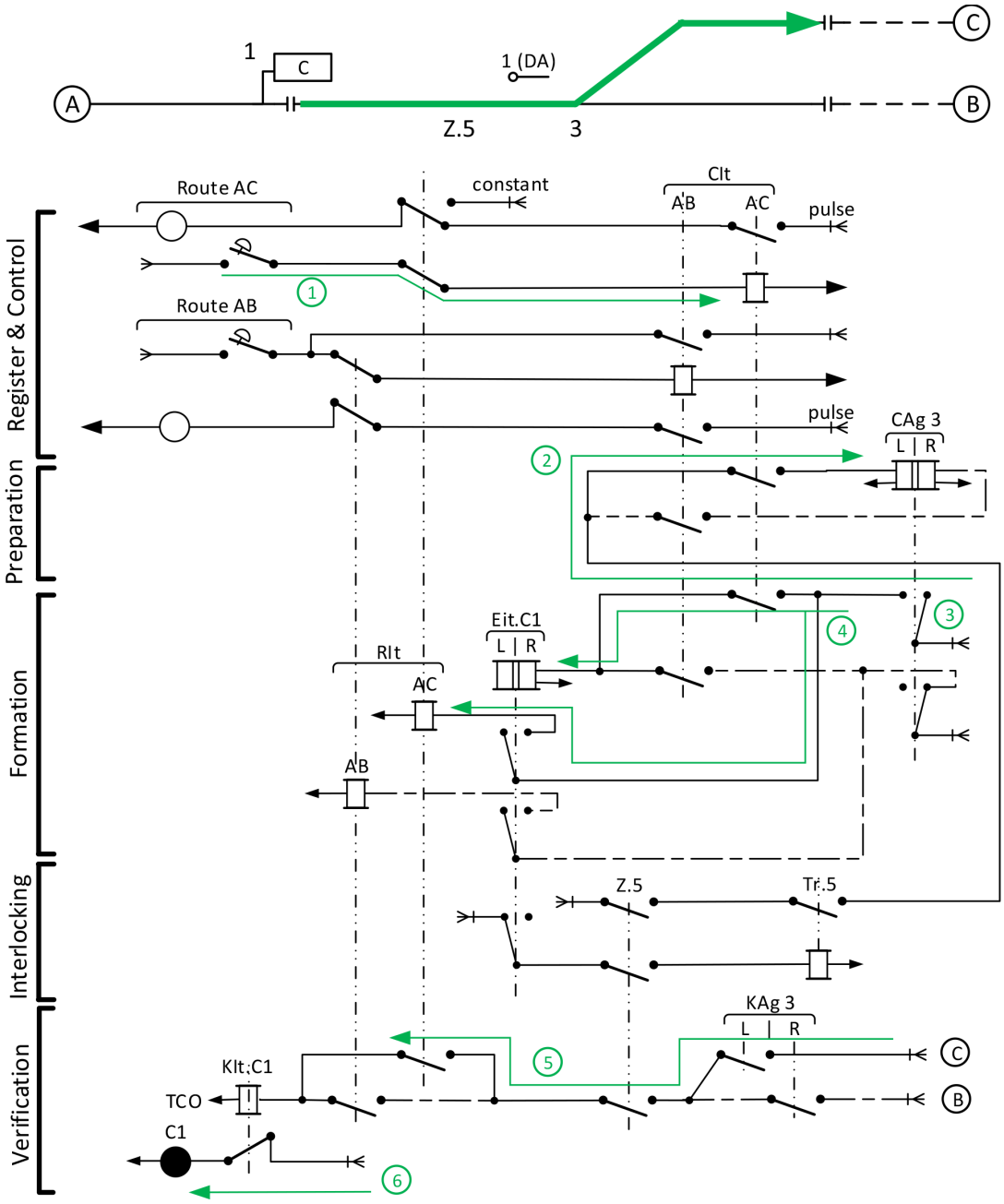


Figure 2.12 – An example of PRS type railway interlocking system

locked, can maintain blocking of this track segment. If this track segment has a point, the interlocked relays can also maintain the position of the point, in order to maintain the formation of the route. Furthermore, the blocked "transit" of this route prevents other conflicting routes from being formed and switches off all the conflicting signal lights. The release of a "transit" depends on either the manual destruction control of the route, or the release action from upstream track segments. Only when both transition relays of a track segment are released, can this track segment be re-used by other routes, and the position of the point be modified.

Fig. 2.12 presents the fixed installation of a simple point, which shows a circuit diagram

of the main units for a route establishing procedure. In this figure, the green arrows are used to indicate the process of establishing the route AC. The realization of the route is the implementation and completion of each phase:

1. Order a route: press the button ⇒ Relay *CIt.AC* could be activated
2. Register the route: relay *CIt* related switches change statuses
3. Preparation of the route: relay *CAg* changes to right position
4. Interlock the route by transit relay: relay *Elt* and relay *RIt.AC* is activated ⇒ *Tr.5* is deactivated in order to block the route AB
5. Verify the formation of the route: relay *KAg* ⇒ *KIt* could be activated
6. Open the entrance signal light: the signal light C1 is switched on (change to green)

In summary, the functions of a interlocking system are activated in successive phases. which take into account the positioning of the points, the occupation of the track circuits, the successive actions of internal relays and switches.

Part II

Methodology Construction

Chapter 3

Formal modelization for railway interlocking system via hierarchical coloured Petri net

In this chapter, we study the modelling of the French railway interlocking system using hierarchical coloured Petri net.

3.1 Introduction

A railway interlocking system is a collection of associated devices, complying with specific interlocking principles. The purpose of the RIS is to maintain transit safety by connecting and arranging the points and signals, so that a hazardous condition cannot arise. At present, computer-controlled relay-based interlocking systems still occupy a large proportion of the practical application in the French railway system. They have the advantages of being less dependent on human operators and faster self-checking ability, but their complex sequence of checks and consequent actions makes the railway interlocking a large concurrent system, which has a potentially complex total behaviour. Hence, detailed verifications and validations are essential before the system is put into service. As a result, the formalization of RIS becomes important to both the development of the computer-controlled interlocking system and the third-part testing of the RIS facilities.

Another strong motivation is the implementation of ERTMS. There are some critical aspects and many detailed application decisions that will impact on the railway network once the system is in service [Murphy 2007]. One of the issues is the interlocking system which is pertaining to each country and not included in the ERTMS specifications. Another issue is that according to the French national transportation department, before applying the new ERTMS system to an existing railway line, a safety assessment must be done in order to prove safety equivalence between the former system and the new one. However,

the current signalling system and its RIS have withstood the test of time, and are the fruit of long-term improvements and modifications. To prove the safety equivalence of the new system without any practical experience, the formal method can be a reasonable solution for this problem. As a primary step, the formal modelization of RIS is needed as a fundamental part of this task.

In order to maintain high-level security of RIS, IFSTTAR is leading a project to develop the formal verification of the French RIS in the context of national rules and the influence of implementing European ERTMS rules on the original system [Collart-Dutilleul et al. 2014; Sun, Collart-Dutilleul, and Bon 2014]. My research contributes to the fundamental parts of the project. It focuses on the formal specification of the French RIS, which is based on the PRCI[1] type system.

This chapter is organized as follows:

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 coloured 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.

3.2 Preliminaries of coloured Petri net

In this section, we begin with the background of Petri net (PN), and discuss the reason for choosing PNs as research tools. Then, we review basic concepts and notation of coloured Petri net (CPN) [Jensen 1981, 1987], including non-hierarchical CPN, hierarchical CPN, CPN with priroitized transitions and the analysis method of CPN tools: the state space.

[1] PRCI (relay-based computer-controlled control, french: **P**oste d'aiguillage à **R**elais à **C**ommande **I**nformatique)

3.2.1 Background

Petri net theory was first introduced in Germany in 1962 by C.A.Petri and documented in his Ph.D. thesis [Petri 1962]. His work has attracted the attention of several research groups and his thesis has been translated into English [Petri 1966].

Petri nets [Holloway, Krogh, and Giua 1997a; Murata 1989; Peterson 1981; Reisigs and Rozenberg 1998] are the technique suited to the specification and development of concurrent and distributed systems. Nets can model systems at different abstraction and conceptual levels. This technique also provides the foundation of a straightforward and intuitive graphical notion for specifications. These clear notations make it possible for a system to be visualized by its corresponding net model. Moreover, all the executions of Petri nets are performed by a rigorous and explicit algorithm process, which is based on a complete mathematical theory. So, they could be used for some formal proofs.

In general, a Petri net comprises a bipartite directed graph consisting of a set of places, a set of transitions, a set of arcs (connecting transitions to places or places to transitions), together with the associated annotations and an initial marking. Places can contain tokens (data values). Distribution of tokens to places is known as the marking of the net. An initial marking is the initial distribution. Typically, places represent system resources (called states), whereas transitions represent events. Transitions are enabled when sufficient resources are available. Enabled transitions can occur. The occurrence of a transition could change the state of the system.

Although Petri net is a formal, mathematical, well-developed theory, 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.

3.2.2 GRAFCET and Petri net

GRAphe Fonctionnel de Commande Étape/Transition (GRAFCET) is a method of representation and analysis of automation. This is a graphical tool for describing the behaviours 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. Promoted by the Agence Nationale pour le Developpement de la Production Automatise (ADEPA), it was accepted in 1982 as a French standard by the Association Française de Normalization (AFNOR) [Union Technique de l'Electricitè 1982]. In 1987, it was accepted as an international standard IEC 1131.3 by the International Electrotechnical

Commission (1EC) [IEC 1988]. The GRAFCET is also known as DFS (Diagramme Fonctionnel en Séquence) or in English, the SFC (Sequential Function Chart).

Later, GRAFCET became widely used in both industry and research areas. The main contribution of GRAFCET is that its formalism allows a clear modelling of inputs/outputs and of their relations. It also permits modelling of concurrency and synchronisation. This makes Programmable Logic Controllers (PLCs) more tractable and simplifies the simulation of the control logic of the system [Zaytoon and Villermain-Lecolier 1999]. Many PLC builders today use the GRAFCET as a specification and/or as a programming language. Large companies using it widely or recognising it as an internal standard, include Alstom, Renault, Siemens, Peugeot, Michelin.

Theory in short

The basic composition of a GRAFCET model [Giua and DiCesare 1993] is quite clear and simple: the steps, the transitions, the links, the actions and their associated receptivity. An example of the GRAFCET can be found in Fig. 3.1.

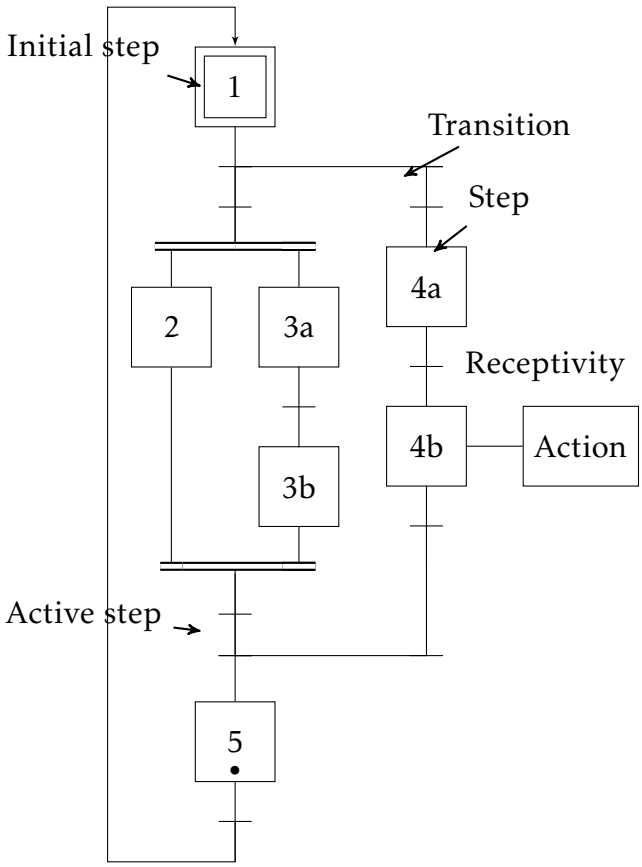


Figure 3.1 – Basic concepts of the GRAFCET

The basic components of the GRAFCET are:

The *steps* are represented by square boxes. A step may be active or inactive, which is represented by the presence of a token in the step. The state of a GRAFCET, at a given time, is defined as the set of the active steps.

Transitions are represented by horizontal bars. They represent the possibility of changing from one state to another. A transition is *enabled* when all immediately preceding steps are active. A transition may be *cleared* (or *fired*) if it is enabled and its receptivity is "TRUE".

The *links* are directed arcs joining steps to transitions and transitions to steps. They express the flow of control. The direction of the links is assumed to be downward unless an arrow specifies otherwise.

A *receptivity* is associated with each transition. Receptivity is a boolean expression that can have true or false values depending on the state or status changes of variables. The receptivity determines the "clear" act of the transition. A variable can be internal (state of stage timing) or external (PLC input).

The *actions* specify what must be done during the activation step. An action can be internal (counter, timer arming) or external (PLC output).

Moreover, there are five rules that are originally used to describe the evolution of a given GRAFCET [Zaytoon and Villermain-Lecolier 1999]:

R1 *Initialization*: Activate the initial steps at the beginning of system analysis.

R2 *Fire a transition*: If a transition is enabled and its receptivity is true, the transition will be fired.

R3 *Activate steps*: Firing a transition will immediately lead to the activation of all the downstream steps.

R4 *Concurrency*: Several simultaneously enabled transitions can actually be fired simultaneously.

R5 *Simultaneously Activate and deactivate a step*: During a transition, if a step is simultaneously active and inactive, it will remain active.

From GRAFCET to Petri net

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 criticised 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 analysed 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 3.1.

Table 3.1 – Structure comparison between GRAFCET and Petri net

GRAFCET		**Petri net**
Step	⇔	place
transition	⇔	transition
link	⇔	arc
receptivity	⇔	guard
action	⇔	auxiliary place

With Table 3.1, the model in Fig. 3.1 can be transformed into a PN model as shown in Fig. 3.2. Their notation formalism is so close that the engineer who is 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].

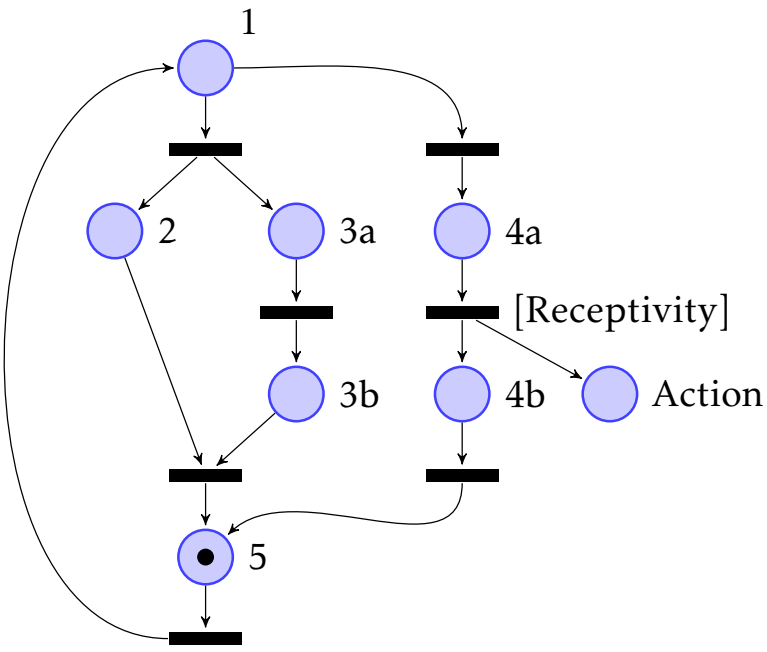


Figure 3.2 – Corresponding Petri net model

The major difference between these two languages is the mechanism of "concurrency". The GRAFCET allows the 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].

3.2.3 Coloured Petri net

Coloured Petri nets (CPNs) are a backward compatible extension of Petri nets. PNs were developed by Prof. Kurt Jensen [Jensen 1981, 1987, 1996] during late 1970s and 1980s. The definitions of PNs are based on distinguishable tokens being arbitrary complex data values and with some concepts from high-level programming language— the Meta language (ML).

The CPN maintains the properties of classical PN and provides a new formalism to allow the distinction between *tokens*. Then, each token is no longer noted with indistinguishable black dots, but with a data value which is named as the *token colour*. Each place in the CPN models can be marked with different tokens, but with the same type which is called the *colour set*. Arcs and transitions are inscribed with symbolic expressions. The arc expressions on the arc going into or out of the place are evaluated, and become a *multi-set* of tokens of the same type as the colour set of the place. There are two types of inscriptions that may be associated with a transition: the guard expression and the code segment. The *guard* expression is a boolean expression that is evaluated and becomes *TRUE* or *FALSE*. The *code segment* contains ML code, and is executed when its parent transition occurs. It is utilized to do some calculation and may bind the variables located on the output arcs. The state of a CPN model (*marking*) is composed of all the tokens including both their numbers and their colours.

In the late 1980s, the hierarchical concept was introduced to PNs [Huber, Jensen, and Shapiro 1991; Jensen 1996]. It provides a mechanism to formally integrate a set of smaller PNs to construct a larger CPN.

In the 1990s the concept of transition priority was introduced to PNs [Bause 1997; Best and Koutny 1992], and later this function was integrated into CPN tools [Westergaard and Verbeek 2011, 2013]. The transition priorities can be a useful mechanism when describing priorities of concurrent events, such as an exceptional event.

The time attribute is also an important concept of PNs, which allows us to model and analyse temporal behaviours and performance of real time systems [Jensen 1996; Jensen and Kristensen 2009].

Commonly, PNs are used to investigate the behaviours of the modelled system. CPN models that are investigated in this thesis are CPN of non-time domain, including non-hierarchical CPN, hierarchical coloured Petri net (HCPN), and CPN with prioritized transitions.

3.2.4 Graphic notation and modelling ability of coloured Petri net

This section explains the graphic notation which is adopted to represent PNs and the associated concepts. With different examples, it also expresses the modelling ability of PNs. First, we use a non-hierarchical CPN example to introduce the basic components of the PNs. Then, we add hierarchical concept to the example to discuss the HCPN and its notations. Finally, we add prioritized transitions into the model to have a better description of the system. All the formal definitions can be found in Appendix A, or in the book, *"Coloured Petri nets: modelling and validation of concurrent systems"* [Jensen and Kristensen 2009] and in the paper of Westergaard and Verbeek [Westergaard and Verbeek 2011, 2013].

A non-hierarchical coloured Petri net example

For a better understanding of CPN notations, we have a toy example in Fig. 3.3. It is a small circular railway network of 7 sections (tracks) and 2 trains (A and B). The simple operating rule is that there must be a safe distance of one section between trains. That means, when moving to a new section, both this section and the next one should be empty. Trains are all moving towards the same direction. The corresponding CPN model is shown in Fig. 3.4

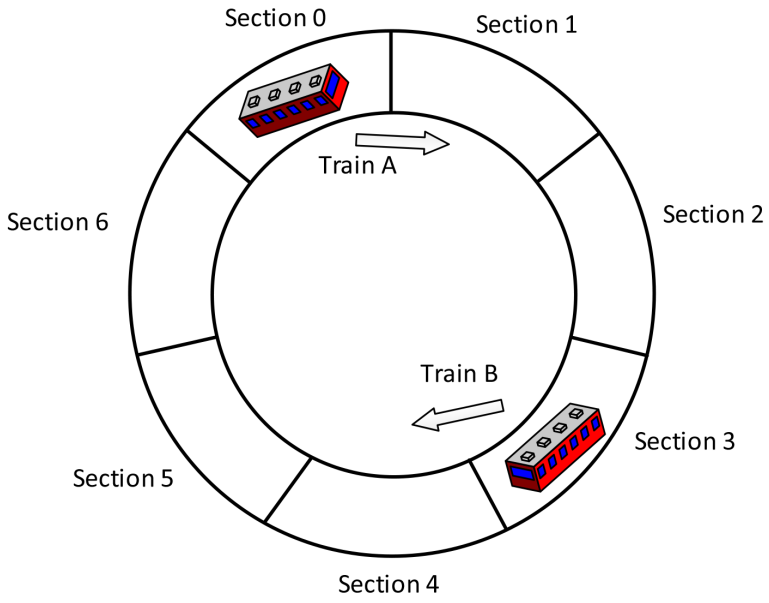
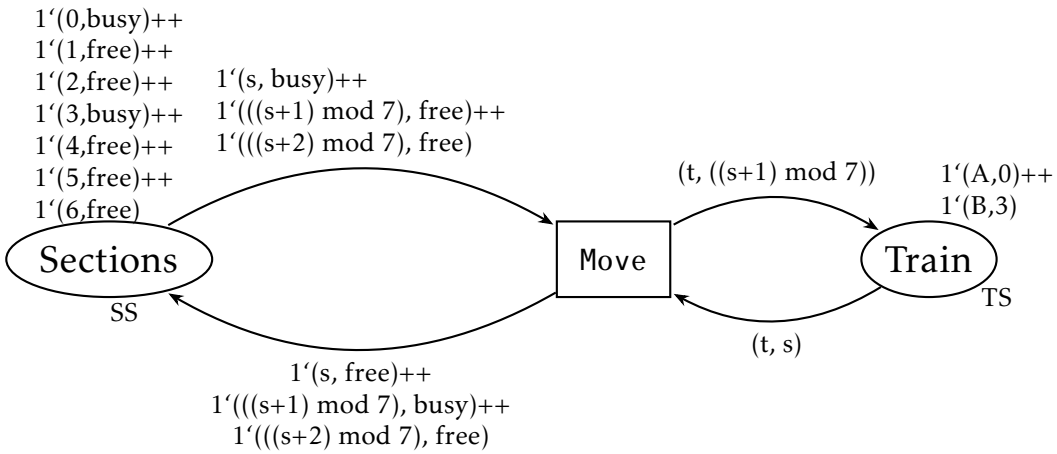


Figure 3.3 – A circular railway network

Coloured Petri net component CPN models have two parts: the Declarations and the net graphs which consist of places, transitions, arcs and associated inscriptions.

Declarations: each CPN model has a set of declarations, which declares a number of colour sets, functions, variables and constants. Once declared, they may be used in the net inscriptions of the corresponding net, in particular in the guards, arc expressions and transition code segment. All these declarations use *"a language called CPN ML, which is*



Declarations:

```
val n=7;
colset Section = int with 0..n-1;
colset State = bool with (busy, free);
colset SS = product Section * State;
colset Train = with A|B;
colset TS = product Train * Section;
var s : Section;
var t : Train;
```

Figure 3.4 – A non-hierarchical coloured Petri net example

relatively close to the style of declarations used in ordinary high-level programming languages" [Jensen 1996]. In Fig. 3.4, the declarations define 1 constant, 5 colour sets and 2 variables.

CPN ML language: The CPN ML language is based on the functional programming language *Standard ML* (SML) [Milner 1997]. *"It embeds the Standard ML language and extends it with constructs for defining colour sets and functions, declaring variables, and writing inscriptions in CPN models"* [Jensen 1996].

Places: places are represented, as usual, by ellipses. The place name, the colour set and the initial marking are located close to their corresponding place. The initial marking is also a multi-set, and it may be a union of several multi-sets. In Fig. 3.4, there are two places. Place *Train* have the colour type of *TS*, which indicates the train name and its location. Place *Sections* with colour type *SS*, represent the circular tracks and their occupation status.

Transitions: transitions are represented by rectangles. They may contain the transition name, the transition condition (guard function), the code segment and the priority. As this model is so small that we can put all the calculation on the arcs, the transition in this model has no further inscription with it.

Arcs: arcs are represented by arrows, and they are annotated by terms of the same type as their associated place. The arc expressions can be as simple as a constant (bool or enumeration value), a single variable, or comprise complex inscription involving functions. In Fig. 3.4, the arc (Train,Move) has an expression of a simple variable, while the expression of

arc (Section,Move) is a multi-set with functions.

State space graph The state space graph (SSG), or so-called reachability graph, is represented by a set of system states and the arcs between different states. The nodes of the SSG are noted by rounded boxes. Each node represents a reachable marking. The numbers in the node identify its ID and the number of predecessors and successors. Each arc represents the occurrence of a transition. The inscription on an edge indicates the name of a transition together with the particular bindings of variables.

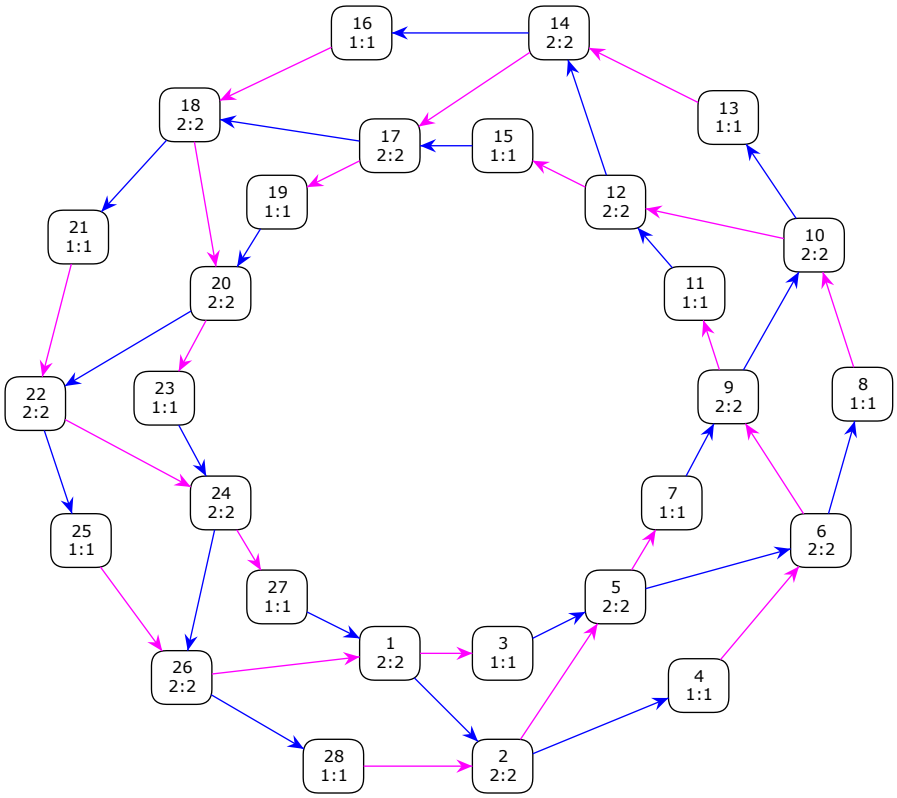


Figure 3.5 – State space graph of the non-hierarchical coloured Petri net example

The SSG generated from the example model of Fig. 3.4 is shown in Fig. 3.5. It contains 28 nodes and 42 arcs. The purple arcs are the movements of *Train A*, while the blue ones represent *Train B*.

Hierarchical coloured Petri net example

Now, we add more details in the example of Fig. 3.3, and get a new example in Fig. 3.6. We suppose that the drivers cannot drive the train without the movement authority (MA) from the signal centre. Each time the signal centre can send out only one MA. The generation of the MAs obeys the same operating rule as in the previous example. This new driving mode fully depends on and obeys the order of the MA. In other words, the drivers are "blind" for the situation of the front routes.

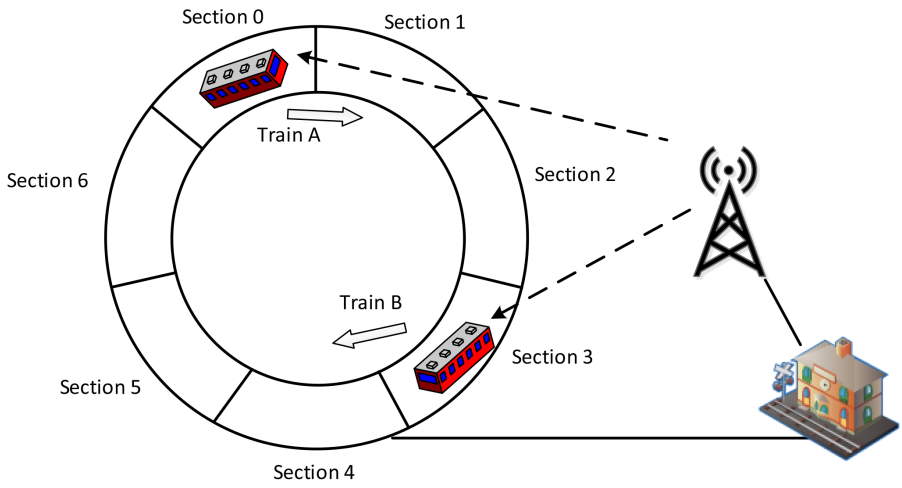


Figure 3.6 – A circular railway network with a signal centre

The corresponding HCPN model is shown in Fig. 3.7. The colour set *MA* represents the movement authority, which is generated in sub-module *Signal_centre* and sent to sub-module *Driver*.

In HCPN, the construction of CPN allows a net to be organised as a module, in a similar way to the process in which programming language is implemented into modules. With the modular structure, any large system could be divided into separate modules and their connection relationships could indicate an overview of the system. Conceptually, nets with modules are nets with multiple layers of detail. Such a hierarchical model is easy to handle when designers only need to concentrate on one small part of the system at a time. Moreover, some repeated components could only be defined once but used repeatedly. This is quite efficient and time-saving.

Substitution transitions: this mechanism can be achieved by breaking the CPN net into smaller pieces by utilizing the substitution transitions. A substitution transition can represent an entire piece of net structure. In Fig. 3.7, the transition *Driver* and *Signal centre* are substitution transitions. They represent the perspectives of the drivers and the signal centre.

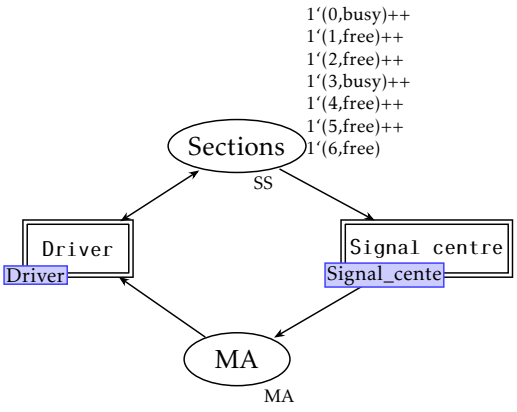
The token exchange between different layers is through *port–socket relation*, which relates the *port* places of the sub-module to the *socket* places of the substitution transition. The sockets are the places that are directly connected to a substitution transition, for example *Sections* and *MA* are sockets of both transition *Driver* and *Signal centre*. The ports are places that are associated with each socket. The port places are marked with blue tags, such as *In*-tag, *Out*-tag and *In/Out*-tag.

Fusion places: the fusion places are a set of special places so that anything that happens to each place in a set also happens to all the other places in the set. The set of the fusion places is called *fusion set*. The fusion places can exist on the same page of a net or on different pages. A fusion place is assigned with a fusion tag. Any fusion places that are marked with

the same fusion tag belong to the same fusion set, which means they are the same place.

Inhibitor arcs: the transition that is connected to the inhibitor arc can only fire if there is no token in the involved place. In Fig. 3.7, the transition *Control* has an inhibitor arc from place *MA*, which prevents *Control* from enabling if place *MA* contains a token.

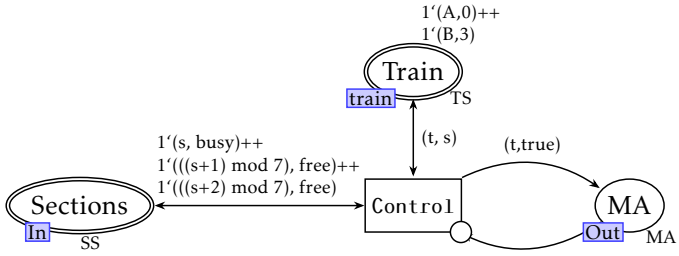
State space of HCPN model: the SSG generated from the example model of Fig. 3.7 is shown in Fig. 3.8. It contains 70 nodes and 84 arcs. The purple arcs and blue arcs represent the movements of *Train A* and *Train B*. The red arcs and cyan arcs indicate the different MAs that are generated for *Train A* and *Train B*.



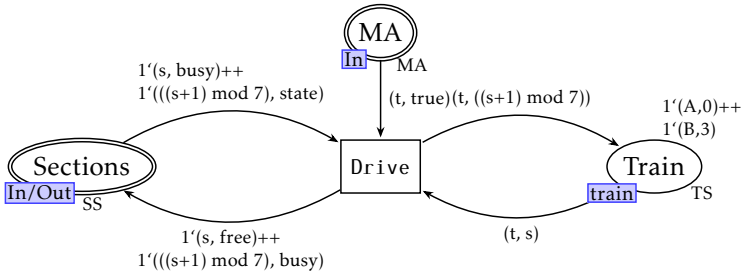
(a) Top layer

```
Declarations:
  val n=7;
  colset Section = int with 0..n-1;
  colset State = bool with (busy, free);
  colset SS = product Section * State;
  colset Train = with A|B;
  colset TS = product Train * Section;
  colset MA = product Train * BOOL;
  var s : Section;
  var t : Train;
  var state : State;
```

(b) Declarations



(c) Sub-model of Singal cnetre



(d) Sub-model of Driver

Figure 3.7 – The hierarchical coloured Petri net model of the circular railway with a signal centre

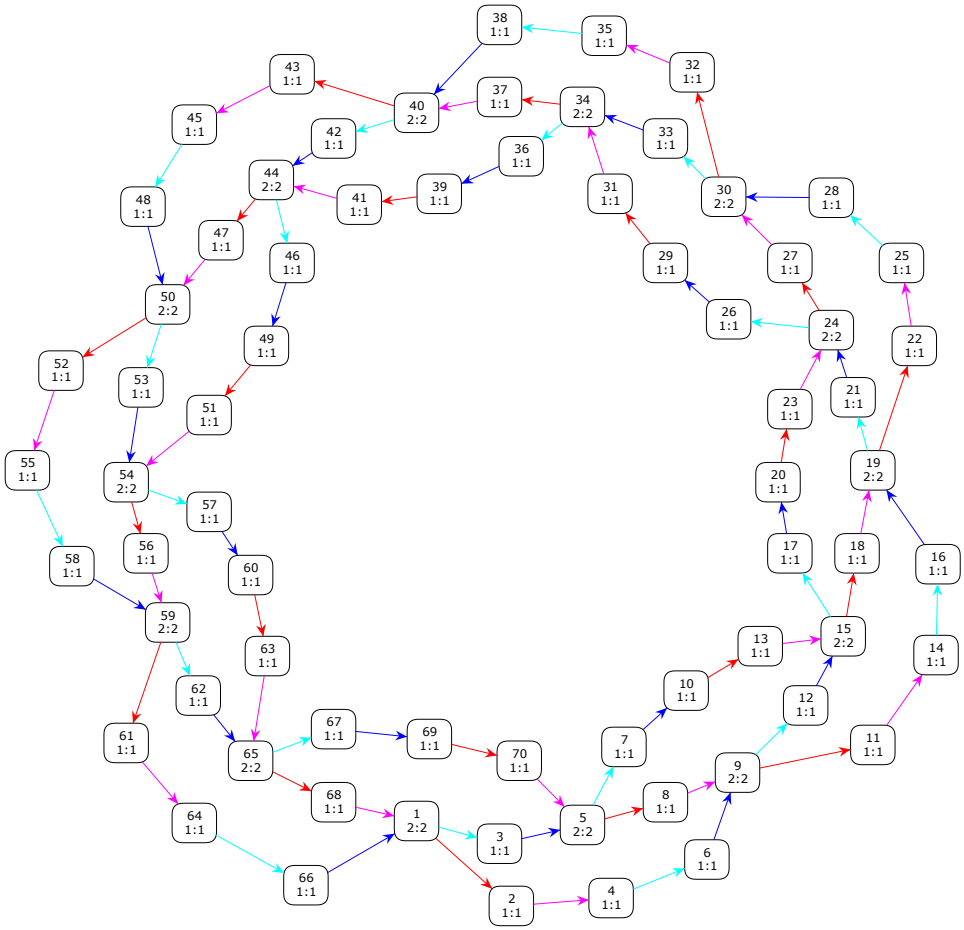


Figure 3.8 – State space graph of the hierarchical coloured Petri net example

Transition prioritized coloured Petri net example

In practice, not all trains have the same priority. Normally, the TGV[2] has higher priority than TER[3]. We suppose that *Train A* is a TGV and *Train B* is a TER. The new scenario is shown in Fig. 3.9.

The corresponding prioritized HCPN model is shown in Fig. 3.10. In this CPN model, we have assigned a priority tag "P_HIGH" to transition *Control A*, while the other transitions have the default priority "P_NORMAL".

The transition priority can be a useful mechanism when describing priorities of concurrent events. For example, a high-priority transition is forced to occur before other transitions if they are enabled. This is useful for handling exceptions. On the other hand, the low-priority transitions can be used to model background tasks that should only be executed when no other transition is enabled. Transition priorities are also useful for analysis, as the internal priorities could reduce the concurrency, leading to smaller state-spaces.

Transition priorities: for low-level PNs, static priorities are defined in [Best and Koutny 1992], which are *relational static* priorities. For high-level PNs, CPN Tools support the *abso-*

[2]TGV (high-speed train, french: **T**rain à **G**rande **V**itesse)
[3]TER (regional express transport, french: **T**ransport **E**xpress **R**égional)

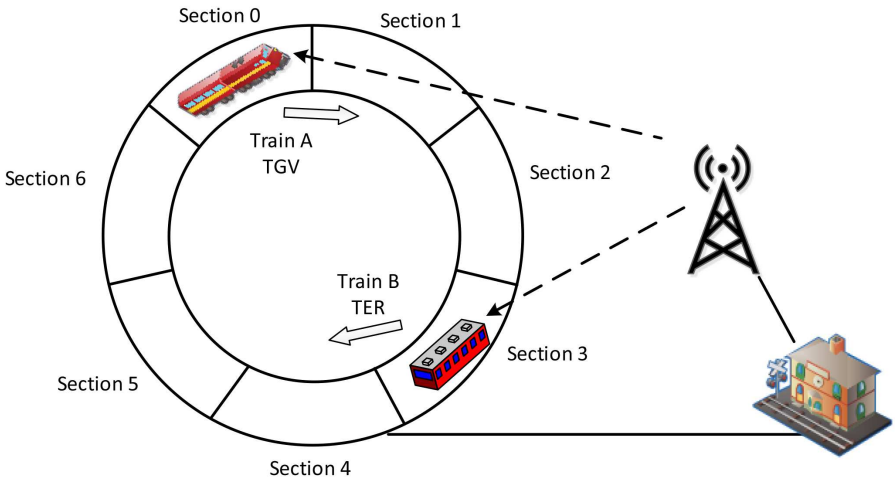
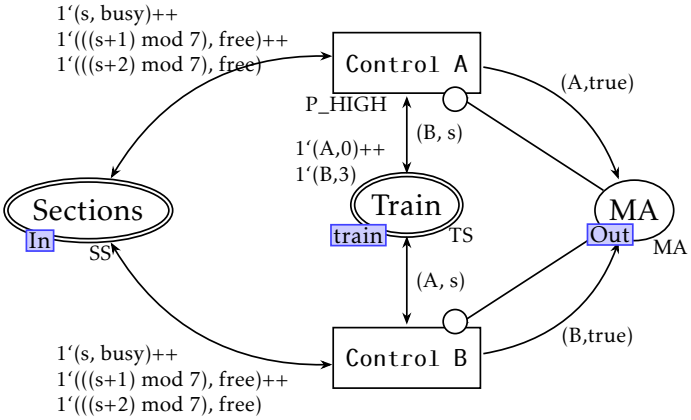


Figure 3.9 – A circular railway network with a signal centre and train priorities



(a) Prioritized singal centre module

Declarations:
val P_HIGH = 100;
val P_NORMAL = 1000;

(b) Declarations

Figure 3.10 – Prioritized hierarchical coloured Petri net model of circular railway

lute static priority concept, which is described in [Westergaard and Verbeek 2013].

We are using Kurt Jensen's definition form in our research, while the publication above is using other forms of PN definitions. To normalize all the definitions, we adapt and integrate the definitions of [Best and Koutny 1992; Westergaard and Verbeek 2013] into Kurt Jensen's definition framework. Then, we have the definitions as below:

Definition 3.1. Static Priority (**Def. 3.1** in [Best and Koutny 1992],**Def. 3** in Westergaard and Verbeek 2013) A static priority system is a pair (CPN, ρ), which:

1. $CPN = (P, T, A, \Sigma, V, C, G, E, I)$ is a CPN.
2. $\rho : T \to \mathbb{P}$ is a **priority function** that assigns a priority to each transition.
3. $\mathbb{P}$ is a finite set of natural numbers, called the **priorities** of the transitions.

The priorities in Definition 3.1 are global and do not depend on the binding of the transition. Intuitively, if $\rho(t) < \rho(t')$ then t' has priority over t. Precisely, it means that t' can be enabled before t. When dealing with such priority systems, we define that if a transition is enabled according to Definition A.5 in Appendix A, it is **pre-enabled**. Then, we can have the new enabling rules for priority systems as follows:

Definition 3.2. Enabling with Priority (**Def. 3.3** in [Best and Koutny 1992],**Def. 4** in [Westergaard and Verbeek 2013]). A transition $t \in T$ is enabled in the marking M if it is pre-enabled and no transition t' with $\rho(t) < \rho(t')$ is pre-enabled.

The algorithm for checking enabling with priority in CPN Tools is shown in Algorithm 3.1. First, it sorts all transitions according to priority and processes them highest-priority-first until it reaches t. If it finds a pre-enabled transition with higher priority than t, it returns false. If it does not find a pre-enabled transition with higher priority than t it returns the pre-enable status of t.

Algorithm 3.1 Algorithm for checking enabling with priority (**Algorithm 3** in [Westergaard and Verbeek 2013])

```
1: SortedTransitions← PrioritySort(T)
2: After any transitions or initialisation
3: procedure CheckEnablingPriority(t) is
4:     for all t' ∈ SortedTransitions do
5:         if ρ(t') > ρ(t) then
6:             if CheckEnabling(t') then
7:                 return false
8:         else
9:             return CheckEnabling(t)
```

In fig. 3.10, corresponding to Definition 3.1, each priority is a natural number that is defined in the declaration table in Fig. 3.10(b). In CPN Tools, the priorities are defined as Positive INT variables. The higher the number, the lower the priority. According to the model declarations, priority relation is P_NORMAL < P_HIGH.

The state space of the CPN model example is shown in Fig. 3.11. It contains 28 nodes and 28 arcs. Compared to the SSG in Fig. 3.8, it is clear that prioritiezd transitions can exclude some occurrences, which could effectively reduce the size of state-space. The result means the technique of prioritized transitions is useful for analysis, as we can assign some internal transitions with higher or lower priority, in order to preempt or relinquish the order of concurrency, leading to smaller state space.

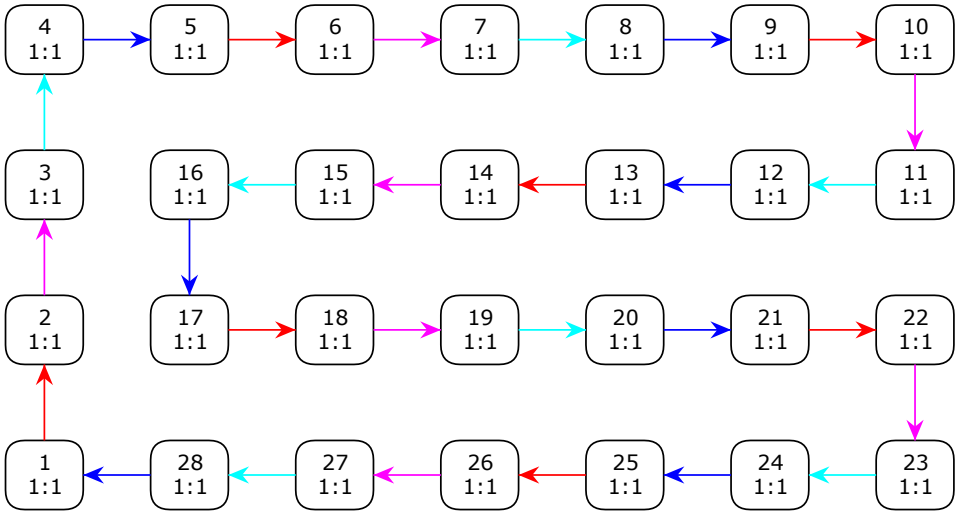


Figure 3.11 – State space graph of the prioritized hierarchical coloured Petri net model

3.3 Initial coloured Petri net specification of railway interlocking system

As we have discussed in Sections 2.4 and 2.5, the railway interlocking system plays a vital role in the safe transportation of a railway 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 signalling rules or the incompatible international standards.

The modelling framework of the whole railway interlocking system could be divided into three parts: the signalling operations, the fixed installations and rolling stock, as in Fig. 3.12. 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, actions on track circuit). In response to train requests, the signalling operations send certain commands to fixed installations(such as points and signal lights) according to its operating principles.

- **Signalling 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** includes track segments, points, signal lights, and other automatic facilities that could be self-acting without the instruction from the train controlling centre. 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

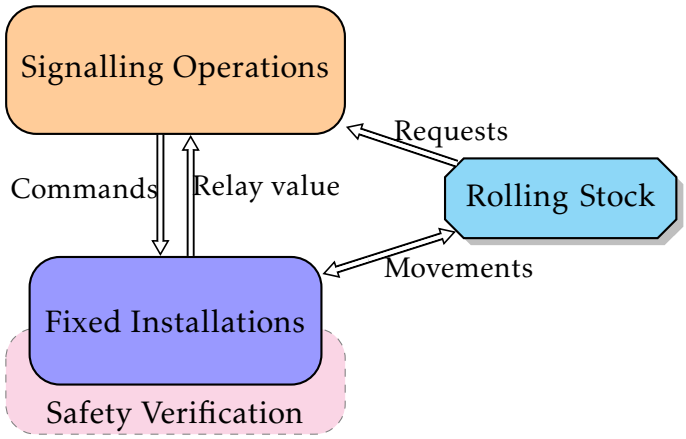


Figure 3.12 – Specification framework of railway interlocking system

and operating instructions.

PNs are a powerful formal tool that have been applied to many railway applications. Their formal definition can be found in Appendix A. Considering the large scale and the space complexity of interlocking systems, one feasible solution is to model the RIS by HCPN. The signalling 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 coloured token which 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 signalling operations and the fixed installations. The net $N^o = (P^o, T^o, A^o, \Sigma^o)$ represents the *operation* part, which implements the route management process and movement authority control. The net $N^i = (P^i, T^i, A^i, \Sigma^i)$ represents the *installation* part, where the train movements are realized by the transitions $t \in T^i$. The notation $N^s = (P^s, T^s, A^s, \Sigma^s)$ denotes the *supplemental* part, which 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 3.3. A basic *unmarked* RIS net is a connected Petri net $N_{RIS} = N^o \cup N^i \cup N^s$, where: $N^o \cap N^i = P_{equip}$, N^o should not be a empty set $N^o \neq \emptyset_{pn}$ 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, position of points. They perform the role of indicator in the operation nets and conduct the train movement in the installation nets.

3.4 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, Collart-Dutilleul, and Bon 2014]. The basic hierarchy of the HCPN model framework is described in Fig. 3.13. The *Main* net is the topmost net, which is the carrier of the whole model, "storing" all the subsystems and their interactions.

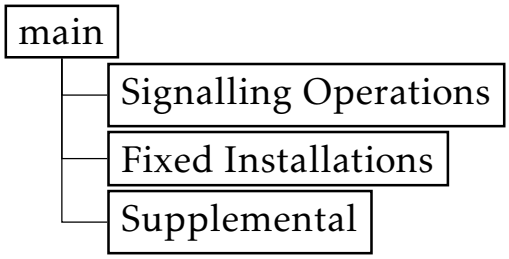


Figure 3.13 – Basic specification framework of railway interlocking system

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

3.4.1 Signalling operation specification

RIS signalling 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, Lemmer, Horste, et al. 1997; Janhsen, Lemmer, Ptok, et al. 1997; Jansen, Meyer Zu Hörste, and Schnieder 1998] could give us some useful inspiration. In such systems, there are three aspects which 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 signalling operations and the fixed installations. Individual scenarios are categorised 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 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 sub-nets can be reused.

Vertical decomposition model

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, to 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 signalling operations, a layered approach, proposed by [Janhsen, Lemmer, Ptok, et al. 1997; Jansen, Meyer Zu Hörste, and Schnieder 1998] is adopted. Dynamics and functionality are expressed by both scenarios and functions. Scenarios show the behaviours of the system in its external environment, which means the railway operation context. Functions can process data received form external components or internal ones. The difference between scenarios and functions is that functions are not subordinated to any scenarios. They are independent of scenarios and can be used within arbitrary scenarios.

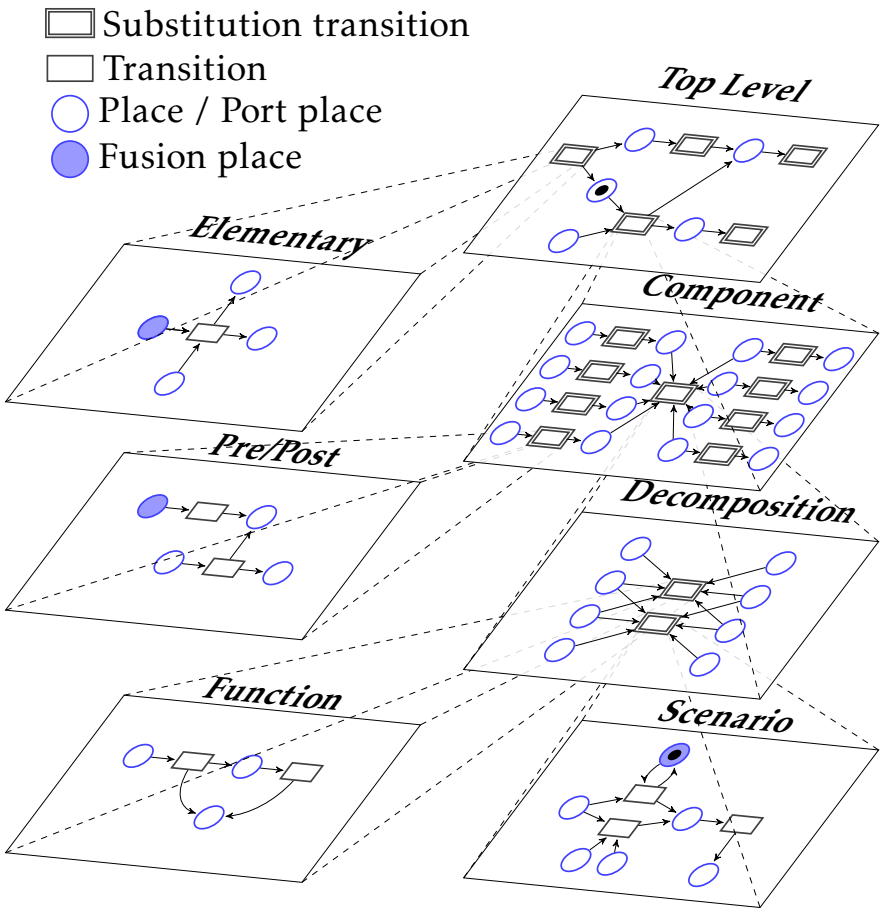


Figure 3.14 – Hierarchical model structure of signalling operation

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.

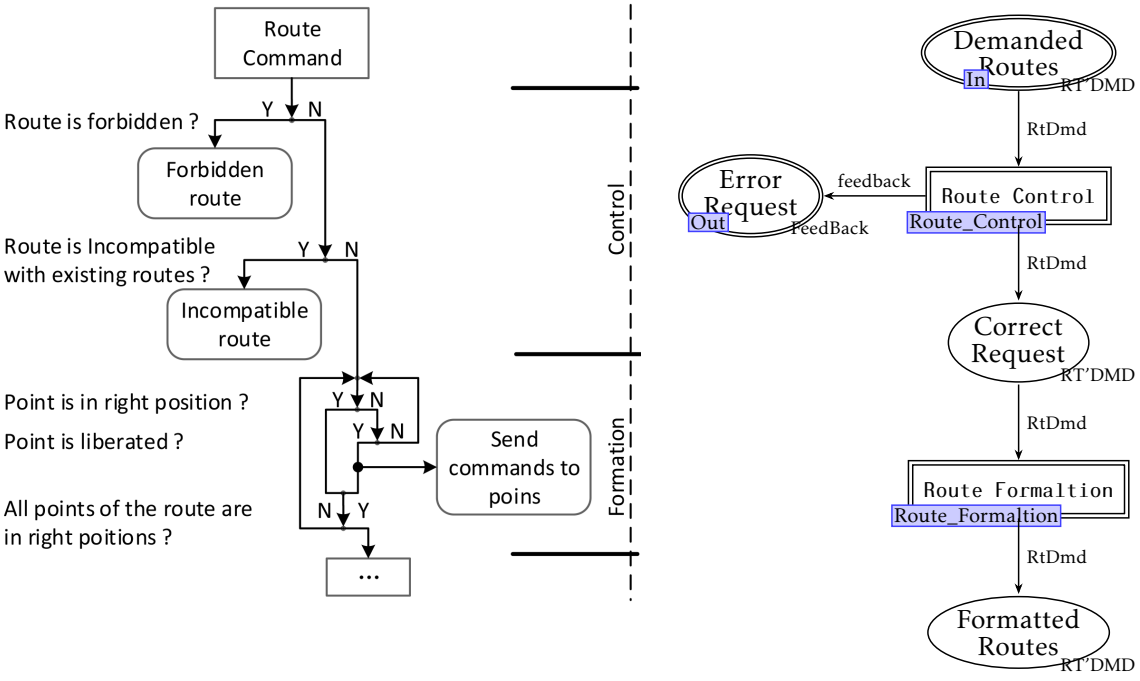
The corresponding vertical decomposition model is in Fig. 3.14 with several levels.

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 the relations between different components. They are used for preprocessing incoming messages and postprocessing outgoing messages.

Examples of signalling operation

To illustrate how to map from signalling operation to the CPN model, here is a small demonstration of the Route formation procedure, an important part of the interlocking opera-

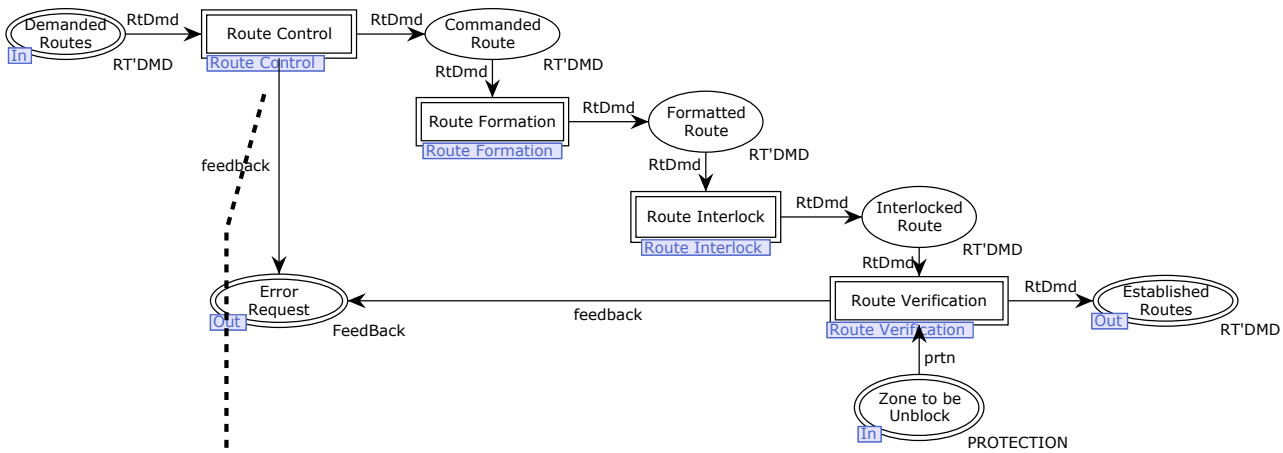


(a) Route establishment flow chart (b) Corresponding HCPN model

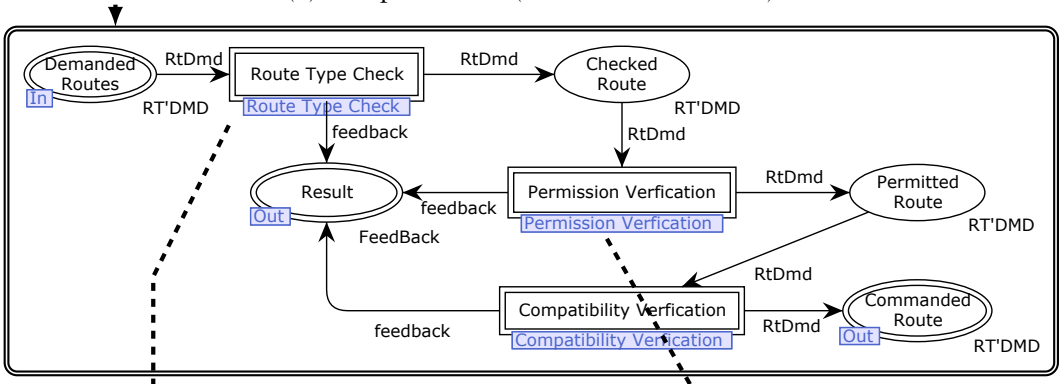
Figure 3.15 – Example of mapping signalling operations (1)

tion. 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. 3.15(a), There is the control flow chart. It receives the route control instruction (route command), and checks whether this instruction is feasible and compatible with existing ones. Then it will format the route according to its formation information, such as the positions of points.

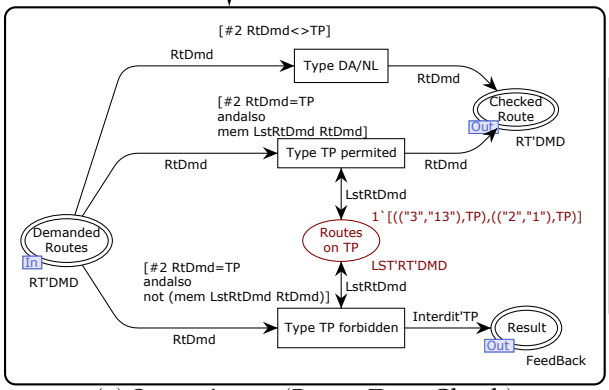
The first 2 consecutive actions of interlocking route establishment are: *control* and *formation*. The control part validates the input of "Route control" instruction and acts as a filter.



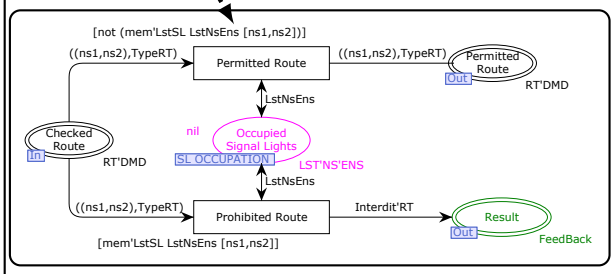
(a) Component net (Route Establishment)



(b) Decomposition net (Route Control)



(c) Scenario net (Route Type Check)



(d) Function net (Permission Verfication)

Figure 3.16 – Example of mapping signalling operations (2)

Only when the requested route is satisfied with current interlocking status, is it 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[4] route). This means that only when a route is partially destroyed because of the use of flexible transit[5], can the corresponding initial signal 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 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, will an instruction be sent to the fixed installation model to change the point. After receiving the new position, the procedure continue 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. 3.16.

Fig. 3.16(a) 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. Fig. 3.16(b) is the decomposition net of route command procedure. It still contains sequences of functions: route type check – permission verification – compatibility verification. Fig. 3.16(c) is a scenario net, because the place "Routes on TP" contains the configuration of a certain scenario. Fig. 3.16(d) 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 signalling 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.

3.4.2 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, Fantechi, and Gnesi

[4]Automatic destruction : A typical French interlocking route type that could be destructed by the passage of the train

[5]Transit souple: a flexible interlocking mechanism, which enable a route to destruct partially after the passage

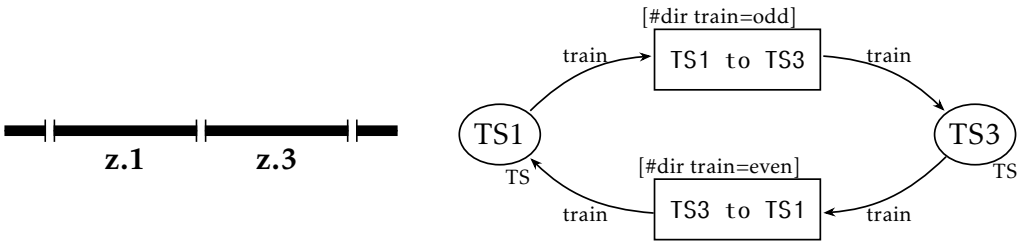
2004]. Its geographical structure allows us to slice the whole railway layout into independent and distributed components that can be individually modelled and physically located next to their relevant units.

The basis railroad components

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, which 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 signalling 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 2 aspects: red (stop intermediately), green (route clear).

Both track and point are referenced as atomic components, which could form the geographical 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.

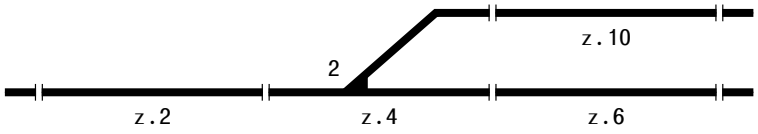
Track segments Fig. 3.17 shows a demo of PN of two successive track segments. Each place represents a track segment. Two transitions move train tokens between the two segments, depending on the direction of the train and supervision by the guard function of the transitions. The direction from left to 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.



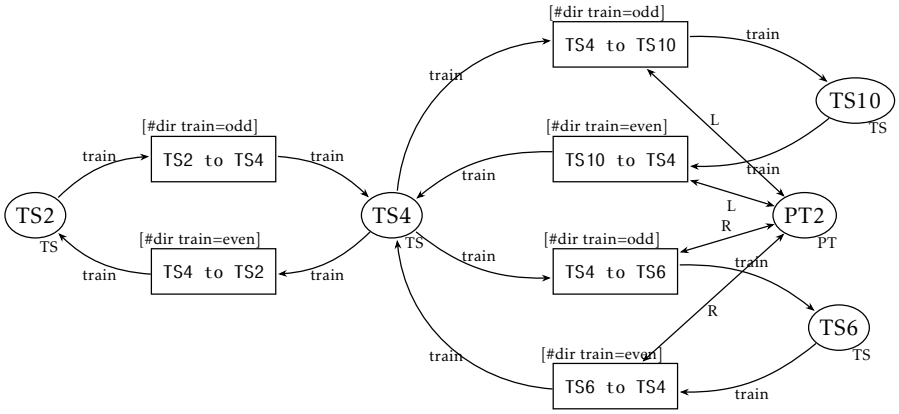
(a) Track segment demo (b) Corresponding CPN model

Figure 3.17 – A Petri net representation of track segments

Points Fig. 3.18 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. 3.18(a). In its corresponding model, the point is represented by a single place which 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 (see the schematic diagram in Fig. 2.7(a)). This point



(a) Point demo

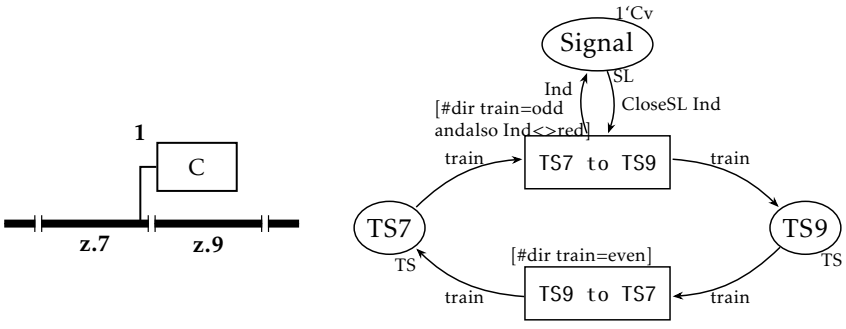


(b) Corresponding CPN model

Figure 3.18 – A Petri net representation of point component

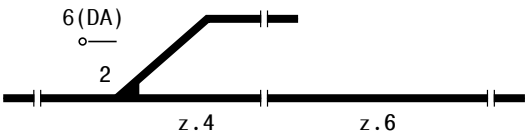
place works as a condition place of 4 transitions (movements). However, its position will not affect the movements between *TS2* and *TS4* as they are constantly connected.

Signal lights Fig. 3.19 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. 3.19(a), the movement from *TS7* to *TS9* is controlled by signal light. So, in Fig. 3.19(b), *Signal* place is only connected to the transition "TS7 to TS9". This transition is only enabled when the token (indicator colour) 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 (≠)".

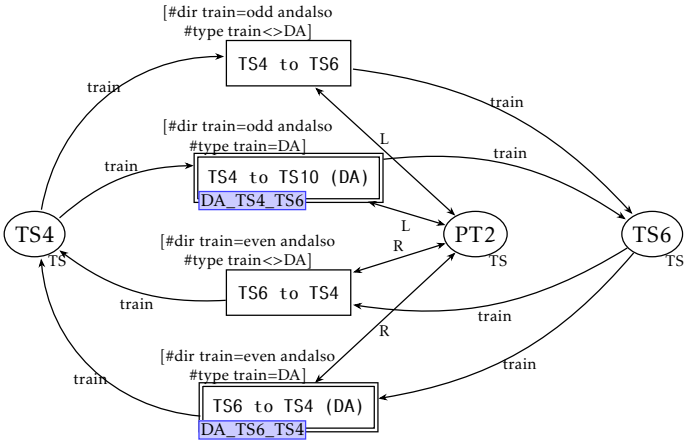


(a) Signal light demo (b) Corresponding CPN model

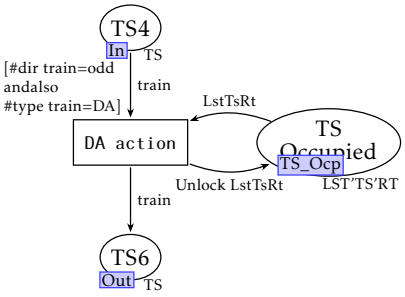
Figure 3.19 – A Petri net representation of signal light



(a) DA mode



(b) Corresponding CPN model



(c) Sub-model of DA_TS4_TS6

Figure 3.20 – A Petri net representation of "DA" mode

Automatic unlock devices In the French system, there is a ground-based automatic mechanism, which 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 signalling operations. The conditions of establishing a DA mode interlocking route are:

- there is a pedal (see in Fig. 3.20(a)) 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. 3.18(b), 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. 3.20(c)).

Example of geographical approach

In our research, a typical station from the *French railway signalization* book [Rétiveau 1987] is studied, shown in Fig. 3.21. It is only half of the station, which 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 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, Collart-Dutilleul, and Bon 2014].

The whole layout is represent by the CPN model in Fig. 3.22, 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 Section 3.4.1, 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.

3.5 A pattern of railway interlocking modelling

An RIS has two main parts: the signalling operations and the fixed installations. In each station, signalling 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 Section 3.4.1.

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, Collart-Dutilleul, and Bon 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.

3.5.1 Generalisation 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.

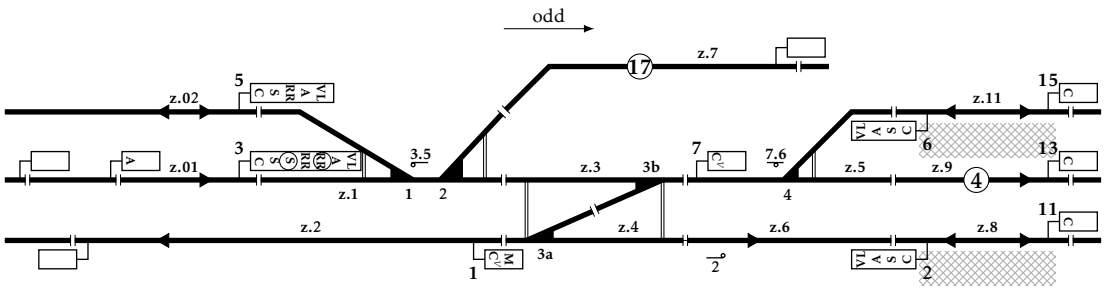


Figure 3.21 – Case study of a station layout

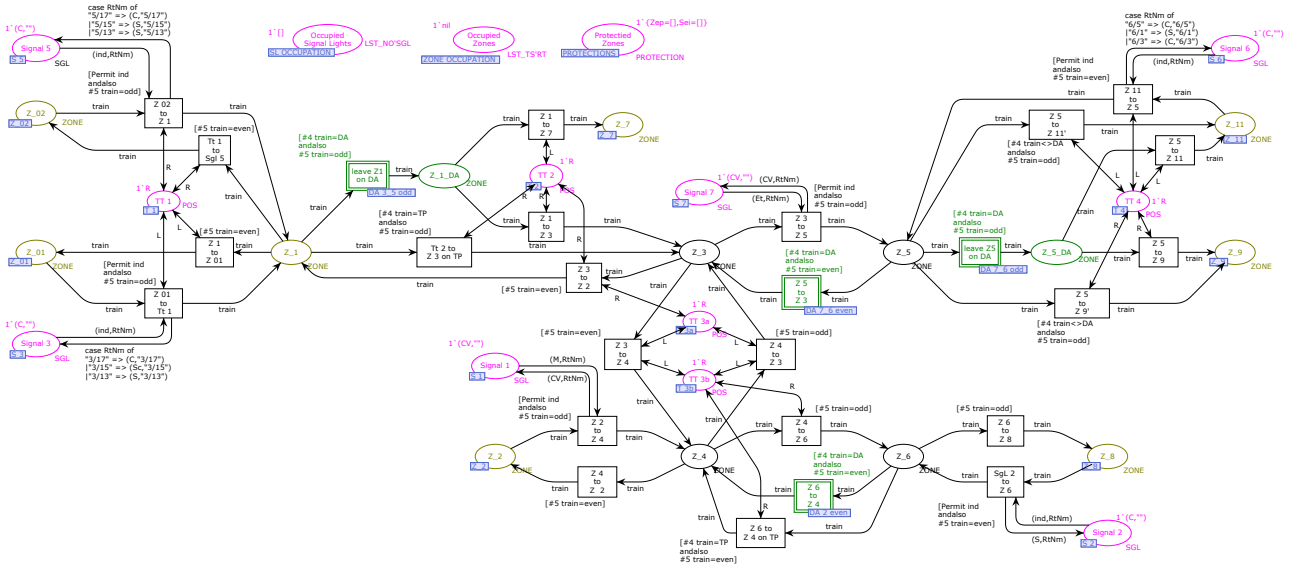


Figure 3.22 – The Petri net model of route layout

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 coloured 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. 3.17, the new model in Fig. 3.23 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 the train token will be refreshed according to the enabled binding elements. The "track connection" place is the constraints 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. 3.24 is the corresponding

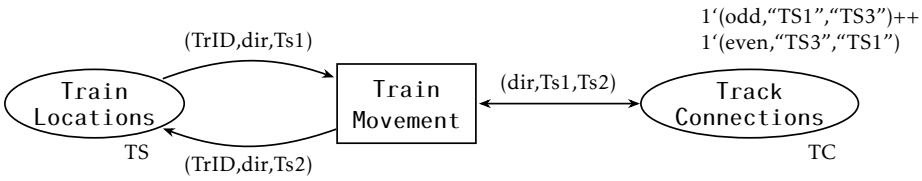


Figure 3.23 – Generalized representation of track segments

model of the example in Fig. 3.18. The new colour set of *TC* contains the point constraints. Only when the point stored in the *Point List* place satisfies the point constraints, can the train move.

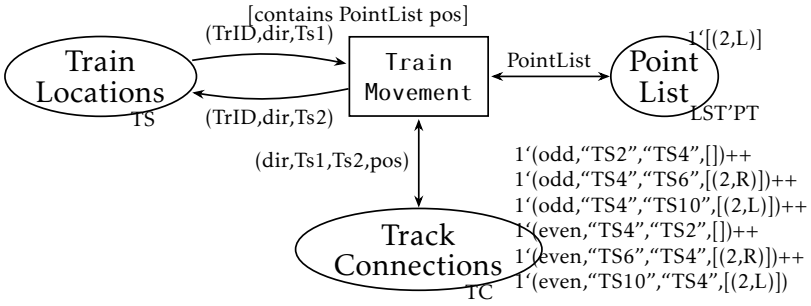


Figure 3.24 – Generalized representation including points

Adding signal lights Similar to a point, a signal light is also the movement constrain. So the introduction of signal lights comes with a new place and a modification to the colour set of *TC*. The new model in Fig. 3.25 is the corresponding model of the example in Fig. 3.19. 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.

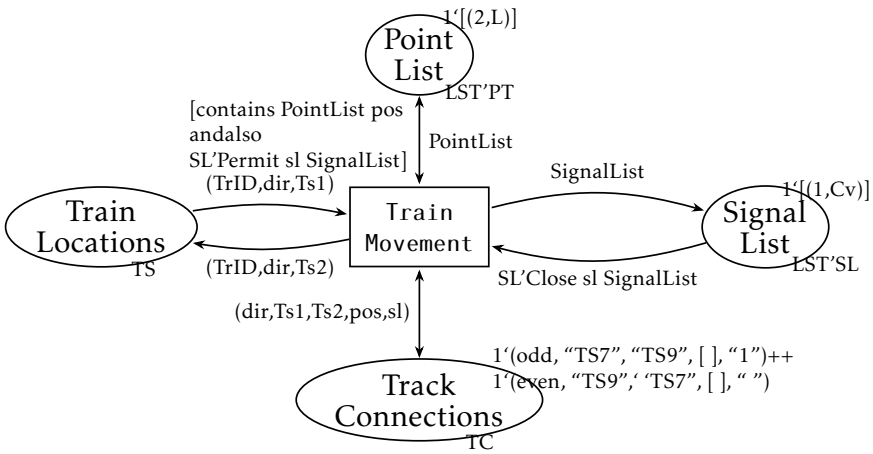


Figure 3.25 – Generalized representation including signal lights

From the above examples, we can conclude that the components of railroad and their combinations can be expressed by generalized structure, using constrain 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 3.2, train, track, point and signal light are normal components that we have introduced in 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.

Table 3.2 – 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 colour)	LstSl
Destruct Auto	exiting track (where DA take place)	TsDa
	effective direction of pedal	DirDa
	tracks to be destructed	TsLstDa
	signal light to release	SlDa
	points to release	PtDa

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[6]. 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:

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

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

1. Release tracks of the route behind the train;
2. Release points of those tracks;
3. Switch off signal light (if any) after passing.

For analysis purposes, we introduce a security guard function, which 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 the Table 3.3. With the help of *CPN ML* language, all the conditions above can be embedded into one transition and can combine into a single model to represent all the DA mode movements.

3.5.2 Example of generalized model

The study case of Fig. 3.22 is modelled by the generalisation concept above. The complete CPN model provides a pattern which 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 4 types of route modes along with their mixed traffic operations. The whole model is really large for a demonstration. Only one layer of the model and its result will be introduced. The other parts of the model are built by successive implementation.

Fig. 3.26 shows the DA module of the general structure, which includes all the necessary elements mentioned before: tokens of train, track segments, track connections, points,

[6]DA: Destruction automatique, TP: Tracé permanent

Table 3.3 – Conditions and equations of “DA” movement

Condition	equation
Route type	TpRt = DA
	Ped=TRUE
Route formation	PosTr = CurTs
	DirTr=DirTs
	PtTs ⊆ LstPt
Singal open	(NmSl,green) ⊆ LstSl
Movement authority	PostTs∈MA
DA activated	TsDa = PosTr
	DirDa = DirTr
To release	TsLstDa
	SlDa
	PtDa
Security check	Ocp of CurTs = Occupied
	Ocp of PostTs = Clear

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 parameter of simulation (Table 3.4):

Table 3.4 – Initial configuration of the model

Train demand route “3/15”	1‘{NmTr=“TER-0315”, DirTr=odd, NmRt=(“3”,“15”), TpRt=DA, PosTr=“”, MA=[]}
List of all point	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 3.5. 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.

Then we use the state space analysis function that is embedded in CPN tools to analyse the space state of this simulation. Its calculation result shows that this “single train” scenario has 26 state 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”.

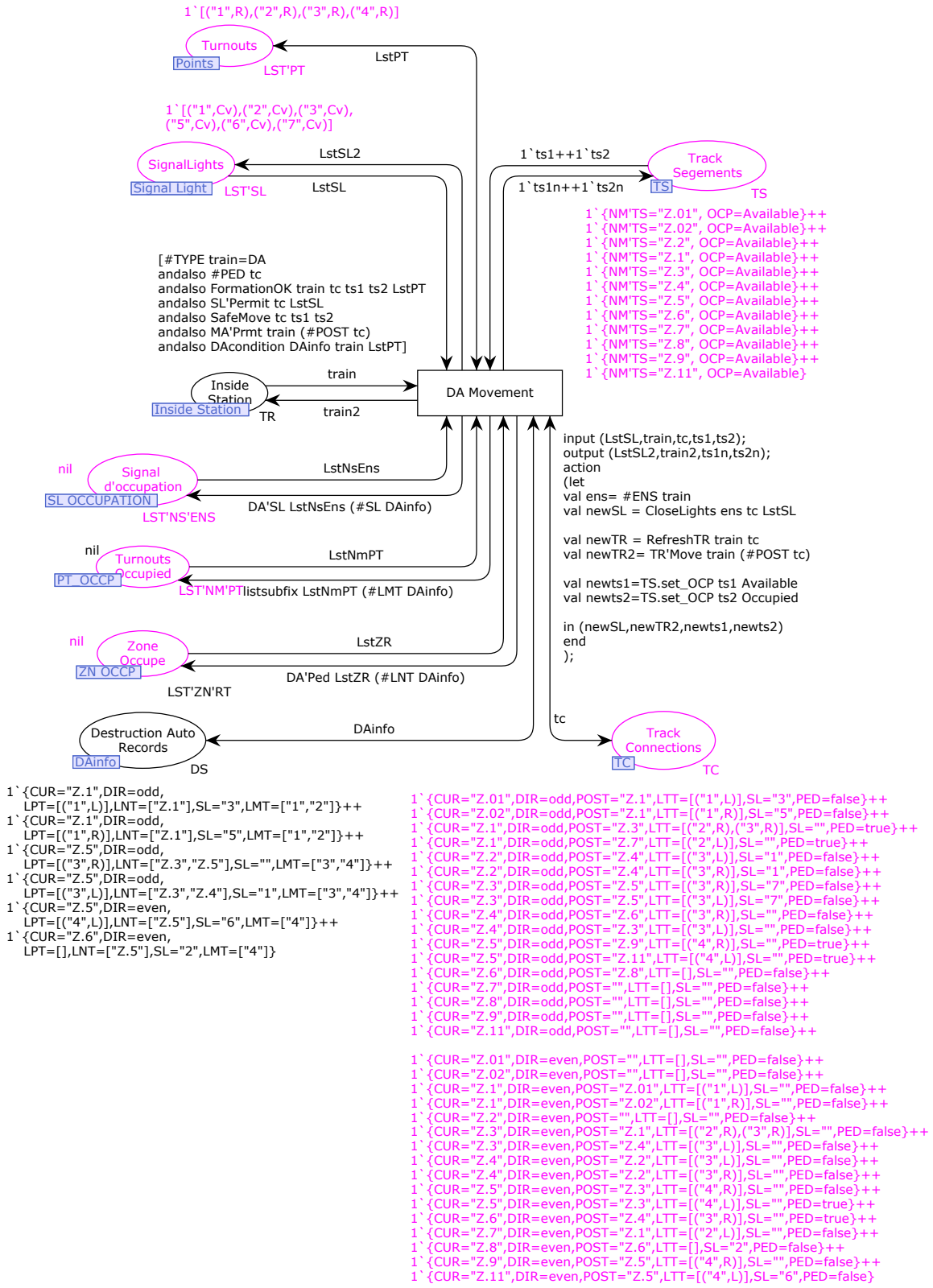


Figure 3.26 – Generalized Petri net model of "DA" route pattern

That means there in no state has two trains on the same track.

Table 3.5 – Result of route "3/15" simulation

Last Action	Train token	Signal lights	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.1, 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.3, 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.5, 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.11, 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)		

3.6 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 analyse the low-level parts of RIS. That is the modelling methodology of the relay-based systems.

3.6.1 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 maybe 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:

1. Route formation: receiving the route command from the dispatcher, and setting point to the right position by point machines.
2. Formation verification for interlocking: verifying the positions of the points relay. If all the relays are properly positioned, the formation will be interlocked.
3. Route verification: verifying the real point positrons, if they are well positioned, then

send a command to signal light control logic

4. Signal light control: switching on the lights and display different colours 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. 3.27. The example contains the main components for route establishment. It is realized by a set of relays and switches that are located in different layers (circuits diagram). However, as shown in Fig. 3.27, the dash-dot line connected elements, in nature, are the same element. They are physically connected together, changing their states at the same times, but located in different circuits. The established procedures of this example are explained as follows:

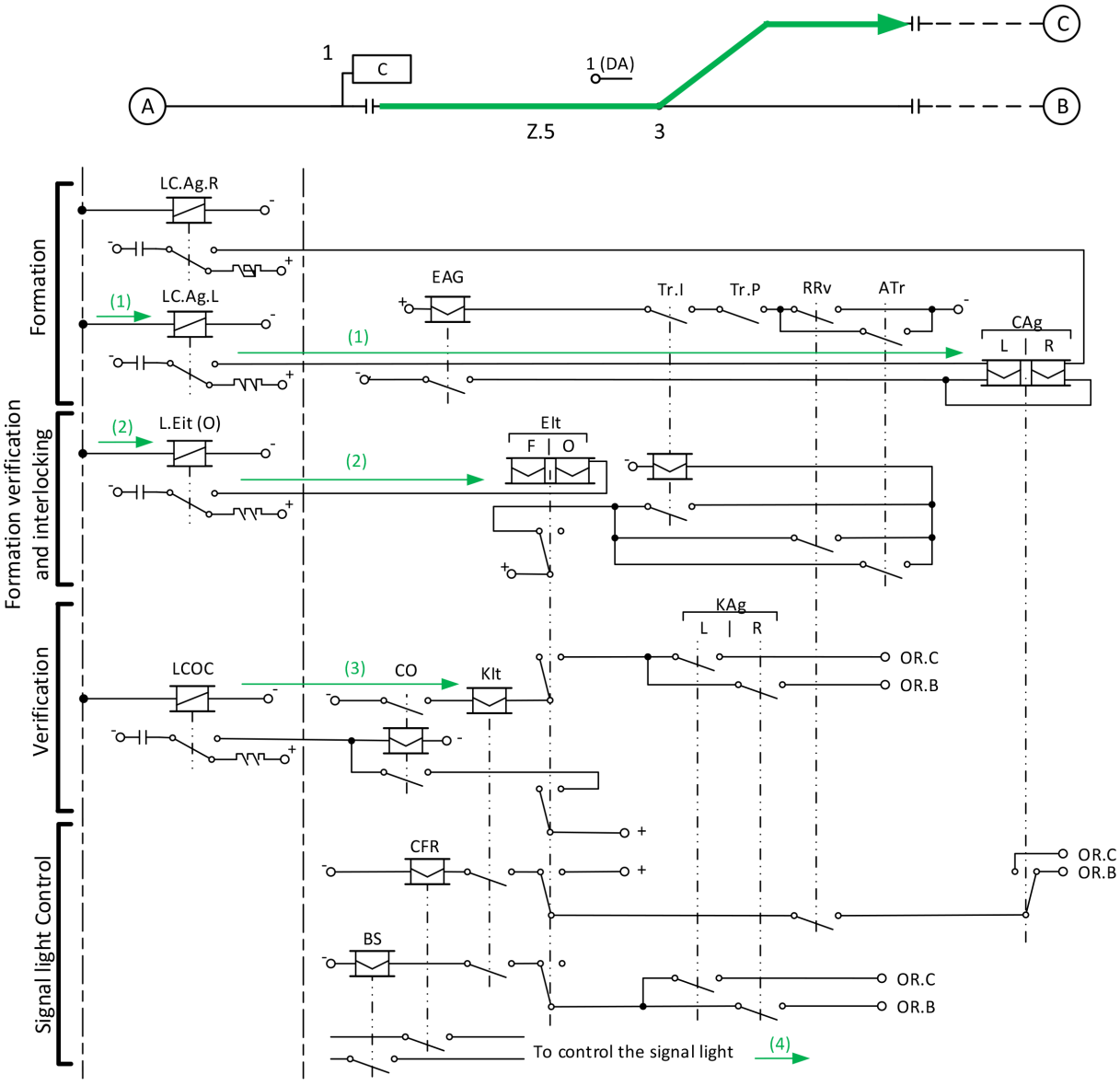


Figure 3.27 – An example of PRCI type system of a single point

1. 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.
2. 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*.
3. 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.
4. Switching on the signal light according to the relay *CFR* and *BS*.

From Fig. 3.27 and its procedures, we know that relays can be activated or deactivated in different layers by commands from the signalling centre, 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 problem are caused by the HCPN models that consist of several sub nets. 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 solution 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 problem.

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 dis-synchronization of firing transitions fails to refresh the system simultaneously, and it may lead the system to uncertain states, such as standstill, live-lock, dead-lock or even an unreasonable state. Such an example can be found in Fig. 3.28.

This example describes two logical processes which 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: $T1n$,$T1m \rightarrow T2m$,$T2n$, and the expected final state is S_{fini}=[3,3/ 1,0,1]. But if the transition $T2n$ fires before $T1m$, 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, which could organize all the marking enabled transitions to be fired in the right orders, as they do in the real system. Moreover, considering the compatibility, the proposed solution should be achieved under the framework of CPN.

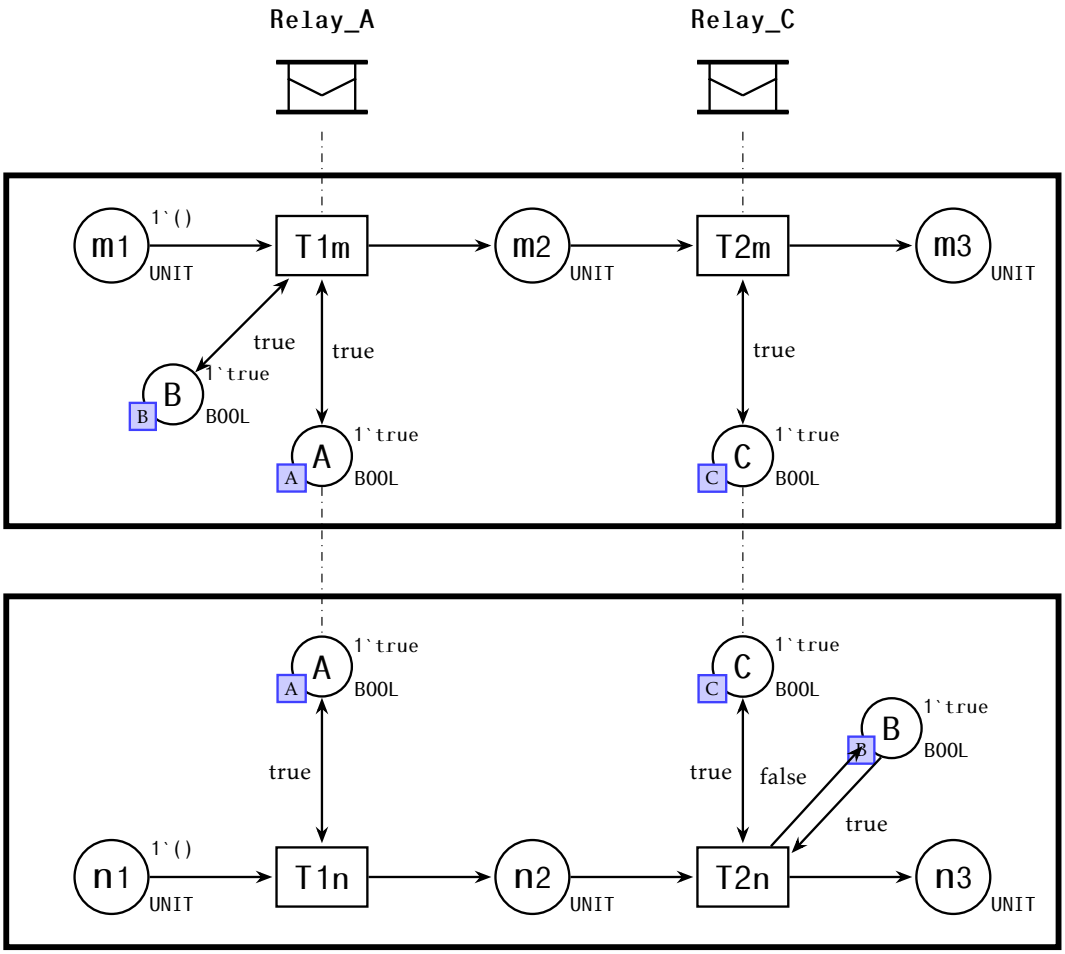
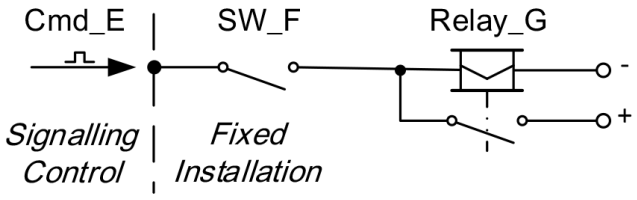


Figure 3.28 – Modelling problem I: synchronous firing

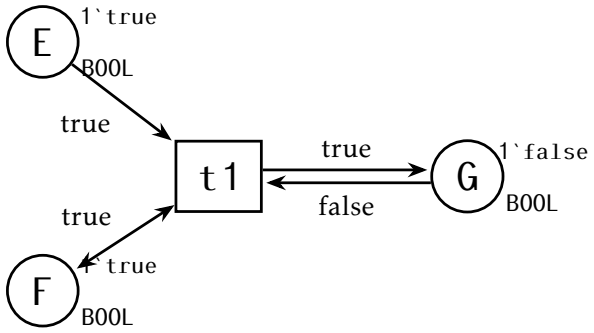
Modelling Problem II: Firing Conditions

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 elements meet the required conditions, the relay is activated. However, there is another "temporary" type of conditions. They are pulse signals which 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. 3.29(a)

The *Cmd_E* is a command from signalling 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. 3.29(b), 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 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.



(a) Example of different conditions



(b) Corresponding model

Figure 3.29 – Modelling problem II: firing conditions

3.6.2 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 which allow an external controller to influence the progression of tokens in the net [Holloway and Krogh 1994; Holloway, Krogh, and Giua 1997b]. Fig. 3.30 illustrates a controlled Petri net, where the squares (*c1*,*c2*,*c3*) indicate the external control place.

As with 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 place 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, which stores all the events in progress in their order of occurrence. This FIFO list has the following properties:

1. Only the head (first element) of the list is referred as the current *activated* event and tt will activate its corresponding transitions.

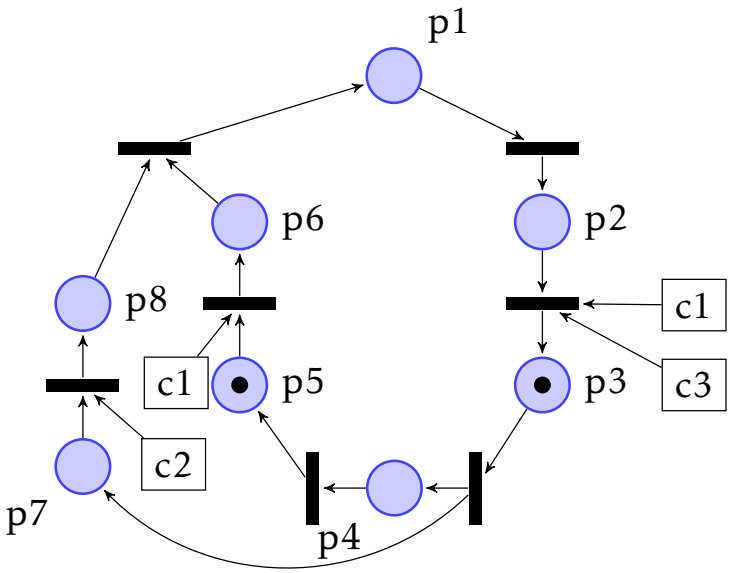


Figure 3.30 – An example of controlled Petri net

2. 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.
3. New events which are induced by internal actions are stored at the end of the list.
4. Only when the system has no more events in this list, can this system 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 becomes "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, which 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.

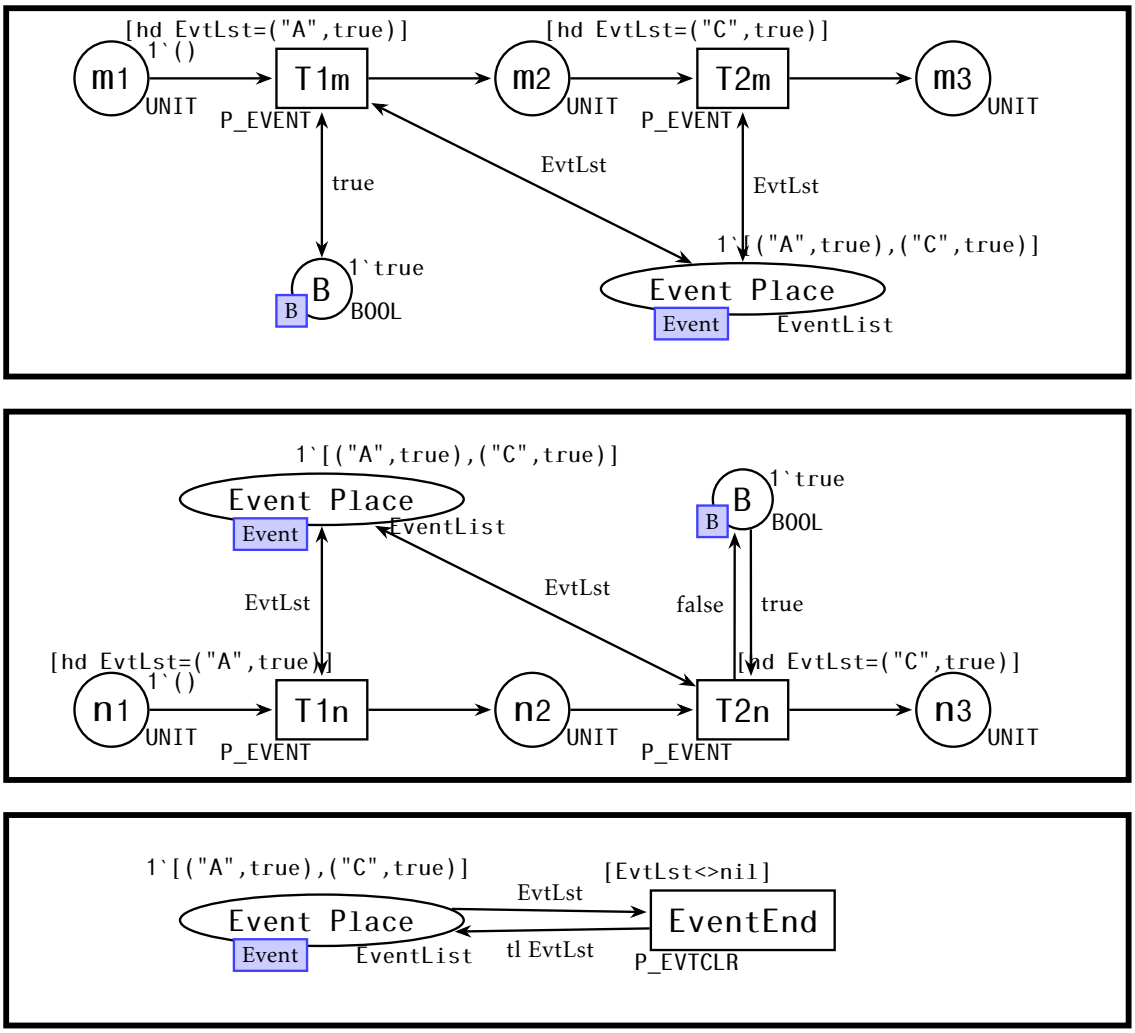
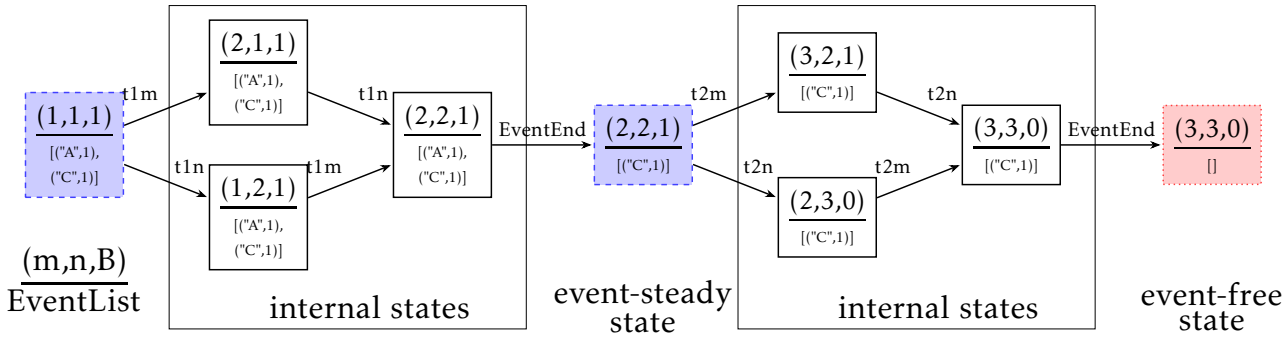


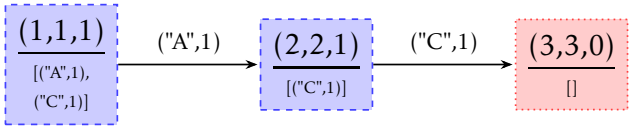
Figure 3.31 – Event-driven coloured Petri net model of Fig. 3.28

Now, we can rebuild the example in modelling problem I: "the synchronous firing" with the event-driven concept. In Fig. 3.31, the colour 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, ("B", false) means relay "B" is inactivated. All the transitions are connected to an "Event Place" which stores the events to be triggered in their order of occurrence. Its colour 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 which 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 become activated.



(a) Space state of Fig. 3.31



(b) Important states for analysis

Figure 3.32 – Simplification rules of system space state

The state space of this model is shown in Fig. 3.32. 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 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 $(m,n,B) = (1,1,1)$, $EventList = [(”A”,1)(”C”,1)]$. 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 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 2 successive system-steady states, there may be more than one path, and the number of combinations of the path depends on the number of the 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 analyse 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. 3.32(b). 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. 3.32(a) can evolve into a quite simple one in Fig. 3.32(b) 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. 3.33. 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 $fire$".

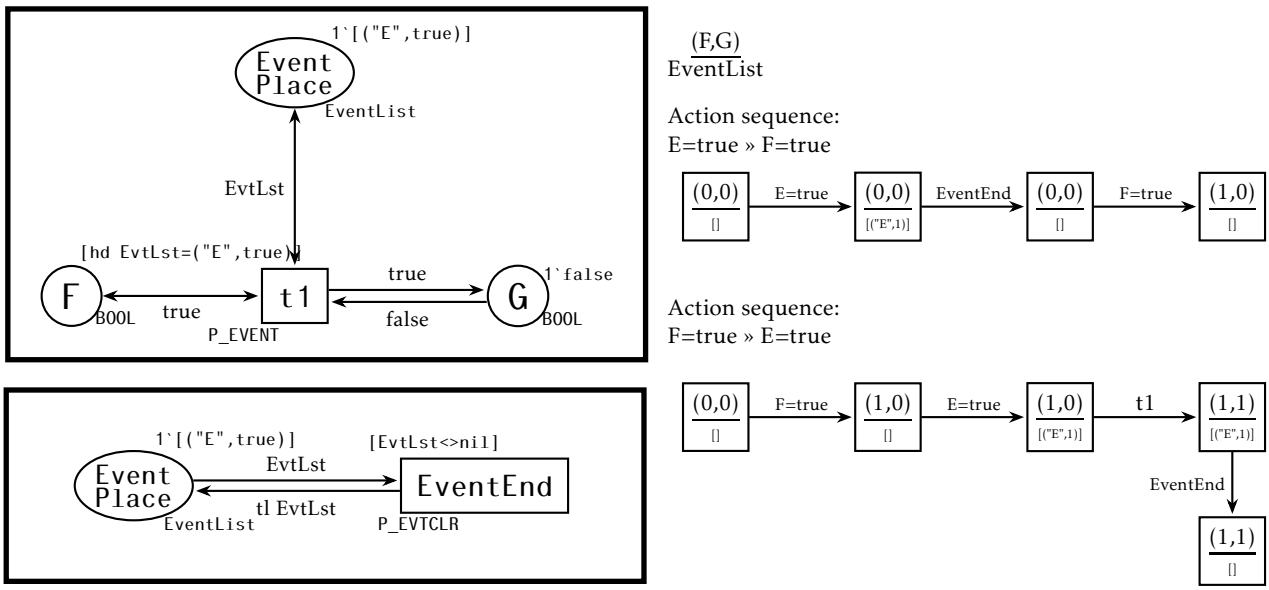


Figure 3.33 – Event-driven coloured Petri net model of Fig. 3.29

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 3.6.

Table 3.6 – 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 centre	X		
Action	command or data send back to controlling centre			X

3.6.3 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 which integrate a powerful state space tool. It could generate the full state space of the PNs mode and it could analyse the state space by means of a CTL-like temporal logic, which 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. Only when the original system reaches a new steady state, can the test layer give a new external input. 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.

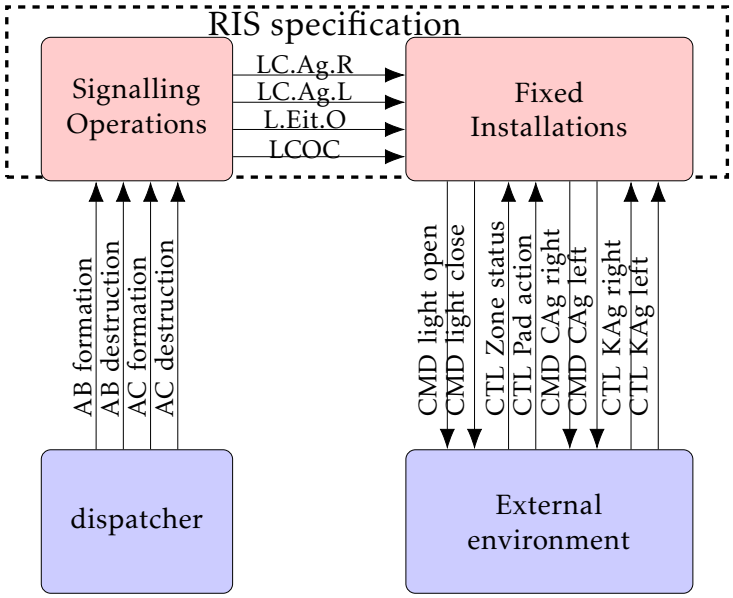
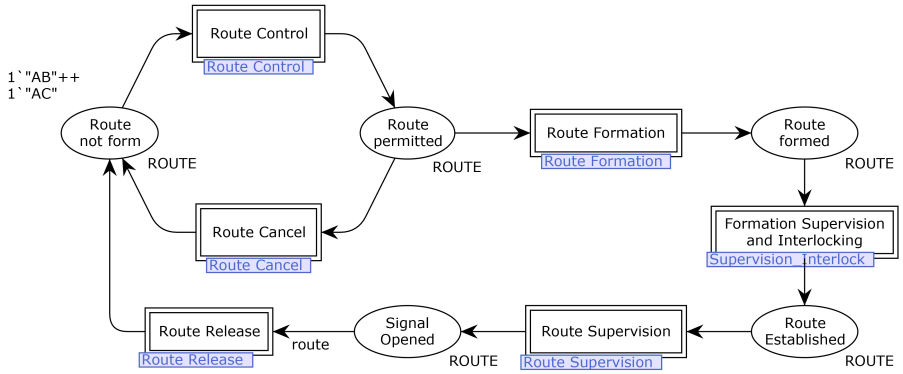


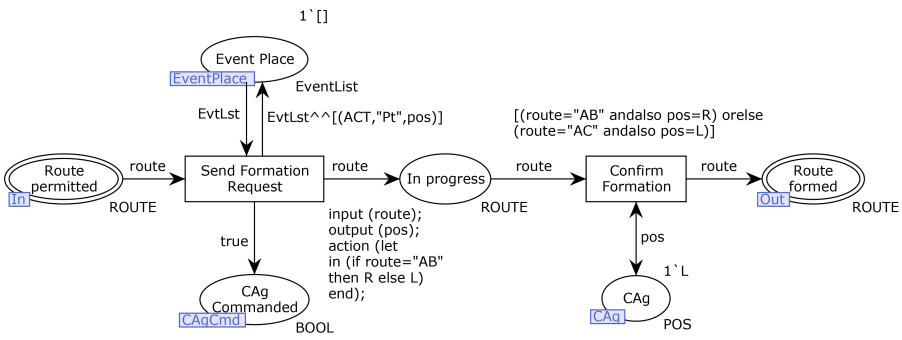
Figure 3.34 – Modelling structure and simulation environment

System modelling

To illustrate a complete practical use, a model of RIS in Fig. 3.27 will be demonstrated. This case study is very simple in that it only contains one point and two interlocking routes.



(a) coloured Petri net model of signalling operations layer



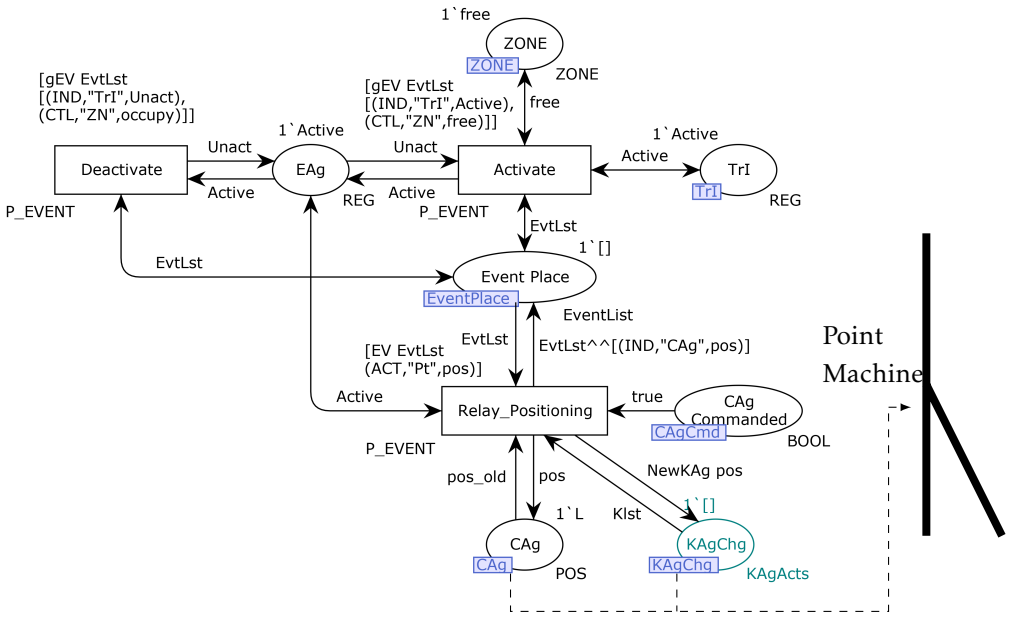
(b) coloured Petri net model of route formation

Figure 3.35 – coloured Petri net model of signalling operations

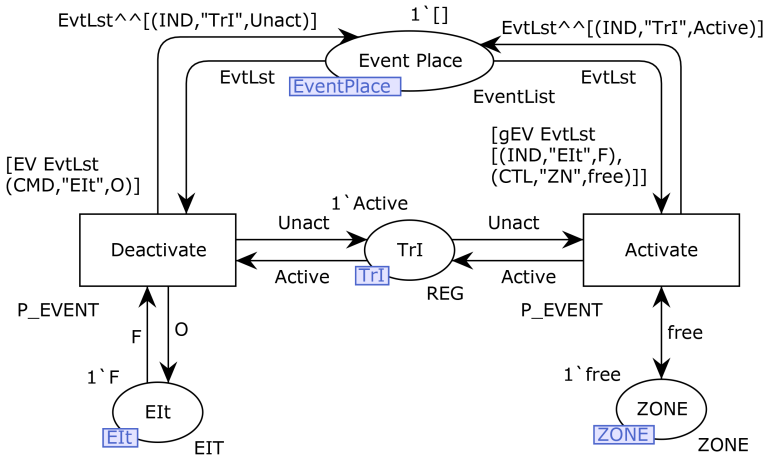
The signalling operations in this model are to send commands to establish or destroy an interlocking route. A reasonable modelling structure and its simulation environment is shown in Fig. 3.34.

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 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 signalling operations as we discussed in Section 3.4.1. There is a simplified version of it in Fig. 3.35(a), which contains different route phases (unformed, permitted, formed, etc) and corresponding transitions. Fig. 3.35(b) is also a simplified version of route formation. As the signalling operations have been discussed before,



(a) coloured Petri net of point layer



(b) coloured Petri net of transit layer

Figure 3.36 – coloured Petri net model of point control

considering the space restrictions, other sub-models of signalling 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 $fun\ EV : Event\ list * Event_x -> BOOL$. It is the guard function of event-related transitions, and will return *true* if the $Event_x$ is at the top of the *Event list*.

The point control (in Fig. 3.36) contains two parts: 1. The point layer which 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. 3.37), which 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.

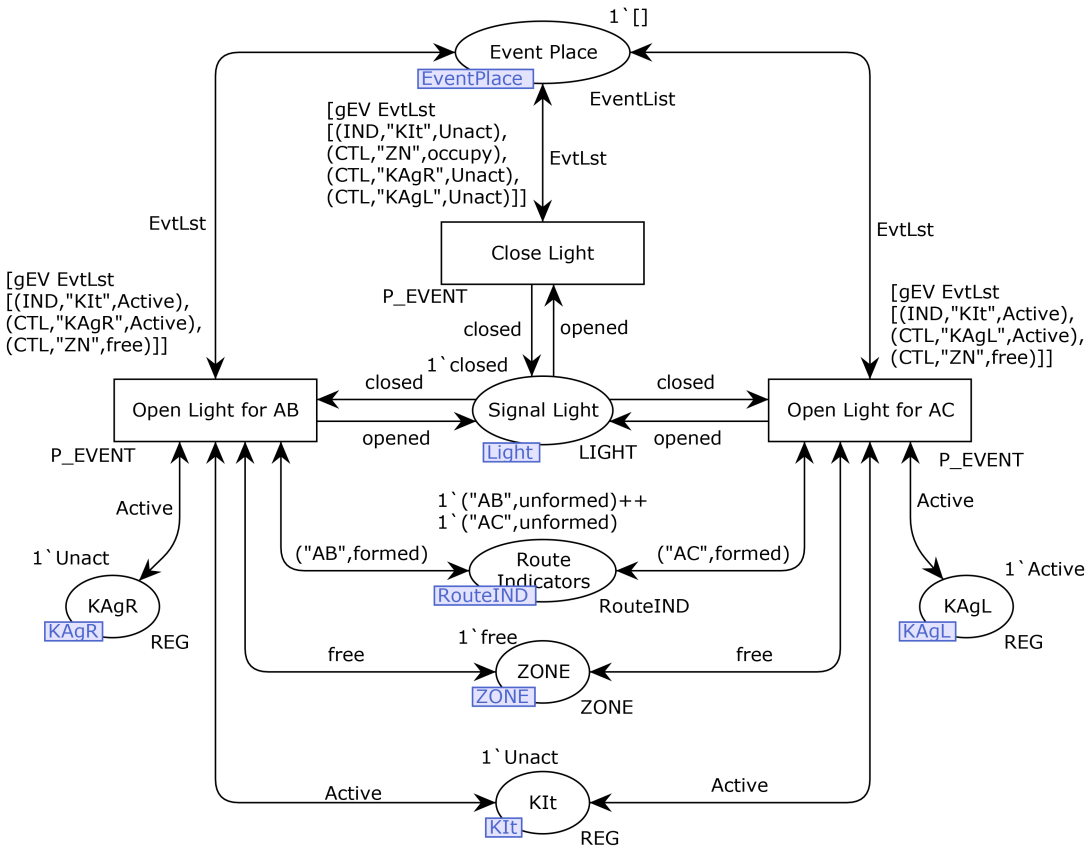


Figure 3.37 – coloured Petri net model of signal light control

For model checking purposes, we need to add a test layer to simulate all the external input events in Fig. 3.34, and allow those events to vary freely. In this mode, the considered external inputs are route command(formation/destruction), zone occupation, pedal action, 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. 3.38

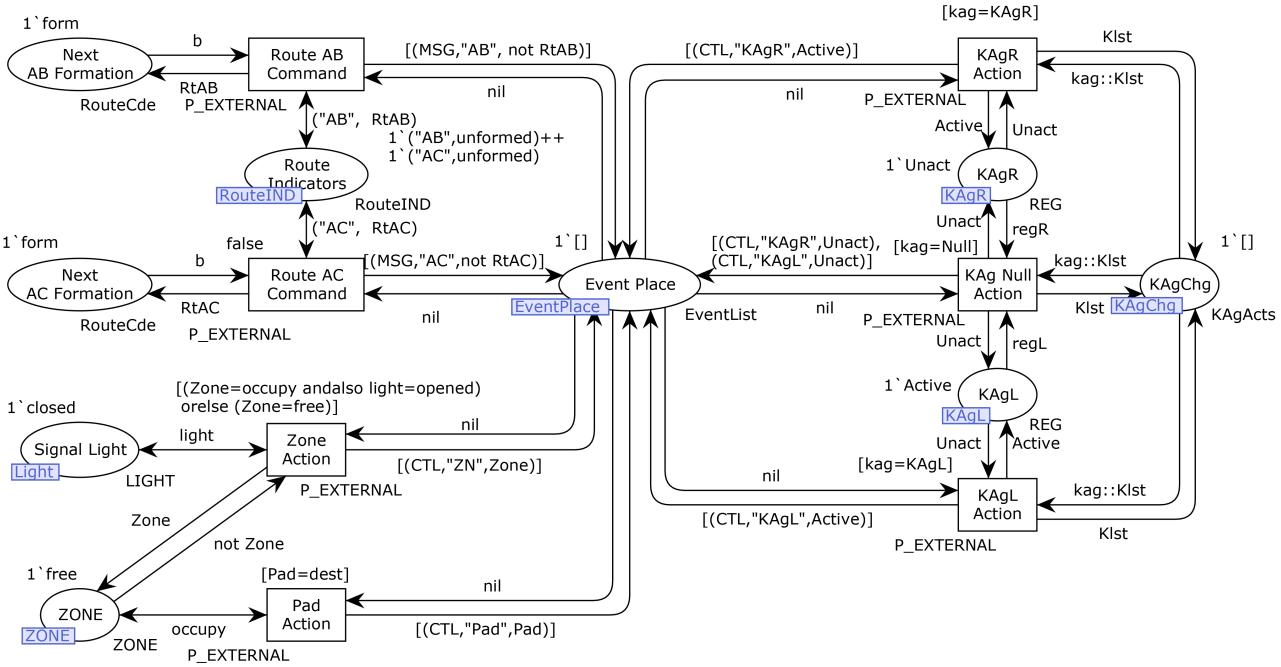


Figure 3.38 – coloured Petri net model of test layer

State space analysis

The safety statements of this system are:

1. φ_1 : If any route is formed or zone is occupied, the relay *CAg* that controls the point cannot change;
2. φ_2 : if no route is formed or zone is occupied, signal light cannot be switched on;
3. 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 $\varphi_1(S) \wedge \varphi_2(S) \wedge \varphi_3(S) -> BOOL$, 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 are shown in the second column of Table 3.7.

Table 3.7 – 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 (6 states)	not holds (6 states)

Although the size of the system state space has been simplified, it is still not readable. Moreover, too much information on CPN marking makes its difficult for 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 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 3.7). Part of the graph is shown in Fig. 3.39, where the node in grey dashed style is the state already visited(duplicates), and red node is the danger state.

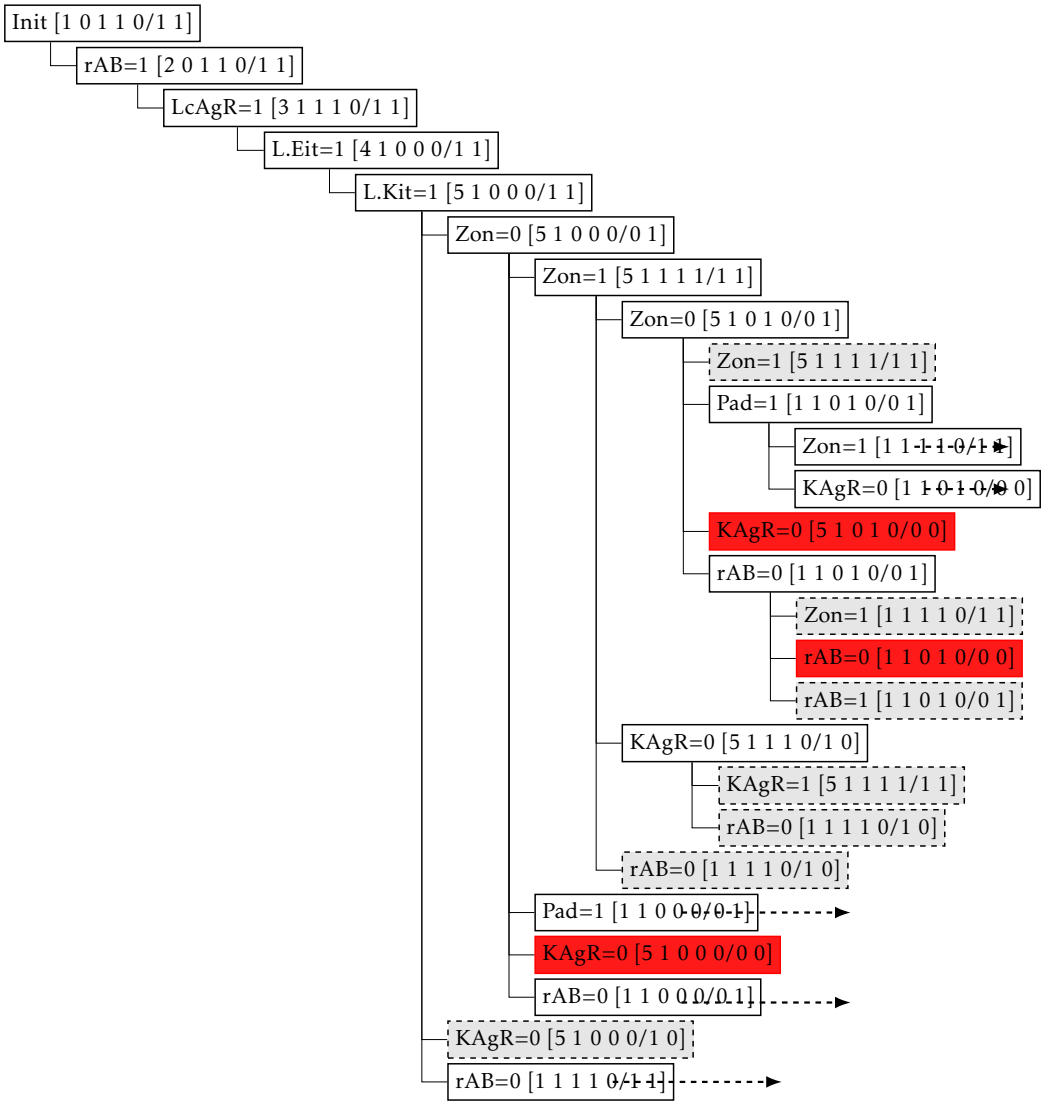


Figure 3.39 – Part of the state space tree

The counter-examples of the verification are generated by giving the paths from initial

state to each danger state. There are 6 paths in this example:

1. 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
2. ⋯ → L.Kit=1 → Zon=0 → rAB=0 → KAgR=0
3. ⋯ → L.Kit=1 → Zon=0 → Pad=1 → KAgR=0
4. ⋯ → L.Kit=1 → Zon=0 → Zon=1 → Zon=0 → KAgR=0
5. ⋯ → L.Kit=1 → Zon=0 → Zon=1 → Zon=0 → rAB=0 → KAgR=0
6. ⋯ → L.Kit=1 → Zon=0 → Zon=1 → Zon=0 → Pad=1 → KAgR=0

All of the counter-examples 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 programmes, 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 analysed 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, while after state abstraction, it has 65 states where 25 are duplicates, no danger state and no counterexample.

3.7 Conclusions

The goal of this chapter is to present a formal specification methodology for RIS. The work reported in this paper exploit both the modelling power of CPN and its rigorous semantics. Its hierarchical and colour features make it possible to propose a generic and compact structure which contains all high-level functions of RIS. This chapter first introduces a geographical approach of RIS, which consists of two parts: a hierarchical model of signalling operations and a geographical model of fixed installations. After that, in order to have a reusable and parametrized structure of fixed installations, we propose a general modelling pattern for French RIS. Models that are less compact can also be derived from this generic

one in order to validate various aspects, while keeping the safety property. For low-level systems (relay-based logic), we introduce an event-based concept, which could make the relay-based system clearer and easier to be constructed. It maintain the system's synchronization during a signal event and it is compatible with classic CPN models. Furthermore, a simplification method fo the state space analysis is proposed, making the system state space more compact, easier to analyse and read, without losing reliability.

Actually, another strong contribution of the paper lies in the fact that there are few approaches for French railway infrastructure modelling in the state of the art. As a result, the presented works are original contributions. This method could also be applied to other similar railway interlocking systems.

Chapter 4

Model transformation: from coloured Petri net to *B* language

This chapter introduce the modelling transformation approach, which is from the CPNs model to abstract *B* machines.

4.1 Introduction

In model driven engineering, everything is a model. Thus, the model transformations are considered as the "heart and soul" [Lúcio et al. 2014] of model driven engineering, and can be used for a variety of purposes, such as the generation of models on different levels of abstraction, the creation of different views on a system and automation of model evolution [Czarnecki and Helsen 2006]. The aim of this chapter is to describe a method of CPN model transformation. This method could help people to quickly shift from a valid design solution to a valid input of *B* development process in the design phase.

On the one hand, a fine behavioural specification has to be able to be assessed by an expert. On the other hand, the implementation result should respect some common industrial constraints. In the French railway context, the PNs and the *B* are two industry recognized tools. The PN models are understood and widely used by expert engineers , because of their powerful properties, especially the graphical "place-transition" notations. Therefore, many practical systems and valid solutions are specified with PNs and other high-level PNs. However, 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]. Let us remark that the assessment of high-level design is a major safe element on its own. In this case, you may not need to execute the complete *B* refinement processes. Consequently, in this chapter, we do not really focus on the *B* refinement.

Actually, in most industrial practices, the accurate requirements are difficult to express and to assess using formal methods. So the guidance and correction from human expertise cannot be avoided [Defossez, Collart-Dutilleul, and Bon 2011]. Although the *B* method is well suited for formal assessment in the development of computer software, its notation form demands a lot of pre-training for set theory and first order logic, which greatly increases the research and development costs. So it has only been implemented in the major safety-critical applications, such as a city metro line's signalling system [Erbin and Soulas 2003]. In contrast, the fine graphical representation of PNs is really an advantage for industrial experts who may not be familiar with mathematical formalism. So there are already many valid solutions modelled by PNs. Moreover, the modelling power of CPNs, a sort of high-level PNs, provides a good formal framework for large scale, complex systems but presents a concise model, which enhances its scope of application [Sun, Collart-Dutilleul, and Bon 2014].

This technique mismatch leads to the following question. Let us assume in the design phase, a model has been validated by industry experts, using dedicated industrial tools or their expertise. The problem is how to prove it formally with respect to the specifications and automatically generate the implementable codes. To be more precise, the problem to be solved is the translation from the CPN formalism into the *B* language formalism. Some similar work of this approach can be found in [Attiogbe 2009; Korečko, HudáK, and ŠIMOŇÁK 2007; Korečko and Sobota 2014], which are based on low-level PNs. Our research is an improvement of the work from [Bon and Dutilleul 2013], which translates a CPN model with only numerical colour sets into *B* machines regarding Atelier B syntax.

This chapter is organized as follows:

First we briefly recall the basics and the outstanding features of the *B* theory, including some notations, concepts and the mathematics syntax of *B* language.

After, we introduce an explicit and detailed transformation framework from non-hierarchical CPNs to AMNs in which we improve the multi-set mechanism and its related specifications in [Bon and Dutilleul 2013], so that the transformed *B* machines can be automatically proved by Atelier B without using "interactive prover".

Then, we introduce the hierarchical concept of HCPN into the previous transformation, and present a new transformation from a HCPN to abstract machine notations.

Finally, we consider the prioritized transition of CPN, and achieve the transformation by giving each operation a priority and adding a priority management operation.

4.2 Preliminaries of the *B* method

The *B* method is a state-based model-ordered method, developed by Prof. Jean-Raymond Abrial [Abrial 1996], designing and coding software systems. This language is founded on

the Zermelo-Fraenkel set theory with the axiom of choice [Cansell and Méry 2012]. It provides a specification formalism called abstract machine. Sets are used for the static part (data or state) of models. Generalised substitutions are used to express the dynamic part (data modification or state changes). Refinements are used for evolving the *B* machine at various levels of abstractions. The developing process is "machines → refinement → implementation". Fig. 4.1 shows the schematic diagram of the software developing process with *B* method. This language is supported by Atelier B tool [ClearSy 2011, 2014] and the B [Lano and Haughton 1996; Robinson 1997].

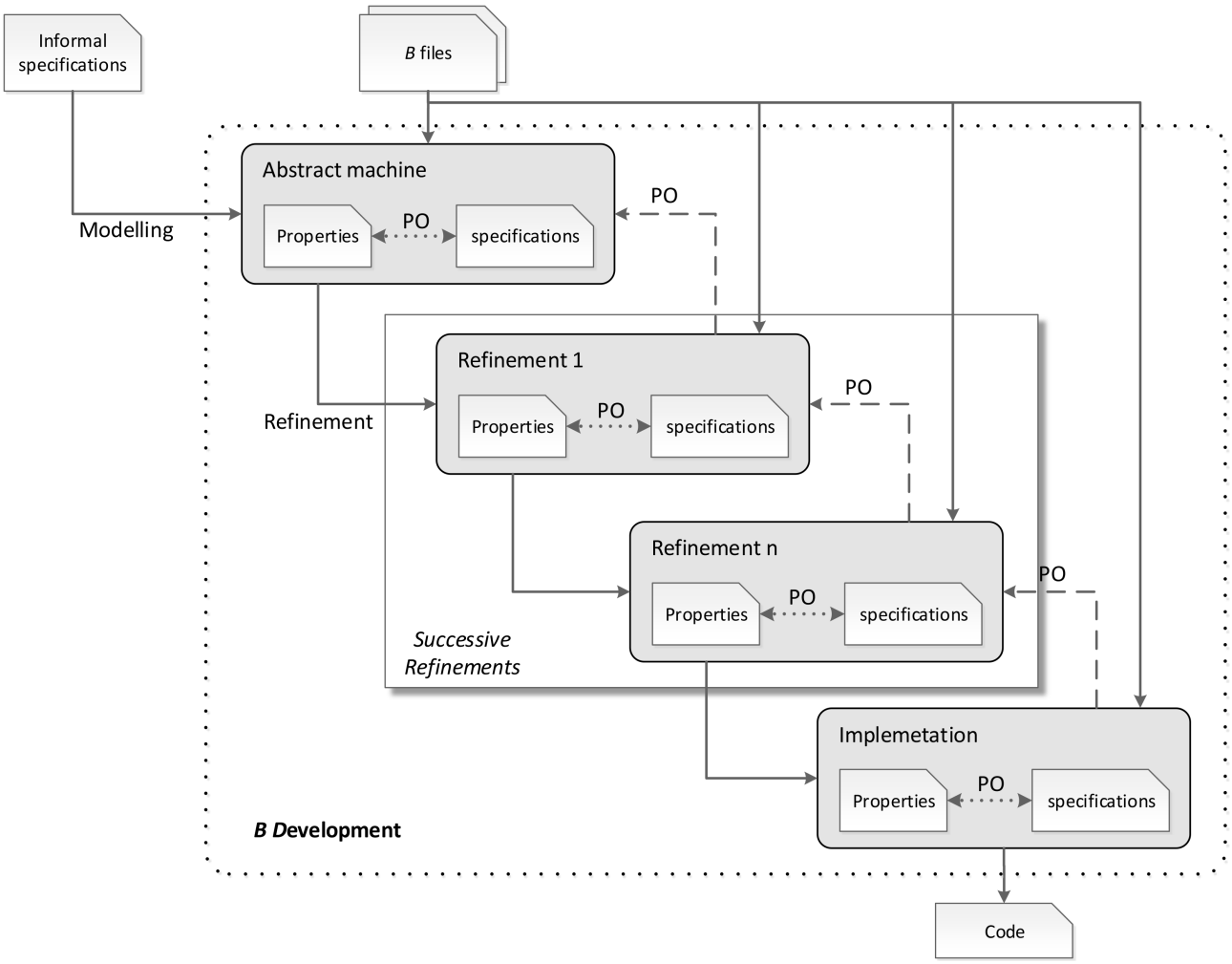


Figure 4.1 – Software development with method *B*

The *B* development tool used in this these is Atelier B tool, which is a robust, commercially available tool developed by ClearSy company. It covers all stages of software development, including the specifications (abstract *B* machine), the refinements, proof tool and code generation. All formal proofs are automatically generated and proved by the proof tool. Atelier B also provides an interactive proof environment used in the case where the automatic proof engine fails.

Proof-based development

"*Proof-based development methods integrate formal proof techniques in the development of software systems*" [Cansell and Méry 2012]. The main process is started with an abstract machine (the *B* model) of the system. Details of the system are gradually introduced into the original machine by generating a series of more concrete machines. The relationship of two successive machines is the called the "refinement". The refinement process can be repeated many times, each refinement representing the intermediary stop between specification and the final implementation. The important feature of the refinement process is that it can preserve those already proved system properties including safety properties and termination.

During each refinement process, the correctness and effectiveness of each step are validated by **proof obligations** (POs) which are responsible for the consistency of the development process. This whole process allows *B* machine to develop into safe code. Such POs could be generated by the proof tool using automatic or interactive proof procedures supported by a proof engine [ClearSy 2014].

Before the generation of the POs, system properties in the *B* model are described by means of the "invariant" predicate. Invariants and proof obligations together keep the consistency of the model. Moreover, invariants are required to ensure that data properties are preserved by the operations of the machine. In each refinement step, the machine is added with a further invariant which relates the more concrete data form. These invariants are also used to generate the new proof obligations for refined machines.

4.2.1 The *B* language

In this subsection, we only present the mathematical concepts and some notations relating to the *B* method, which are necessary for understanding the following chapters.

Sets and predicates

The *B* development of a machine is started with analysis of the mathematical structure: sets, constants and properties over sets and constants.

Constants can be defined using first order logic and set-theoretical nations of *B* A set can be defined as $\{x \mid s \wedge P(x)\}$ or a Cartesian product $s \times t$ or a set of sets $\mathbb{P}(s)$, intersection of set using $\cup$ and $\cap$. A pair is denoted as (x, y) or $x \mapsto y$. A relation over two sets s and t is an elements of $\mathbb{P}(s \times t)$. A relation r has a domain $dom(r)$ and a range $ran(r)$. A function f from sets s and t is also a relation that mapping from each element in $dom(f)$ to at least one element in t. A function si either partial $f \in A \nrightarrow B$, or total one $f \in A \rightarrow B$

The formal syntax of *B* formulas is summed up as follows, where V is the set of variables, I is the set of identifiers, E is the set of expressions, S is sets, P is the predicates, BIG is an infinite set.

$V := I \mid V \mapsto V$

$E := V \mid [V := E]E \mid E \mapsto E \mid S$

$S := S \times S \mid \mathbb{P}(S) \mid V \mid P \mid BIG \mid I$

$P := P \wedge P \mid P \Rightarrow P \mid \neg P \mid \forall V.P \mid [V := E]P \mid E = E \mid E \in S \mid I$

In Table 4.1, some set-theoretical notations are listed in both symbolic and mathematical ways. Those notations are used to effectively describe the system.

Table 4.1 – Set-theoretical notations of *B* language

Name	Syntax	ASCII	Explanations
Binary relation	$s \leftrightarrow t$	<->	$P(s \times t)$
Composition of relations	$r_1; r_2$	;	$\{x, y \mid x \in a \wedge y \in b \wedge \exists z.(z \in c \wedge x, z \in r_1 \wedge z, y \in r_2)\}$
Inverse relation	r^{-1}	~	$\{x, y \mid x \in P(a) \wedge y \in P(b) \wedge y, x \in r\}$
Domain	$dom(r)$	dom(r)	$\{a \mid a \in s \wedge \exists b.(b \in t \wedge a \mapsto b \in r)\}$
Range	$ran(r)$	ran(r)	$dom(r^{-1})$
Identity	$id(s)$	id(s)	$\{x, y \mid x \in s \wedge y \in s \wedge x = y\}$
Restriction to the domain	$s \lhd r$	<|	$id(s); r$
Restriction to the range	$r \rhd s$	|>	$r; id(s)$
Subtraction to the domain	$s \lhd\!\!\!- r$	<<|	$(dom(r) - s) \lhd r$
Subtraction to the range	$r \rhd\!\!\!- s$	|>>	$r \rhd (ran(r) - s)$
Image	$r[w]$	r[w]	$ran(w \lhd r)$
Override	$q \lhd\!\!+ r$	<+	$(dom(r) \lhd\!\!\!- q) \cup r$
Partial Function	$s \nrightarrow t$	+->	$\{r \mid r \in s \leftrightarrow t \wedge (r^{-1}; r) \subseteq id(t)\}$

Logical structure of *B* machine

The notation of the specifications in B method is known as the abstract machine notation (AMN), which could describe different levels of abstractions. Fig. 4.2 presents a *B* method notation and its refinement. Normally, an abstract machine could contain different clauses to describe its properties and operations. These clauses can be classified into three categories: composition, declarative and executive [Boulanger 2014]. In Fig. 4.2, each category is marked with a character style, the compositions are normal style, the declaratives are *italics* style, and the executives are **bold** style.

Compositions The compositions are used to define the relationships between different abstract machines which belong to the same system. The possible relations are INCLUDES, SEES, IMPORTS, EXTENDS or USES. Each clause indicates a different visibility and access permission on the components of the other abstract machines.

The INCLUDES clause is to bring together the components of machines instances (sets, constants and variables) as well as their properties (PROPERTIES and INVARIANT clauses), in order to make up a complex abstract machine using other abstract machine. The including component cannot modify included variables, but can access them by “read”. However,

MACHINE M(p) SEE N *CONSTRAINTS* *C* *SETS* *St* *CONSTANTS* *k* *PROPERTIES* *Bh* *VARIABLES* *v* *INVARIANT* *I* ***DEFINITIONS*** ***D*** **INITIALISATION** **T** **OPERATIONS** **y ← op(x) =** **PRE P THEN S END** **...** END	REFINEMENT R(p) REFINES M *CONSTRAINTS* *C1* *SETS* *St1* *CONSTANTS* *k1* *PROPERTIES* *Bh1* *VARIABLES* *w* *INVARIANT* *J* ***DEFINITIONS*** ***D1*** **INITIALISATION** **T1** **OPERATIONS** **y ← op(x) =** **PRE P THEN S END** **...** END

Figure 4.2 – General structure of *B* machine (left) and refinement (right)

the included component allows the including component to modify included variables by included operations. The included invariant is preserved by the including component and is used by the proof tools for generating proof obligations of the including component.

The SEES clause is used to reference other machine instances, which means it can access the constituents (sets, constants and variables) of other machines without modifying them.

The IMPORTS links an implementation and an abstract machine instance. The implementation creates the imported abstract machine instance to use its data and operations to implement its own data and operations.

In an abstract machine or a refinement, the EXTENDS clause is equivalent to an inclusion of the machine instances and the promotion of all of the operations of the included machine instances.

The USES clause is used for the included machines, to share the data of one of the included machines. Using machines can refer to share variables in their invariants, and data of the used machine are shared among using machines. When a machine uses another machine, the current project must contain a machine which includes the used and using machines.

Declarative The declarative clauses describe all the statements of data (sets, variables, constants) and their invariant conditions, which will always be verified in order to keep the validity of the machine. All the declarations depend on the set theory and first order predicates. The *B* language is built on manipulation of sets, rather than on the manipulation of types. For each variable, it will calculate all the associated values. Table 4.2 introduces the basis sets.

Table 4.2 – The basic sets

Expression	Name	ASCII	Explanation
$\mathbb{B}$	Boolean	BOOL	$\{true, false\}$
$\mathbb{N}$	Natural number	NATURAL	$\{0, 1, \cdots\}$
NAT	Finite set of natural numbers	NAT	$\{0, 1, \cdots, MAXINT\}$
$\mathbb{Z}$	Integer	INTEGER	$\{\cdots, -2, -1, 0, 1, 2, \cdots\}$
INT	Finite set of integer	INT	$\{-MAXINT, \cdots, -2, -1, 0, 1, 2, \cdots, MAXINT\}$

Those basic sets are associated with some operators which make it possible to construct more complex sets. Table 4.3 introduces the major operators for the creation of *B* sets.

Table 4.3 – Operators of complex sets

Expression	Name	ASCII	Explanation
$\emptyset$	The empty set	{}	The empty set
$\{e_1, e_2, e_3\}$	A list	{e1,e2,e3}	A set which contains only the elements e_1,e_2 and e_3
$P * Q$	Cartesian product	P*Q	The Cartesian product of P by Q
$S \leftrightarrow T$	Set of all the relationship of S*T	S<->T	Set of all the relationships from S to T, equal to P(S*T)

In the *B* machines, all the mathematical notations are written in the form of ASCII notation. Table 4.4 shows a comparison of ASCII notations and the corresponding mathematical notation.

Executives The executive part presents the dynamic actions of an abstract machine which includes the initialization and the operations. They make use of the generalized substitution language (GSL). A subset of GSL is shown in Table 4.5.

The ***DEFINITIONS*** clause contains explicit declarations or definition files of definitions of a machine. A definition may have parameters and could be used in other clauses for predicates, expressions or substitutions. In Atelier B, each use of a definition is directly replaced by the corresponding text, where formal parameters take the place of real parameters. The

Table 4.4 – Comparison of mathematical and ASCII notations

Mathematical notation	ASCII notation	Explanation
$\subseteq$	<:	included or equal
$\cup$	\/	Union
$\cap$	/\	Intersection
$\in$	:	Belong to
$:\in$	: :	Becomes an element of
$\forall x.(\cdots)$	!x.(...)	Universal quantifier
$\exists x.(\cdots)$	#x.(...)	Existential quantifier
$\wedge$	&	logical "AND"
$\vee$	or	logical "OR"
$\neg$	not	logical "NOT"
$\lambda x.(\cdots)$	%x.(...)	Lambda function

Table 4.5 – A subset of generalized substitutions

Substitution	**Meaning of GS**
x:=E	$[x := E]R \Leftrightarrow$ substitution of all the free occurrences of x in R by E
skip	$[skip]R \Leftrightarrow R$
S || T	$[S\|\|T]R \Leftrightarrow [S]R_s \wedge [T]R_t$
S;T	$[S;T]R \Leftrightarrow [S]R_s,[T]R_t$
CHOICE S_1 OR S_2 END	Do S_1 or S_2
PRE E THEN S_1 END	If predicate E holds, do S_1, otherwise do anything
SELECT E THEN S_1 END	If E holds, do S_1, otherwise do not execute
IF E THEN S_1 ELSE S_2 END	If E holds, do S_1, otherwise do S_2
VAR v IN S_1 END	For any values of local variables from the list v do S_1
ANY v WHERE E THEN S_1 END	For any values of variables from v that satisfy E do S_1, If no values satisfy E, do not execute

scope of a definition located in a component is the whole of the component, including the text situated before the declaration of the definition.

Refinements

Abstract *B* machines are expressed in non-deterministic formalism, and they follow the rules of set language and first order logic. So some of the algorithms such as loops and

recursion are forbidden in the abstract machines. In engineering applications, these algorithms cannot be avoided, so progressive refinements are performed. These refinements are made in order to put in concrete form the manipulated structures and solve indeterminism in order to obtain a software written in a common programming language. Specifically, to refine means to modify data and operations. By refining, the non-deterministic parts of an abstract machine are replaced by some more programming-like substitutions, such as WHILE loops. The interfaces (such as parameters and operation headers) of a refining and a refined component have to be the same. For example, in Fig. 4.2 invariant J of the refinement R defines properties of w but also a relation between v and w. Whether J is maintained by all the operations of R is also verified by proving the POs.

Proof obligations The conditions of verification are the proof obligations They are generated from the declaration part of the model using mathematical theory. They are generated from the executive part of the model for the preservation (when re-calling the operations) of the invariant. Proof obligations state the correctness of safety properties with respect to the invariant. During the refinement process, they make clear the relationship between abstract model and concrete model.

More precisely, a number of proof obligations ensure that [Cansell and Méry 2012]:

- Each abstract event is correctly refined by its corresponding concrete version
- Each new event refines *skip*
- No new event takes control for ever
- Relative deadlock-freeness is preserved

4.3 From non-hierarchical coloured Petri net to *B* machine

With the extensive applications of formal methods, sometimes the integration of formal languages is unavoidable. The translation from PNs to *B* machines seems to have been a research interest in recent years. Published works give efforts to this approach from different perspectives. The work in [Korečko, HudáK, and ŠIMOŇÁK 2007; Korečko and Sobota 2014] presents a mapping method of low-level PNs (the PNs with undistinguishable tokens) to abstract *B* machines and to Event *B* machines. A similar work [Attiogbe 2009] presents an encoding of PT nets to the Event-B language. Our research is based on the work of [Bon 2000; Bon and Dutilleul 2013], which successfully translates a non-hierarchical CPN with numerical colour sets to *B* machines with Atelier B syntax. Nevertheless, the practical experience shows this work generates a lot of unproven POs using "automatic prover" in Atelier B.

Our study continues to use the basic data structures defined in [Bon and Dutilleul 2013].

Based on this, we introduce a more explicit framework of the translation, simplify the operations for multi-set structures, make the translation compatible with non-numeric types of colour sets. Then, we reform the mechanism of multi-set specifications, in order to ensure the target *B* machines can be automatically proved by Atelier B without using "interactive prover".

The translation introduced in this section is for non-hierarchical CPNs. It will consist of three parts. First, the framework (or template) of the translation is presented, for mapping the PN's "Place-transition" structure to the *B* machine formalism. Second, the colour property of CPNs is achieved by introducing the multi-set mechanism, which defines the token representation and its mathematical operators. It allows binding tokens with different colour types to each places, and modifying the content of the tokens. Then, different colour sets in CPNs are mapped into AMNs and user-define declarations are discussed. Finally, two non-hierarchical CPN models are introduced to demonstrate the translation process.

4.3.1 Framework of non-hierarchical translation

For a CPN $N = (P, T, A, \Sigma, V, C, G, E, I)$, we define that:

(i) $P = \{p_1, ..., p_m\}$,

(ii) $T = \{t_1, ..., t_n\}$,

(iii) $A = \{(p,t) \mid p \in P \wedge t \in T\} \cup \{(t,p) \mid t \in T \wedge p \in P\}$,

(iv) $\Sigma = \{\sigma_1, ..., \sigma_{k-1}, \sigma_k, ..., \sigma_l\}$,

(v) $C = \{c_1, ..., c_m\}$,

(vi) $G = \{G(t) \mid t \in T\}$ and

(vii) $I = \{I(p) \mid p \in P\}$.

Let β be the mapping $\beta : CPN \rightarrow AMN$, then the image of CPN N under β is a *B* machine Bm, $Bm = \beta[N]$ with the following structure.

For all the elements mentioned in Fig. 4.3, they have the following mapping relationship:

1. "MultiSets.def" is a external definition file, which will be introduced in next subsection.

For each colour set $\forall k \in \{1..l\}$, the mapping relations are shown below.

2. $Colset_k = \beta[\sigma_k]$ is the colour set , where $\sigma_k = C(p_i) \in \Sigma$.

3. $(Colset_k)_{\mathrm{MS}} = \beta[C(p_i)_{\mathrm{MS}}]$, is the multi-set of colour .

For each place related elements $\forall i \in \{1..m\}$, the relations are shown as follows.

4. $VP_i = \beta[M(p_i)]$, is the marking of each place, where $p_i \in P$.

MACHINE M
SETS $Colset_1, Colset_2, \cdots, Colset_{k-1}$
CONSTANTS $Colset_k, Colset_{k+1}, \cdots, Colset_l$
PROPERTIES
$Colset_k := EXPR_V^k \ \& \ Colset_{k+1} := EXPR_V^{k+1} \ \& \cdots \& \ Colset_l := EXPR_V^l$
VARIABLES $VP_1, VP_2, \cdots, VP_m$
INVARIANT
$VP_1 : (Colset_1)_{\mathrm{MS}} \ \& \ VP_2 : (Colset_2)_{\mathrm{MS}} \ \& \cdots \& \ VP_m : (Colset_l)_{\mathrm{MS}}$
INITIALISATION
$VP_1 := VI_1 \parallel VP_2 := VI_2 \parallel \cdots \parallel VP_m := VI_m$
DEFINITIONS
"MultiSets.def";
…
VAR_$t_j == EXPR_V^{t_j}$; // For trainsition T_j
E_p_1_$t_j == EXPR_{\mathrm{VAR_}t_j}$;
E_p_2_$t_j == EXPR_{\mathrm{VAR_}t_j}$;
…
E_t_j_$p_1 == EXPR_{\mathrm{VAR_}t_j}$;
E_t_j_$p_2 == EXPR_{\mathrm{VAR_}t_j}$;
…
GRD_$t_j == EXPR_V$;
CDT_$t_j == EXPR_V$;
…
OPERATIONS
OP_1= SELECT CDT_1 THEN ACT_1 END;
OP_2= SELECT CDT_2 THEN ACT_2 END;
…
OP_n= SELECT CDT_n THEN ACT_n END;
END

Figure 4.3 – The framework of a *B* machine transformed from coloured Petri net

5. $VP_i \in (Colset_k)_{\mathrm{MS}}$, where $Type[VP_i] = Colset_k$
6. $VI_i = \beta[I(p_i)\langle\rangle]$ is the initial value , where $Type[VI_i] = Colset_k$.
7. System marking $\beta[M]$ is defined as $\beta[M] = \sum_{i=1}^{m} VP_i$
8. The initial marking $\beta[M_0]$ is defined as $\beta[M_0] = \sum_{i=1}^{m} VI_i$

For each transition related elements $\forall j \in \{1..n\}$, the relations are shown below.

9. $OP_j = \beta[t_j]$ is the transition, where $t_j \in T$.
10. $VAR_j = \beta[Var(t_j)]$ is the variables of transition t_j.
11. $GRD_j = \beta[G(t_j)]$ is the guard function.
12. CDT_j is the condition of each operation, $CDT_j = (VAR_j \in Colset_k) \wedge cdt(p_1, t_j) \wedge cdt(p_2, t_j) \wedge \cdots \wedge cdt(p_m, t_j) \wedge GRD_j$.
13. $ACT_j = act(p_1, t_j) \wedge act(p_2, t_j) \wedge ... \wedge act(p_m, t_j)$.

The detail definitions are that for $\forall p, t$ $(p \in P, t \in T)$ there exists:

14. E_a is the arc expression, where $E_{p,t} = \beta[E(p,t)]$ and $E_{t,p} = \beta[E(t,p)]$.
15. $E_a = \begin{cases} EXPR_t \in (Colset_k)_{\mathrm{MS}} & if\ a \in A \\ \Phi_{\mathrm{MS}} & if\ a \notin A \end{cases}$
16. $cdt(p,t) = \begin{cases} VP \geq E_{p,t} & if\ p \in {}^*t \\ \mathsf{true} & if\ p \notin {}^*t \end{cases}$
17. $act(p,t) = \begin{cases} VP := VP - -E_{p,t} + +E_{t,p} & if\ p \in {}^*t \cup t^* \\ \mathsf{skip} & if\ p \notin {}^*t \cup t^* \end{cases}$

In general, the clauses SETS, CONSTANTS and PROPERTIES define all the new sets used in the model. Each element of VARIABLES VP in the B machine M represents the marking $M(p)$ of given place p in the CPN N. The INVARIANT clause assigns a multi-set of different colours to each place. The INITIALISATION clause ensures each place has the same initial value as $m_0(p)$. The DEFINITIONS clause prepares all the variables of transition, arc expression functions and guard functions for the following operations. In OPERATIONS, each OP represents a transition t, the predicate CDT is similar to the CPN enabling condition, and the ACT is the token computation formula. The substitution *SELECT* in each operation is the conditional bounded choice, its actions can only be executed when its condition CDT is true.

4.3.2 Multi-set concept

One of the important features of CPNs is the different data types. A CPN marking corresponds to a multi-set, which contains information of token numbers and token colours. Each behaviour of the transitions is associated to the marking modifications. In order to integrate this mechanism into the B machine framework, the multi-set specifications should be introduced.

In CPN ML language, there is a predefined set of basic types named *simple colour sets*. These simple colour sets can be used to compose structured colour sets using a set of *colour*

set constructors [Jensen and Kristensen 2009]. While in the AMN, only simple data types are predefined. The composed ones will be presented in the next subsection. The mechanism to be introduced must be able to cope with all types of multi-sets. However, it is both unwise and unnecessary to pre-define all types of multi-sets in a single abstract machine. In order to be more adaptable and flexible, we introduce a series of parameterized definitions, which could associate the token numbers with their colours, and allow the modification of tokens via operations. These definitions are written in a "MultiSets.def" file, which could be directly included in any abstract *B* machines, and these definitions can help to complement the clauses in the previous framework. Listing 4.1 gives the content of the file.

Listing 4.1 – Specification of multi-set and their operations

```
DEFINITIONS
  MS(ss)==(ss<->NATURAL);
  Ms_Empty(ss)== {elt|elt: ss * {0}};
  Ms_Subset(ms1,ms2,ss)== !elt.(elt:ss => ms1(elt)>=ms2(elt));
  Ms_Add(ms1,ms2,ss) == (%ee.(ee:ss & ms1(ee):ran(ms1) & ms2(ee):ran(ms2) & ms1(ee)>=0 &
      ms2(ee)>=0 | ms1(ee) + ms2(ee)));
  Ms_Less(ms1,ms2,ss)== (%ee.(ee:ss & ms1(ee):ran(ms1) & ms2(ee):ran(ms2) | ms1(ee) - ms2(
      ee)))
```

The defined function `MS(ss)` is to give the formalism of the marking in *B* machines. The function `Ms_Empty(ss)` generates an empty token with colour type *ss* for the initialization propose and further token operations. The function `Ms_Subset(ms1,ms2,ss)` makes a comparison for multi-sets $ms1$ and $ms2$ which has the same colour type *ss*, the function returns true if and only if the number of each element in $ms1$ is equal to or greater than that in $ms2$. The function `Ms_Add(ms1,ms2,ss)` and `Ms_Less(ms1,ms2,ss)` are addition and subtraction algorithm for multi-sets.

From a mathematical point of view, part of the preconditions in `Ms_Add` and `Ms_Less` seems redundant, such as $ms1(ee) : ran(ms1)$, $ms2(ee) : ran(ms2)$, $ms1(ee) >= 0$, $ms2(ee) >= 0$. But they are some important additional theories, which can help the Atelier-B prover to prove the POs. Using this technique, the multi-set mechanism can be automatically proved. It is applicable for all data types.

For a better understanding, we take an example of a simple colour set to demonstrate

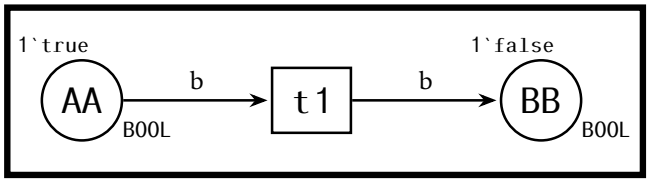


Figure 4.4 – Toy example of multi-set demonstration

the above mentioned definitions. In Fig. 4.4, there is a simple transition with two places. Each place has the same colour type *BOOL*. The initial marking of this CPN is that place *AA* has a token $1`true$, while place *BB* has a $1`false$. After firing the transition $t1$, *AA* has no token and *BB*'s marking is $1`true++1`false$. In the AMN, the token representations will be defined as following.

1. The variables in the *B* machine are also denoted with AA and BB.

2. Each variable has a data type, AA:MS(BOOL), BB:MS(BOOL), where
   MS(BOOL) $\Rightarrow \{true|->n, false|->n\}$, $n \in$ NATURAL

3. Each variable needs initialization,
   AA := Ms_Empty(BOOL) <+ true*{1} $\Rightarrow$ {true|->1, false|->0},
   BB := Ms_Empty(BOOL) <+ false*{1} $\Rightarrow$ {true|->0, false|->1}.

4. After firing transition $t1$,
   AA = Ms_Empty(BOOL) $\Rightarrow$ {true|->0, false|->0},
   BB = Ms_Empty(BOOL) <+ {true, false}*{1} $\Rightarrow$ {true|->1, false|->1}

4.3.3 Colour sets and coloured Petri net declarations

The sets of token colours in CPN are specified by ML programming language. They are classified into two categories, the basic types which inherited from Standard ML are simple colour sets, the others are compound colour sets using the previously declared colour sets. In this paper only non-time related colours are involved.

Table 4.6 – Comparison of basic data types

Colour set	CPN specification	*B* notation
Unit	comprises a single element **colset** name = **unit**	no direct corresponding
Boolean	set of true and false **colset** name = **bool**	BOOL
Integer	numerals without a decimal point **colset** name = **int**	INTEGER
String	sequences of printable characters **colset** name = **string**	STRING Only in operation input
Enumerated	Enumeration values **colset** name = **with** id0|id1|$\cdots$|idn	SET={$E_1, \cdots, E_n$}
Index	sequences of values comprised of an identifier **colset** name=**index** id **with** exp1..exp2	no direct corresponding

In the types supported by the CPN Tools, the simple colour sets and their corresponding data types in *B* language are enumerated in the following Table 4.6. Similar data types for Integer, Boolean and Enumerated can be found in AMN.

For the other simple colour types, we can only have a non-direct translation. The **Unit** colour sets can be represented with a set containing only one element, such as `unit={new_unit}`. The **Index** colour set is a concrete instance of positive integer. So it could be denoted by a finite positive integer set `index={int-exp1..int-exp2}`, where `index:NAT`. The **String** colour sets is a finite string character set. while in the AMN, string type can only be used to type operation input parameters. In practical translation, we recommend using a finite enumerate set to indicate all the strings needed.

For the compound colour sets, CPN use constructors to define more complex colour sets.

The **Product** color sets ***colset*** *Name* = ***product*** *name1* $*$ *name2* $*\cdots*$ *namen*, *where n>=2*. In *B* notation, there is the same Cartesian product '$*$', and definition will be `Name:=name1 * name2 *⋯* namen`.

The **record** colour set ***colset*** *Name* = ***record*** *id1 : name1* $*$ *id2 : name2* $*\cdots*$ *idn : namen*. It is a fixed-length colour set and has the same function as Product. The only difference is that a product colour is position-dependent, while in record colour, each component has a unique label to be identified. In *B* language, the record concept is the same, and the notations are: The set of records is `SET = struct(id1:name1, ⋯, idn:namen)`, an extensive record is `REC = rec(id1:name1, ⋯, idn:namen)`, where *n* is an integer greater than or equal to 1. The access to a record field (quote operator) is `idi'REC`.

The **List** colour set is a variable-length data type. The values are a sequence whose colour must be the same type, ***colset*** *name* = ***list*** *name0*. As one of the most flexible structures in CPNs, the list structure has many pre-defined operations which can achieve functions such as inquiry, self-loop and recursion. But this concrete programming language is only accepted in the Implement phase of *B* method. So the mapping of the lists form CPNs to *B* machines is conditional, and the most suitable candidate is "Sequence expressions". Table 4.7 is a partial mapping of two data structures. The other list functions could only be realized inside the operations.

Table 4.7 – Mapping of *List* data type

CPN specification	***B* notation**
colset Lst = **list** CS	`Lst:=seq(CS)`
nil or [ ] empty list	[ ]
	Continued on next page

Table 4.7 – continued from previous page

CPN specification	*B* notation
hd Lst head, the first element of the list	`first(Lst)`
tl Lst tail, the last element of the list	`tail(Lst)`
length Lst length of list	`size(Lst)`
elt**::**List prepend element as head of list	`elt -> Lst`
rev Lst reverse list	`rev(Lst)`
List.nth(Lst, nth) nth element in list	`Lst(nth)`
List.take(Lst, n) returns first n elements of list	`Lst /|\ n`
List.drop(Lst, n) returns what is left after dropping first n elements of list	`Lst |/ n`
List.null Lst returns true if list is empty	`Lst=[]`
mem Lst elt returns true if element is in the list	`elt:ran(Lst)`
contains Lst1 Lst2 returns true if element is in the list	`ran(Lst1)<:ran(Lst2)`
union Lst1 Lst2 (or Lst1^^Lst2) the concatenation of two lists	`Lst1^Lst2`
ins Lst elt inserts element at the end of list	`Lst<-elt`

Other pre-defined *list* functions are more programming like functions. They cannot be written in a compact form with set theory and first order logic. In practice, depending on the context, *list* functions may appear in different forms in the operations. Sometimes the function can only be realized in the implementation phase, where recursion and loop are allowed in *B* machines. Here is an example (Fig. 4.5) of achieving the same function inside

the operation.

CPN specification	***B* notation**

`ins_new l x`
if x is not a number of list l, then x is inserted at the end of l, otherwise l is returned
	`ann<-ins_new(ll,xx)==`
	`IF xx : ran(ll)`
	`THEN aa := ll`
	`ELSE aa := ll<-xx`
	`END`

Figure 4.5 – Example of mapping coloured Petri net function into *B* operation

Besides the declaration of colour sets, the CPN Tools also allow constant declaration and user-define function declaration. A constant declaration binds a value to an identifier, which then works as a constant. It should be defined in the *B* constant part. The functions of a *B* machine are directly used in the operations, they can be accomplished by the substitution which is similar to programming language, such as IF condition and Case condition. Some simple parameterized function may appear in the definition as reusable modules.

4.3.4 Transformation equivalence

Earlier in this section, the methodology of translation from non-hierarchical CPNs to *B* machines was defined formally, as well as the multi-set concept and some common token colours. With the presented translation rules, we now present a global design for proving that the model translation is equivalent, also known as strongly consistent. To better illus-

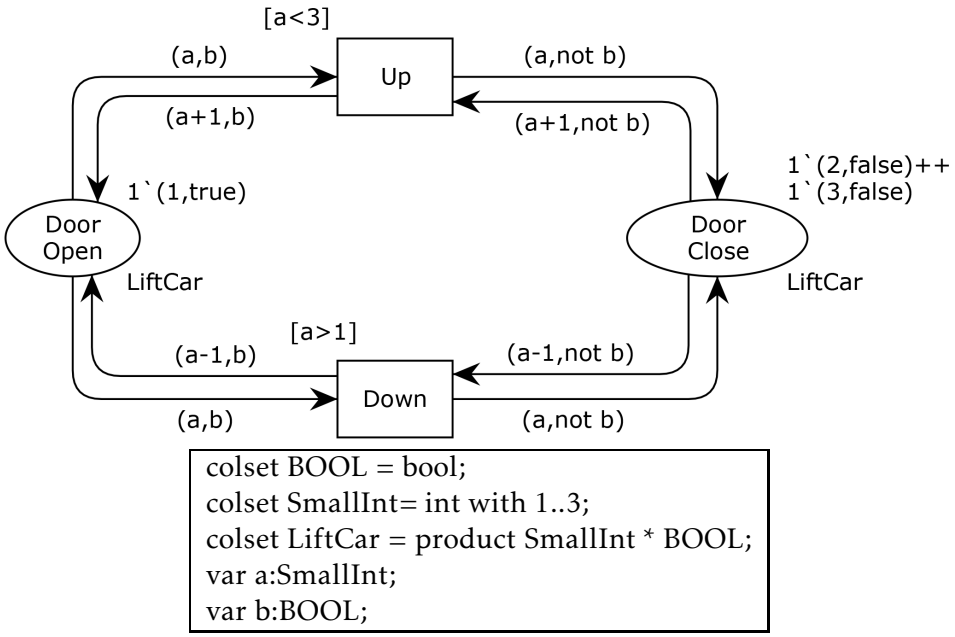


Figure 4.6 – Coloured Petri net of an elevator

trate the verification properties a small example will be introduced.

This is a toy model of an elevator system, the tokens representing the locations and availabilities of the lift car. There are 3 floors, the lift car can move upwards or downwards in a certain state of the system.

Conformance

The basic needs is to ensure that the transformed specifications are well-formed with respect to its transformation language. In the CPNs, the state (or the static property) of the system is represented by the marking, which is the mapping from places into multi-set of tokens. In the transformed specifications, the variables are used to express the system state, which is the marking, $VP == \beta[M(p)]$. The variables obey the same invariants as the marking in CPNs: The colour set invariant and the initialization invariant.

Let a *B* machine Bm be the image of a CPN N after the translation β, where $P = \{p_1, \cdots, p_m\}$. the set of colour set as $\Sigma = \{\sigma_1, \cdots, \sigma_l\}$. The state of the *B* machine $State(Bm)$ is defined as follows:

1. $VP_i \equiv \beta[M(p_i)]$, where $i \in \{1..m\}$
2. $State(Bm) = VP_1 \wedge \cdots \wedge VP_m$
3. $State_0(Bm) = VI_1 \wedge \cdots \wedge VI_m$
4. $Colset_k \equiv \beta[\sigma_k]$, where $k \in \{1..l\}$

Lemma 4.1. State of system. The relation of the system state before and after the translation is shown as:

1. $State(Bm) = VP_1 \wedge \cdots \wedge VP_m = \beta[M(p_i)] \wedge \cdots \wedge \beta[M(p_m)] = \beta[M] \Leftrightarrow M$
2. $State_0(Bm) = VI_1 \wedge \cdots \wedge VI_m = \beta[I(p_i)] \wedge \cdots \wedge \beta[I(p_m)] = \beta[I] \Leftrightarrow I == M_0$
3. $Invariant(Bm) = VP_1 : Colset_1 \wedge \cdots \wedge VP_m : Colset_l \Leftrightarrow C : P \rightarrow \Sigma$

The system state of the elevator example can be transformed into the specifications in Listing 4.2.

Listing 4.2 – Corresponding system state of Fig. 4.6

```
VARIABLES
  OpenDoor, CloseDoor
INVARIANT
  OpenDoor : MS(LiftCar) &
  CloseDoor : MS(LiftCar)
INITIALISATION
  OpenDoor := Ms_Empty(LiftCar) <+ {(1|->TRUE)|->1} ||
  CloseDoor := Ms_Empty(LiftCar) <+ {(2|->FALSE)|->1,(3|->FALSE)|->1}
```

The colour set is related to different data types, and each new type (or non-common type) is pre-defined in the declaration part of CPNs. Such types should also be declared in the *B* machines (see Listing 4.3). They will be used as an important part of invariants, because all the variables and operation must comply with their own "colour (data types)" .

Listing 4.3 – Corresponding colour set of Fig. 4.6

```
CONSTANTS
  SmallInt, LiftCar
PROPERTIES
  SmallInt<<:INT & SmallInt=1..3 &
  LiftCar=(SmallInt*BOOL)
```

In this way, all the system states have the corresponding images and their possible invariants are held at all time.

Execution Semantics

The dynamic properties of the CPNs are expressed by firing the transitions, while such properties can only be approached by operations in *B* machines. According to Definition A.5, a transition is called enabled at marking M, if and only if there are some qualified binding elements.

In order to present the enabling conditions, it is necessary to have the variables of the transition. In Definition A.4, the variable is defined as $t \in T : Var(t) = \{v \mid v \in Var(G(t)) \vee \exists a \in A : v \in Var(E(a))\}$ In the *B* machine, variables of a transition are denoted as $VAR_j = \beta[Var(t_j)]$. According to *B* language syntax, each variable should have its data type in the condition CDT_j of the operation t_j. We denote the image of a binding $b \in B(t_j)$ is b^β, then the enabled conditions for the *B* machines are:

1. $\beta[G(t)\langle b\rangle = true] \Rightarrow \beta[G(t)\langle b\rangle] = true$
   $\Rightarrow \beta[G(t)]\beta[\langle b\rangle] = true \Rightarrow GRD_j\langle b^\beta\rangle = true$

2. $\forall p \in P : \beta[E(p,t)\langle b\rangle \leq M(p)] \Rightarrow \beta[E(p,t)\langle b\rangle] \leq \beta[M(p)]$
   $\Rightarrow E_{p,t}\langle b^\beta\rangle \leq VP \Rightarrow \text{Ms\_Subset}(VP_p, E_{p,t}, Colset_p)\langle b^\beta\rangle.$

So we can claim that the enabling rules of the translation are consistent.

Lemma 4.2. Enabling condition Let a transition t be enabled by a binding element $(t,b) \in BE$, the image of such transition is $OP_t = \beta[t]$, which could also be enabled by the correspondence condition (t, b^β), As they satisfy the following conditions:

1. $\beta[G(t)\langle b\rangle = true] \Rightarrow GRD_j\langle b^\beta\rangle = true$

2. $\forall p \in P : \beta[E(p,t)\langle b\rangle \leq M(p)] \Rightarrow \text{Ms\_Subset}(VP_p, E_{p,t}, Colset_p)\langle b^\beta\rangle.$

Here we have a demonstration of the operation OP_UP, which is the image of transition "UP" in the elevator example.

Listing 4.4 – Corresponding definitions part of Fig. 4.6

```
DEFINITIONS
  "MultiSets.def";

VAR_UP == aa;
  E_OpenDoor_UP== (Ms_Empty(LiftCar) <+ {(aa|->TRUE)|->1});
  E_UP_OpenDoor== (Ms_Empty(LiftCar) <+ {((aa+1)|->TRUE)|->1});
  E_CloseDoor_UP== (Ms_Empty(LiftCar)<+ {((aa+1)|->FALSE)|->1});
  E_UP_CloseDoor== (Ms_Empty(LiftCar)<+ {(aa|->FALSE)|->1});
  GRD_UP== (aa<3);
  CDT_UP== (aa:SmallInt & GRD_UP & Ms_Subset(OpenDoor, E_OpenDoor_UP, LiftCar) & Ms_Subset(
      CloseDoor,E_CloseDoor_UP,LiftCar));
...
```

If a binding element $(t,b) \in BE$ is enabled in M, it may occur, and it leads to a new marking M'. We say that the marking M' is directly reachable from M, which is also written as $M \xrightarrow{(t,b)} M'$.

Lemma 4.3. Token transit. Let a marking M be fired by a binding element $(t,b) \in BE$, and changed into a new marking M'. Then corresponding data changes in B machine are expressed as:

1. $OP_{t_j} \equiv \beta[t_j]$
2. $$\begin{aligned}\beta[M'(p_i)] &= \beta[M(p_i) - -E(p_i,t)\langle b\rangle + +E(t,p_i)\langle b\rangle] \\ &= \beta[M(p_i)] - -\beta[E(p_i,t)\langle b\rangle] + +\beta[E(t,p_i)\langle b\rangle] \\ &= VP_i - -E_{p_i,t}\langle b^\beta\rangle + +E_{t,p_i}\langle b^\beta\rangle \\ &= Ms\_Add(Ms\_Less(VP_i, E_{p_i,t}, Colset_i), E_{t,p_i}, Colset_i)\langle b^\beta\rangle \\ &= VP_i'\end{aligned}$$

   It could also be written in another form:

3. $M(p_i) \xrightarrow{(t_j,b)} M'(p_i) \Leftrightarrow VP \xrightarrow{OP_{t_j}\langle b^\beta\rangle} VP'$.

The complete form of operation OP_UP is shown as follows:

Listing 4.5 – Corresponding operations part of Fig. 4.6

```
OPERATIONS
  Op_UP=
  SELECT #(VAR_UP).(CDT_UP)
  THEN ANY VAR_UP
```

```
    WHERE CDT_UP
    THEN
     OpenDoor := Ms_Add(Ms_Less(OpenDoor,E_OpenDoor_UP,LiftCar),E_UP_OpenDoor,LiftCar) ||
     CloseDoor:= Ms_Add(Ms_Less(CloseDoor,E_CloseDoor_UP,LiftCar),E_UP_CloseDoor,LiftCar)
    END
  END;
```

Hence, for each transition and its image operation, the enabling conditions and occurrence rules are coherent. So the correctness of behaviour (dynamic) properties can be proved.

Model Checking

The CPN and the *B* language are both tool-supported formal methods based on system state space. Since we have proved the static and dynamic properties of the translation by theorem analysis, we can use model checking as an auxiliary method of verification. Model checking has a mathematical representation of a system, and its result consists of a systematic exhaustive exploration of the mathematical model. There is already some research using model checking for verification of model transformations [Calegari and Szasz 2013]. Because the well known "state explosion" limitation exists, this approach can only be applied in a small system or be used as an auxiliary method.

The comparison is shown in Fig. 4.7, the state space of CPN is calculated by the CPN Tools, and the state space of B-machine is calculated by ProB. After initialisation, each model has 3 states , and 4 state changes. Their state space can be considered as the same, and this result reinforces the reliability of our translation method.

4.3.5 Demonstration for mapping non-hierarchical coloured Petri net

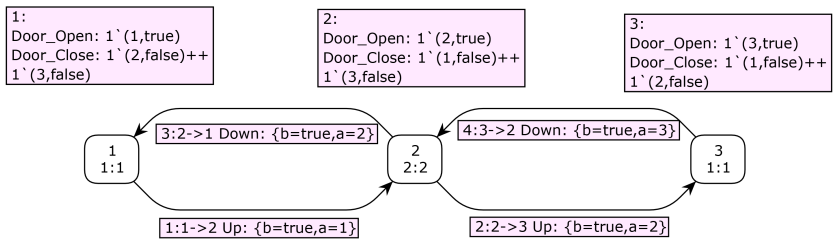
This section presents two examples to illustrate the translation process of non-hierarchical CPN. In order to allow the readers to have a better understanding of our method, only well-known cases are chosen in this paper, rather than some railway based cases.

Dining philosophers problem

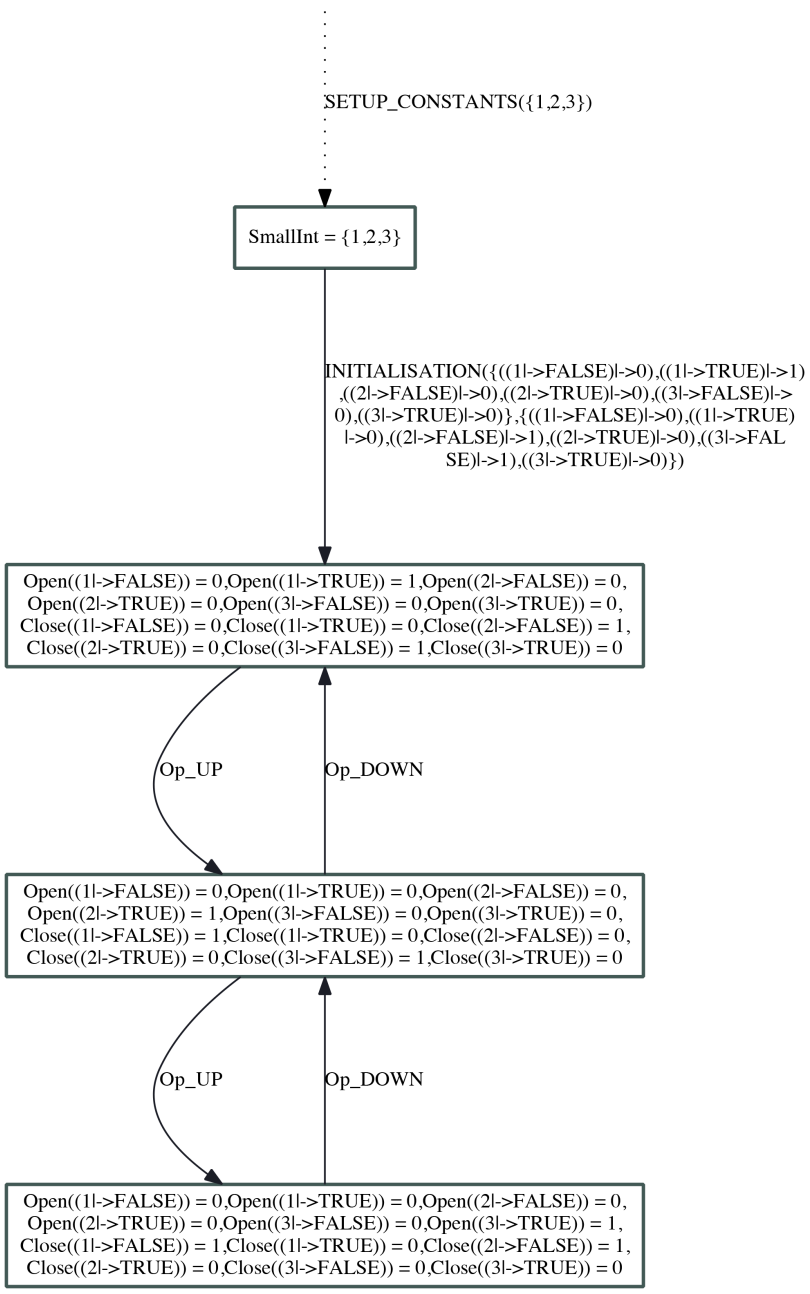
An example which can be seen in Fig. 4.8(a) is the dining philosophers problem. In this case, five Chinese philosophers are having dinner at a round table. There is only one plate and five chopsticks on the table, and each chopstick is placed between two neighbouring philosophers. To eat the dish, each philosopher has to use two chopsticks next to him. Once a philosopher starts eating, his two neighbours have to wait until the chopsticks are unoccupied.

This system is modelled by CPNs shown in Fig. 4.8(b), which could well express the

concept of synchronization and concurrency. Philosophers are defined as a set of PH, and the chopsticks are defined as a set of CS. The $Chopsticks()$ is a mapping function which

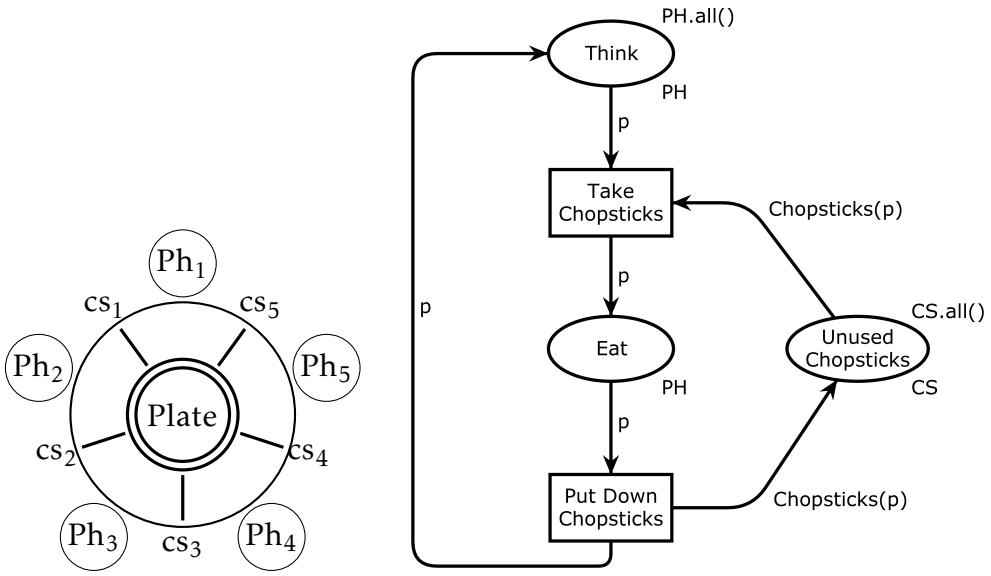


(a) State space of CP-net



(b) State space of B-machine

Figure 4.7 – State space comparison between the transformation



(a) Problem describesion (b) CP-net of dining philosophers

```
val n = 5;
colset PH = index ph with 1..n;
colset CS = index cs with 1..n;
var p: PH;
fun Chopsticks(ph(i)) = 1‘cs(i) ++ 1‘cs(if i=n then 1 else i+1);
```

Figure 4.8 – Dining philosophers problem

indicates the relationship between the philosophers and the chopsticks he can use.

The translation job could be roughly divided into three steps. First, the basic framework of the *B* machine should be built. Each place and each transition of the CPN is mapped into the entries of variables and operations in the *B* machine. Then, the colour informations and user-defined CPN declarations are filled into the appropriate clauses. Finally, the detailed specifications of each arc, guard and transition function are complemented. For a clear mind, these steps are described separately. But in practice, they can be applied at the same time.

The corresponding *B* machine is shown in Listing 4.6. In this machine, we named the variables and the operations in the same way as when they are in the places and transitions. The expressions starting with "E_" are the arc expressions. The source and destination of each arc is noted in an abbreviate form. For example, the arc `E_T_TkChs` means the arc from place *Think* to transition *Take_Chopsticks*. Similarly, the arc `E_PdChs_U` means the arc from transition *Put_Down_Chopsticks* to place *Unused*. The expression starting with "CDT_" is the enabling condition of each operation. Statements "skip" and "TRUE" are omitted in the machine. Because for each generalized substitution S, it holds that $S||skip = S$, and for each predicate P that $P \wedge TRUE = P$.

Listing 4.6 – *B*-machine for dining philosophers

```
MACHINE  dining_philosophers
```

```
CONSTANTS  val_n, PH, CS
PROPERTIES
   val_n : INT & val_n = 5 &
   PH <<:INT & PH = 1..val_n &
   CS = 1..val_n
VARIABLES
   Think, Eat, Unused
INVARIANT
   Think : MS(PH) & Eat : MS(PH) & Unused : MS(CS)
INITIALISATION
   Think := Ms_Empty(PH) <+ PH*{1} ||
   Eat := Ms_Empty(PH) ||
   Unused := Ms_Empty(CS) <+ CS*{1}
DEFINITIONS
   "MultiSets.def"
   Chopsticks(pp)== {pp,((pp mod val_n)+1)};

   VAR_TkChs== pp;
     E_T_TkChs== (Ms_Empty(PH) <+ {pp|->1});
     E_U_TkChs== (Ms_Empty(CS) <+ Chopsticks(pp)*{1});
     E_TkChs_E== (Ms_Empty(PH) <+ {pp|->1});
   CDT_TkChs== pp:PH & Ms_Subset(Think,E_T_TkChs,PH) & Ms_Subset(Unused,E_U_TkChs,CS);

   VAR_PdChs== pp;
     E_E_PdChs== (Ms_Empty(PH) <+ {pp|->1});
     E_PdChs_T== (Ms_Empty(PH) <+ {pp|->1});
     E_PdChs_U== (Ms_Empty(CS) <+ Chopsticks(pp)*{1});
   CDT_PdChs== pp:PH & Ms_Subset(Eat,E_E_PdChs,PH)

OPERATIONS
   TakeChopsticks=
      SELECT #(VAR_TkChs).(CDT_TkChs)
      THEN ANY VAR_TkChs WHERE CDT_TkChs
         THEN
            Think:= Ms_Less(Think,E_T_TkChs,PH) ||
            Unused:= Ms_Less(Unused,E_U_TkChs,CS) ||
            Eat:= Ms_Add(Eat,E_TkChs_E,PH)
         END
      END;

   PutDownChopsticks=
      SELECT #(VAR_PdChs).(CDT_PdChs)
      THEN ANY VAR_PdChs WHERE CDT_PdChs
         THEN
            Think:=Ms_Add(Think,E_PdChs_T,PH) ||
            Unused:=Ms_Add(Unused,E_PdChs_U,CS) ||
            Eat:=Ms_Less(Eat,E_E_PdChs,PH)
```

```
        END
    END
END
```

In this machine, the user-defined function "Chopsticks(p)" is quite different from its original form in CPN, `fun Chopsticks(ph(i))= 1`cs(i) ++ 1`cs (if i=n then 1 else i+1)`. That is because the conditional bounded choice "SELECT" will use this function in the predicate. But the original function is a "IF-THEN-ELSE-END" substitution which cannot be a predicate. So this function should be written as a formula, which can be proved by *B* language.

The state of the dining_philosophers machine is shown in Table 4.8. It has 9 proof obligations, and could be automatically proved (PRa) by Atelier-B Ver.4.1.0.

Table 4.8 – Component status for dining_philosophers

AutoProved

	nPO	nPRi	nPRa	nUn	%Pr
Initialisation	3	0	3	0	100
Take_Chopsticks	3	0	3	0	100
Put_Down_Chopsticks	3	0	3	0	100
dining_philosophers	**9**	**0**	**9**	**0**	**100**

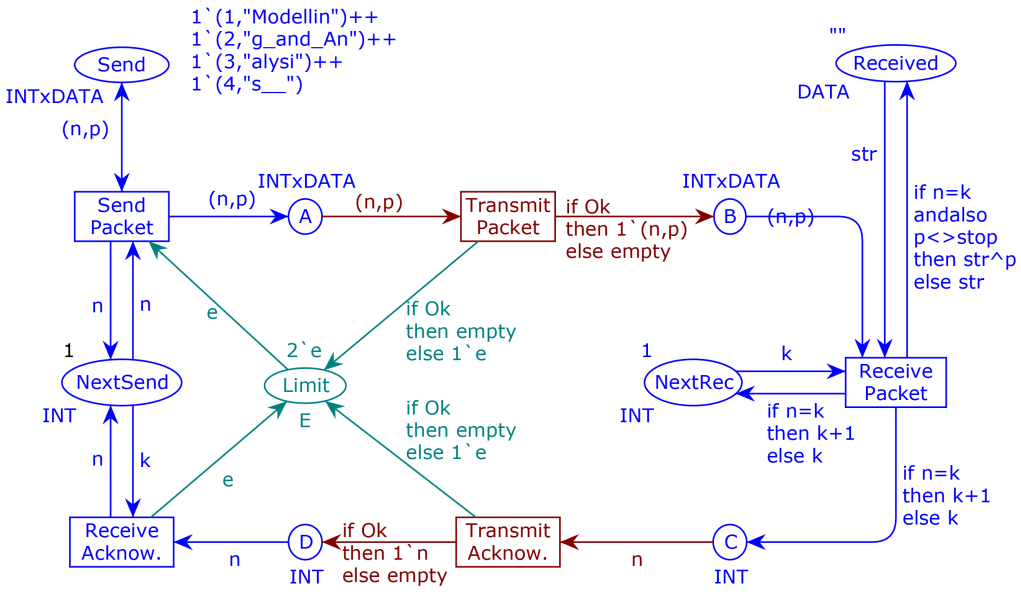


Figure 4.9 – Coloured Petri net of simple protocol

Simple protocol

In the previous example, all the arc inscriptions are quite simple, and only simple colour sets are involved. So the following example will demonstrate with compound colour sets and conditional arc inscriptions.

The example in Fig. 4.9 describes a simple protocol by which a sender can transfer a number of packets to a receiver. At most two packets could be sent at once. The communication process may lose packets, and different packets may overtake each other. Hence, it may be necessary to retransmit packets and to ignore doublets and packets that are out of order.

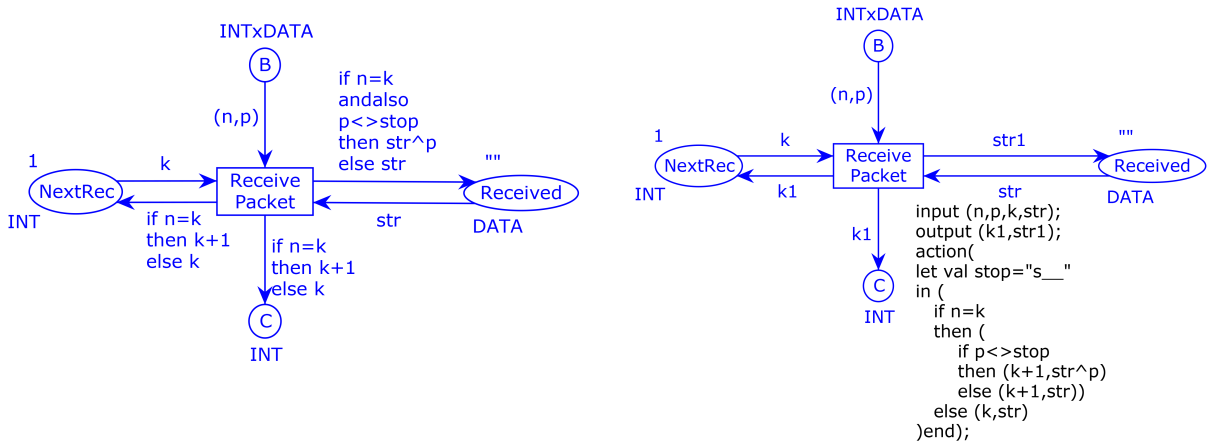
The string colour set in the *B* machine becomes an enumerate set with all necessary strings. and the combined string result could be a string sequence. The declarative part of the *B* machine is shown in Listing 4.7.

Listing 4.7 – *B*-machine declarative part of simple protocol

```
MACHINE  SimpleProtocol
SETS  DATA={Modellin, g_and_An, alysi, s__}; EV={ee}
CONSTANTS
   INTxDATA,val_stop,DataLst
PROPERTIES
   INTxDATA = INTEGER*DATA & val_stop = s__ & DataLst = seq(DATA)
VARIABLES
   Send, Limit, Received, NextSend, NextRec, AA, BB, CC, DD
INVARIANT
   Send : MS(INTxDATA) & Limit : MS(EV) &
   Received : MS(DataLst) &
   NextSend : MS(INTEGER) & NextRec : MS(INTEGER) &
   AA : MS(INTxDATA) & BB : MS(INTxDATA) &
   CC : MS(INTEGER) & DD : MS(INTEGER)
INITIALISATION
   Send := Ms_Empty(INTxDATA) <+ {(1|->Modellin)|->1,(2|->g_and_An)|->1,(3|->alysi)
       |->1,(4|->s__)|->1} ||
   Limit := Ms_Empty(EV) <+ {ee|->2} ||
   Received := Ms_Empty(DataLst) <+ {[]|->1} ||
   NextSend := Ms_Empty(INTEGER) <+ {1|->1} ||
   NextRec := Ms_Empty(INTEGER) <+ {1|->1} ||
   AA := Ms_Empty(INTxDATA) ||
   BB := Ms_Empty(INTxDATA) ||
   CC := Ms_Empty(INTEGER) ||
   DD := Ms_Empty(INTEGER)
```

There are five transitions in the model. The transitions *TransmitPacket*, *TransmitAcknow* and *ReceivePacket* have conditional output arcs. In our framework the arcs are defined in the Definition clause of the machine, but the conditional expressions cannot be applied in

Definitions. So a transformation of transition is needed to generate a new formalism, which has the same function but simpler arcs.



(a) Original transition (b) New transition

Figure 4.10 – Demo of transition transformation

The transformation ensures consistency of original transition but has a different notation. All the arc inscriptions will be concentrated inside the transition function, and maintain conciseness of all the arc expressions. A transformation demo of transition *ReceivePacket* is shown in Fig. 4.10.

After transforming the three transitions into the simple form, we can continue to translate the CPN, and part of the operations are shown in Listing 4.8, where TrPck is short for "TransmitPacket", RcPck is short for "ReceivePacket". The state of the this machine is shown in Table 4.9.

Listing 4.8 – Part of operations of simple protocol

```
DEFINITIONS
  "MultiSets.def"
  // TransmitPacket
  VAR_TrPck== nn,pp;
    E_A_TrPck== (Ms_Empty(INTxDATA) <+ {(nn|->pp)|->1});
    E_TrPck_B== (Ms_Empty(INTxDATA) <+ {(nn|->pp)|->1});
    E_TrPck_L== (Ms_Empty(EV) <+ {ee|->1});
  CDT_TrPck== (nn:INTEGER & pp:DATA & Ms_Subset(AA, E_A_TrPck, INTxDATA));

  // ReceivePacket
  VAR_RcPck== nn,pp,str,kk;
    E_B_RcPck== (Ms_Empty(INTxDATA) <+ {(nn|->pp)|->1});
    E_R_RcPck== (Ms_Empty(DataLst) <+ {str|->1});
    E_RcPck_R(str1)== (Ms_Empty(DataLst) <+ {str1|->1});
    E_N_RcPck== (Ms_Empty(INTEGER) <+ {kk|->1});
    E_RcPck_N(kk1)== (Ms_Empty(INTEGER) <+ {kk1|->1});
    E_RcPck_C(kk1)== (Ms_Empty(INTEGER) <+ {kk1|->1});
```

```
  CDT_RcPck== (nn:INTEGER & pp:DATA & str:DataLst & kk:INTEGER & Ms_Subset(BB,E_B_RcPck,
      INTxDATA) & Ms_Subset(NextRec,E_N_RcPck,INTEGER) & Ms_Subset(Received,E_R_RcPck,
      DataLst))
OPERATIONS
  Op_TransmitPacket=
    SELECT #(VAR_TrPck).(CDT_TrPck)
    THEN ANY VAR_TrPck,OK
      WHERE CDT_TrPck & OK:BOOL
      THEN
        AA:= Ms_Less(AA,E_A_TrPck,INTxDATA) ||
        IF OK=TRUE
        THEN BB:= Ms_Add(BB,E_TrPck_B,INTxDATA)
        ELSE Limit:= Ms_Add(Limit,E_TrPck_L,EV)
        END
      END
    END;
  Op_ReceivePacket=
    SELECT #(VAR_RcPck).(CDT_RcPck)
    THEN ANY VAR_RcPck
      WHERE CDT_RcPck
      THEN
        BB:= Ms_Less(BB,E_B_RcPck,Colset_INTxDATA) ||
        IF nn=kk
        THEN
         IF pp/=val_stop
         THEN Received := Ms_Add(Ms_Less(Received,E_R_RcPck,DataLst),E_RcPck_R(str<-pp),
             DataLst)
         ELSE Received := Ms_Add(Ms_Less(Received,E_R_RcPck,DataLst),E_RcPck_R(str),
             DataLst)
         END ||
         NextRec := Ms_Add(Ms_Less(NextRec,E_N_RcPck,INTEGER),E_RcPck_N(kk+1),INTEGER) ||
         CC := Ms_Add(CC,E_RcPck_C(kk+1),INTEGER)
        ELSE
         Received:=Ms_Add(Ms_Less(Received,E_R_RcPck,DataLst),E_RcPck_R(str),DataLst) ||
         NextRec:= Ms_Add(Ms_Less(NextRec,E_N_RcPck,INTEGER),E_RcPck_N(kk),INTEGER) ||
         CC := Ms_Add(CC,E_RcPck_C(kk),INTEGER)
        END
      END
    END
```

4.4 Extending transformation using hierarchical concept

In this section, the hierarchical concept is discussed and two solutions are suggested. Then, we choose the *multi-systems* solution to expand our research.

Table 4.9 – Component status for SimpleProtocol

AutoProved

	nPO	nPRi	nPRa	nUn	%Pr
Initialisation	6	0	6	0	100
Op_SendPacket	2	0	2	0	100
Op_ReceiveAcknow	3	0	3	0	100
Op_TransmitPacket	2	0	2	0	100
Op_TransmitAcknow	2	0	2	0	100
Op_ReceivePacket	4	0	4	0	100
SimpleProtocol	**19**	**0**	**19**	**0**	**100**

4.4.1 Hierarchy concept

A HCPN organised a set of module instances, in a way similar to that in which programs are organized into modules. The hierarchy concept makes it possible to build up a system at different abstraction levels. Each module is an individual CPN. If $T_{sub} \neq \emptyset$ exists, then any substitution transition $t \in T_{sub}$ could be further represented by a lower level module. The connections with higher-level module are indicated by P_{port}.

The hierarchy concept of CPNs lets a system model have several nets. In practice, the hierarchy concept can solve two kinds of problems. First, *single system*, which means a large single system is constituted by multiple non-autonomic systems, such as a train's automatic driving system. This is a complete system, but is formed by different functional components. Second, *multi-systems* (or composite system), which means a system is integrated or composed of several autonomic systems, such as a railway signalling control process, involving train control centre, track side equipment and trains.

In this paper, we only deal with the second concept of hierarchy. To be precise, a CPN of a large system will be translated into a set of abstract machines. Each single net will have its own image *B* machine under the translation.

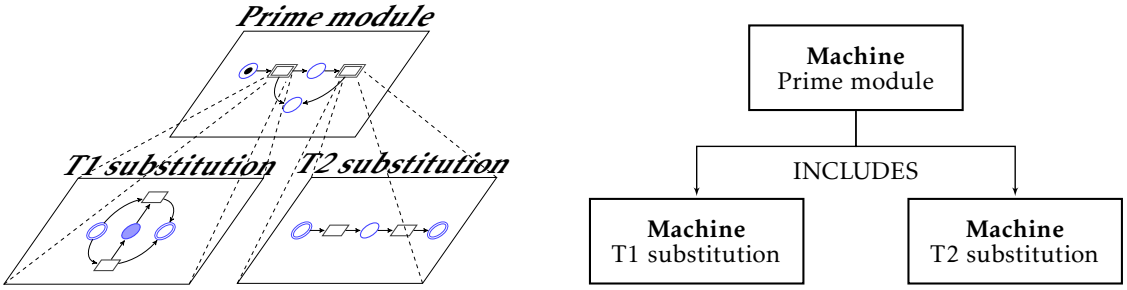
In CPNs, tokens (or markings) can be modified via transitions of different levels. For

Table 4.10 – Comparison of composition clauses

	Operation of M_2		
Constituents of M_1	SEES	INCLUDES	USES
Sets, enumerated set elements, concrete constants,	read	read	read
Abstract constants	read	read	read
Concrete variable	read - non write	read - non write	read - non write
Abstract variable	read - non write	read - non write	read - non write
Operations	read - non write	read write	

example, places directly connected to the T_{sub} in the upper level can be modified by the transitions inside the T_{sub}. Therefore, the relations between each *B* machine need to comply with the same rule. In the framework of our early research, the places and the transitions correspond to the variables and the operations in a *B* machine. Then, the next question is to find a proper composition clause, which allows an operation to read and write the variables inside another machine. The visibility comparison of each composition clause is shown in Table 4.10.

The most approximate clause is INCLUDES, which is used to build a new abstract machine using other abstract machines. It could bring together the components (sets, constants, variables) of machine instances as well as their properties (Properties and Invariant clauses) [ClearSy 2014]. The schematic diagram is shown in Fig. 4.11.



(a) Coloured Petri net Hierarchy of module instances

(b) *B* project structure

Figure 4.11 – Comparison of hierarchy structure between coloured Petri nets and *B* machines

4.4.2 Framework of model transformation

According to Definition A.9, let $CPN_H = (S, SF, PS, FS)$ be a HCPN, and $s_0^*, s_1^*, s_2^* \in S$ and $s_0^* \neq s_1^*$, $s_0^* \neq s_2^*$, $s_1^* \neq s_2^*$ be three instances of different modules. $s_0^* = ((P^{s_0^*}, T^{s_0^*}, A^{s_0^*}, \Sigma^{s_0^*}, V^{s_0^*}, C^{s_0^*}, G^{s_0^*}, E^{s_0^*}, I^{s_0^*}), P_{port}^{s_0^*}, T_{sub}^{s_0^*}, PT^{s_0^*})$, s_1^* and s_2^* has the similar form. If s_1^* is a sub-module of s_0^*, it should satisfy the following condition:

$$\exists(sf, t_0).\{sf \in SF \wedge t_0 \in T_{sub}^{s_0^*} \wedge sf = t_0 \rightarrow s_1^*\}$$

Let β be a the mapping function $\beta : CPN \rightarrow AMNs$, then the image of a net N in the CPN model N under β is a *B* machine Bm, $Bm = \beta[N]$. If such a previous condition holds, we can declare that the machine Bm_1 should "INCLUDES" the machine Bm_0.

Before we give the framework of the transformation, it is important to introduce how to express the hierarchy concept in the *B* machines.

Framework of hierarchy in *B* machines

In HCPNs, the data transmissions are achieved by *compound places* (in Definition A.11) across the hierarchical structure of the model. Such compound places are located in different modules but their values are glued together. However, there is no such relevant mechanism in the *B* notation. A *B* machine can only have its own local variables (local places), while other types of variables, such as port places and fusion places, should be defined in other higher-level machines, and be accessed by operations.

In HCPNs, a compound place can have many instances. In contrast, in *B* machines all the places of a same compound place only allow a unique instance, and the set of all the unique place instances is denoted as PI_{one}, where $PI_{one} \subset (P_{port} \cup FS)$. The locations of each place instance $p \in PI_{one}$ in *B* machines should obey the following two rules:

(i) For a fusion place $fs \in FS$, the instance of fs^* should be defined in the **prime module** S_{PM}. (in Definition A.10), which is also the root machine of the *B* project;

(ii) For a port-socket relation pair (p_1, p_2), where $(p_1, p_2) \in [p]$ and an arc $(s_1^*, t, s_2^*) \in A_{IH}$ exists, $p_1 \in (P^{s_1^*} - P_{port}^{s_1^*})$, $p_2 \in P^{s_2^*}$. Then p_1 will be defined in machine s_1^*, while p_2 will not appear in any machines.

In *B* machines, if a machine M_1 includes another machine M_0, then the INCLUDES mechanism specifies that M_1 can only modify the variables in M_0 by the operations in M_0. So some access operations need to be pre-defined in the same machine with the instance of the compound places. Supposing there are two different module instants s_0^* and s_1^*, that s_0^* is included by s_1^*, consequently s_0^* should prepare some operations for s_1^*. The operations may be parametrized functions which can be recalled by other machines when s_0^* is included.

The access methods of compound places are defined as following rules.

(i) The place instances VP can be read directly by included machines, which means they can be used in CDT expressions of another machine.

(ii) Let s_1 be a sub-module of s_0, and $(pi_0, s_0) \sim_{cp} (pi_1, s_1)$. If a transition $t \in T^{s_1}$ exists, that is only connected to one compound place instance, where $({}^*t \cup t^*) \cap (FS \cup P_{port}^{s_1}) = \{pi_1\}$, then the image of such compound place pi is `VPI`, defined in M_0 with data type `Colset_PI`. Meanwhile, in machine M_0, the following pre-defined operations may exist for the operation $\beta[t]$ in M_1.

(a) Addition

```
OpArc_Add_VPI(ms)=
PRE ms:MS(Colset_PI)
THEN VPI:=Ms_Add(VPI,ms,Colset_PI)
END
```

(b) Subtraction

```
OpArc_Less_VPI(ms)=
PRE ms:MS(Colset_PI) & Ms_Subset(VPI,ms,Colset_PI)
THEN VPI:=Ms_Less(VPI,ms,Colset_PI)
END
```

(iii) Let s_1 be a sub-module of s_0, if a transition $t \in T^{s_1}$ connected to more than one compound places. The set of these places denotes $P^{s_1}_{ins}$ and the corresponding compound place set in s_0 denotes $P^{s_0}_{ins}$. For $\forall p \in P^{s_1}_{ins}$, $\exists pi \in P^{s_0}_{ins}$ such that $(pi, s_0) \sim_{cp} (p, s_1)$ and $({}^*t \cup t^*) \cap (FS \cup P^{s_1}_{port}) = P^{s_1}_{ins}$. Then the operation pre-defined in M_0 should be as

```
OpArc_T(ms_P_1, ···, ms_P_n)= PRE
ms_P_i:MS(Colset_P_i) &        // for ST(p_i) =IN
···
ms_P_j:MS(Colset_P_j) &        // for ST(p_j) =OUT
Ms_Subset(VP_j,ms_P_j,Colset_P_j) &
···
ms_P_k:MS(Colset_P_k) &        // for ST(p_k) =I/O
Ms_Subset(VP_k,ms_P_k,Colset_P_k) &
···
THEN
VP_i:=Ms_Add(VP_i,ms_P_i,Colset_P_i)
···
VP_j:=Ms_Less(VP_j,ms_P_j,Colset_P_j)
···
VP_k:=Ms_Less(Ms_Add(VP_k,ms_P_k^in,Colset_P_k),ms_P_k^out,Colset_P_k)
···
END
```

The reason for combining all the actions into one operation is to keep the consistency of the invariants, because in the Atelier-B, the *Type Checker* Specifies "Two included operations cannot be called in parallel" [ClearSy 2011].

Framework of machine details

In order to describe the details of a transformed machine, we give the following assumptions. A module instance s^*_1 that includes s^*_0, and s^*_1 is included by s^*_2. Then we can have:

(i) the set of all the places is $P^{s^*_1}$,

(ii) the set of port places $P^{s^*_1}_{port}$,

(iii) the set of fusion places $P_{fs}^{s_1^*}$,

(iv) the pure local places are $P_{pure}^{s_1^*} = P^{s_1^*} - P_{port}^{s_1^*} - P_{fs}^{s_1^*} = \{p_1, \cdots, p_m\}$,

MACHINE M_1
INCLUDES M_0
SETS $Colset_1, Colset_2, \cdots, Colset_{k-1}$
CONSTANTS $Colset_k, Colset_{k+1}, \cdots, Colset_l$
PROPERTIES
$Colset_k := EXPR_V^k$ & $Colset_{k+1} := EXPR_V^{k+1}$ & $\cdots$ & $Colset_l := EXPR_V^l$
VARIABLES $VP_1, VP_2, \cdots, VP_m$
INVARIANT
$VP_1 : (Colset_1)_{\text{MS}}$ & $VP_2 : (Colset_2)_{\text{MS}}$ & $\cdots$ & $VP_m : (Colset_l)_{\text{MS}}$
INITIALISATION
$VP_1 := VI_1 \parallel VP_2 := VI_2 \parallel \cdots \parallel VP_m := VI_m$
DEFINITIONS
"MultiSets.def";
$\cdots$
$\text{VAR}_j == EXPR_V^{t_j}$; // For trainsition T_j
$E_{p_1,t_j} == EXPR_{\text{VAR}_j}$;
$E_{p_2,t_j} == EXPR_{\text{VAR}_j}$;
$\cdots$
$E_{t_j,p_1} == EXPR_{\text{VAR}_j}$;
$E_{t_j,p_2} == EXPR_{\text{VAR}_j}$;
$\cdots$
$GRD_j == EXPR_V$;
$CDT_j == EXPR_V$;
$\cdots$
OPERATIONS
OP_1= SELECT #(VAR_1).(CDT_1) THEN
ANY VAR_1 WHERE CDT_1 THEN ACT_1 END END;
$\cdots$
OP_n= SELECT #(VAR_n).(CDT_n) THEN
ANY VAR_n WHERE CDT_n THEN ACT_n END END;
$\cdots$
$OpArc_1(ms)$= PRE $ArcCDT_1$ THEN $ArcACT_1$ END;
$\cdots$
$OpArc_v(ms)$= PRE $ArcCDT_v$ THEN $ArcACT_v$ END;
END

Figure 4.12 – The framework of a B machine transformed from hierarchical coloured Petri net

(v) the pure non-substitution transitions are $T^{s_1^*}_{pure} = T^{s_1^*} - T^{s_1^*}_{sub} = \{t_1, \cdots, t_n\}$,

(vi) the set of transitions that are both in s_2^* and are directly connected with $P^{s_2^*}_{port}$ is $T^{s_2^*}_{dc} = \{t \mid t \in T^{s_2^*} \wedge \exists p \in P^{s_2^*}_{port} \Rightarrow (p,t) \cup (t,p) \neq \phi\} = \{t_1, \cdots, t_v\}$,

(vii) the set of local colour sets are $\Sigma^{s_1^*} = \{\sigma_1, \cdots, \sigma_{k-1}, \sigma_k, \cdots, \sigma_l\}$,

(viii) the places that should be defined in the machine M_0 are $P^{s_0^*} = P^{s_1^*}_{port} + P^{s_1^*}_{fs}$,

(ix) the pre-defined operations in M_0 are $OpArc^{s_0} = \{OpArc_1^{s_0}, \cdots, OpArc_u^{s_0}\}$.

Then we defined that the image of a non-hierarchical CPN s_1^* under mapping β is a *B* machine M_1, $M_1 = \beta[s_1^*]$ with the following structure in Fig. 4.12. For such a mapping, the general mapping relations that are defined as follows hold:

1. "MultiSets.def" is an included definition file, which stores the multi-set algorithms that were introduced in Section 4.3.2.

For each token colour $\forall k \in \{1..l\}$, the corresponding relations are:

2. $Colset_k = \beta[\sigma_k]$ is a colour set, where $\sigma_k = C(p_i) \in \Sigma^{s_1^*}$, $p_n \in P^{s_1^*}_{pure}$.

3. $(Colset_k)_{\mathrm{MS}} = \beta[C(p_i)_{\mathrm{MS}}]$, is the multi-set of colours.

For each place related element $\forall i \in \{1..m\}$, the relations are shown as follows.

4. $VP_i = \beta[M(p_i)]$, is the marking of each place, where $p_i \in P^{s_1^*}_{pure}$.

5. $VP_i \in (Colset_k)_{\mathrm{MS}}$, where $Type[VP_i] = Colset_k$.

6. $VI_i = \beta[I(p_i)\langle\rangle] = \beta[M_0(p_n)]$ is the initial marking.

7. The markings M of the system are defined as $M = \sum_1^m VP$.

For each transition related element $\forall j \in \{1..n\}$, the relations are shown as below:

8. $OP_j = \beta[t_j]$ is the transition, where $t_j \in T^{s_1^*}_{pure}$.

9. $VAR_j = \beta[Var(t_j)]$ is the variables of transition t_j.

10. $GRD_j = \beta[G(t_j)]$ is the guard function of transition t_j.

11. CDT_j is the condition of each operation, $CDT_m = (VAR_j \in Colset_k) \wedge cdt(p_1, t_j) \wedge cdt(p_2, t_j) \wedge \cdots \wedge cdt(p_i, t_n) \wedge GRD_j$.

12. $ACT_j = act(p_1, t_j) \wedge act(p_2, t_j) \wedge ... \wedge act(p_i, t_n) \wedge OpArc_1^{s_0} \wedge \cdots \wedge OpArc_u^{s_0}$.

The details for $\forall p,t\ (p \in P_{pure}^{s_1^*}, t \in T_{pure}^{s_1^*})$ have the relations as:

13. E_a is the arc expression, where $E_{p,t} = \beta[E(p,t)]$ and $E_{t,p} = \beta[E(t,p)]$.

14. $$E_a = \begin{cases} EXPR_t \in (Colset_k)_{\mathrm{MS}} & if\ a \in A \\ \Phi_{\mathrm{MS}} & if\ a \notin A \end{cases}$$

The enabling conditions are the same as those in non-hierarchical CPNs. They are achieved by the nested usage of two substitution statements "SELECT-THEN-END" and "ANY-THEN-END". Their combination effect can represent the enabling conditions and random token choice.

15. $$cdt(p,t) = \begin{cases} VP \geq E_{p,t} & if\ p \in {}^*t \\ \mathsf{true} & if\ p \notin {}^*t \end{cases}$$

After an operation is activated, the occurrence rules are executed by actions. For any pure place in s_1, where $p \in P_{pure}^{s_1^*}$, the related actions in machine M_1 are defined as:

16. $$act(p,t) = \begin{cases} VP := VP - -E_{p,t} + +E_{t,p} & if\ p \in {}^*t \cup t^* \\ \mathsf{skip} & if\ p \notin {}^*t \cup t^* \end{cases}$$

But for compound place related transitions, if $p', p \in [p]$ where $p' \in s_0$, $p \in s_1$, $p' \in PI_{one}$, $p \notin PI_{one}$. Place p' has its image VP in B machine M_1, while place p does not have any images. Their actions should be achieved by recalling the operations pre-defined in machine M_0.

17. $$act(p,t) = \begin{cases} OpArc(ms) & if\ p \in {}^*t \cup t^* \\ \mathsf{skip} & if\ p \notin {}^*t \cup t^* \end{cases}$$

4.4.3 Transformation equivalence

In the previous subsection, we introduce a formal systematic transformation, which consists of a collection of mapping rules. With such rules, it is important to prove the correctness and consistency of our methodology.

Static properties

In the CPNs, the state of a system (static properties) is specified with its marking, while in abstract B machines, it will be presented in the form of variables.

Lemma 4.4. State of system. Let a CPN has a set of sub-nets $\{N_1, \cdots, N_m\}$. Each net N_i has its own image $Bm_i = \beta[N_i]$. In N_i, we denote the set of pure places as $P^i = \{p_1^i, \cdots, p_n^i\}$, the set of colour set as $\Sigma^i = \{\sigma_1^i, \cdots, \sigma_s^i\}$. The state of the system is defined as:

1. $VP_j^i \equiv \beta[M(p_j^i)] \equiv \beta[M([p_j^i])]$, where $i \in \{1..m\}, j \in \{1..n\}$.

2. $State(Bm_i) = VP_1^i \wedge \cdots \wedge VP_m^i \Leftrightarrow M(N_i)$.

3. $State_0(Bm) = VI_1^i \wedge \cdots \wedge VI_m^i \Leftrightarrow I^i = M_0(N_i)$.

4. $M = \sum_{i=1}^{m} M(N_i) = \sum_{(i=1,j=1)}^{(m,n)} VP_j^i = \sum_{i=1}^{m} State(Bm_i)$.

The invariants of a token in CPN is its colour set $\sigma \in \Sigma$, while the corresponding invariants in *B* machine are listed in *Invariant* clause. The invariants of the state are defined as:

5. $Colset_l^i \equiv \beta[\sigma_l^i]$, where $l \in \{1..s\}$

6. $(Colset_l^i)_{\mathrm{MS}} \equiv \beta[(C_{p_j^i})_{\mathrm{MS}}]$, where $C(p_j^i) = \sigma_l^i \in \Sigma^i$

7. $Invariant(Bm_i)$
   $= VP_1^i : Colset_1^i \wedge \cdots \wedge VP_m^i : Colset_s^i$
   $\Leftrightarrow C^i : P^i \to \Sigma^i$

8. $C : P \to \Sigma = \sum_{i=1}^{m} M(P^i \to \Sigma^i) = \sum_{i=1}^{m} Invariant(Bm_i)$

With this mapping relation, the consistency of static properties before and after transformation is guaranteed.

Dynamic properties

The Dynamic properties (state changes) of the CPNs is the performance of transitions, while the same mechanism is expressed by operations in *B* machines. According to Definition A.13, the process of firing a transition can be divided into two parts: enabling, token transit.

Lemma 4.5. Enabling condition. Let a set of pure transitions be $T^i = \{T_1^i, \cdots, T_w^i\} \in N^i$. For a pre-enabled transition $t_v^i \in T^i$, the variables of t_v^i are $Var(t_v^i)$, and its binding value is $b(Var(t_v^i))$. We denote the mapping image of a given binding $b_v^i \in B(t_v^i)$ as $b_v i, \beta$. Then in machine Bm_i, the corresponding conditions are expressed as:

1. $VAR_v^i \equiv \beta[Var(t_v^i)]$

2. $b_v^{i,\beta} \equiv \beta[b_v^i]$

3. $\beta[G(t_v^i)\langle b_v^i\rangle] = true \Rightarrow \beta[G(t_v^i)]\beta[\langle b_v^i\rangle] = true$
   $\Rightarrow GRD_v^i\langle b_v^{i,\beta}\rangle = true$

4. $\beta[E(p_i, t_v)\langle b\rangle \leq M(p_j^i)] \Rightarrow \beta[E(p_i, t_v)\langle b\rangle] \leq \beta[M(p_i)]$.
   $\Rightarrow E_{p_j^i, t_v^i}\langle b^\beta\rangle \leq VP_i \Rightarrow \mathrm{Ms\_Subset}(VP_j^i, E_{p_j^i, t_v^i}, Colset_l^i)\langle b^\beta\rangle$,
   $where \quad Colset_l^i = \beta[C(p_j^i)]$.

Definition 4.1. Token transit. Let transition t_v^i be enabled by marking M with binding element $(t_v^i, b_v^i) \in BE$. After firing t_v^i, the new marking is M'. Then corresponding data changes in B machines are expressed as:

1. $OP_v^i \equiv \beta[t_v^i]$

2. $\beta[M'(p_j^i)]$.
$= \beta[M(p_j^i) - -E(p_j^i, t_v^i)\langle b\rangle + +E(t_v^i, p_j^i)\langle b_v^i\rangle]$
$= VP_j^i - -E_{p_j^i, t_v^i}\langle b_v^{i,\beta}\rangle + +E_{t_v^i, p_j^i}\langle b_v^{i,\beta}\rangle$
$= Ms\_Add(Ms\_Less(VP_j^i, E_{p_j^i, t_v^i}, Colset_l^i), E_{t_v^i, p_j^i}, Colset_l^i)\langle b_v^{i,\beta}\rangle$
$= VP_j^{\prime i}, \ where \ Colset_l^i = \beta[C(p_j^i)].$

3. $\beta[M] = \sum_{(i=1,j=1)}^{(m,n)} VP_j^i.$

It could also be written in another form:

4. $M(p_j^i) \xrightarrow{(t_v^i, b_v^i)} M'(p_j^i) \Leftrightarrow VP_j^i \xrightarrow{OP_v^i\langle b_v^{i,\beta}\rangle} VP_j^{i\prime}.$

5. $M \xrightarrow{(t_v^i, b_v^i)} M' \Leftrightarrow \sum_{(i=1,j=1)}^{(m,n)} VP \xrightarrow{OP_v^i\langle b_v^{i,\beta}\rangle} \sum_{(i=1,j=1)}^{(m,n)} VP'.$

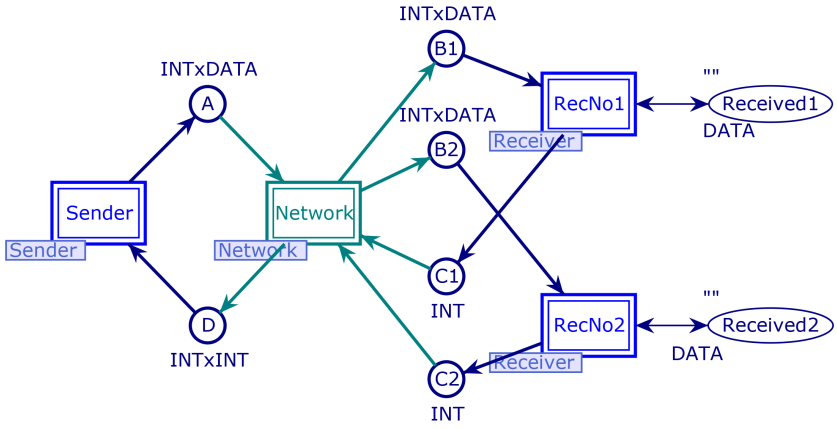
So far, enabling condition and token transit are matched for the transitions in the HCPNs and the operations in B machines. So the behaviours of the two systems are considered coherent.

4.4.4 Demonstration for mapping hierarchical coloured Petri net

This section presents a demonstration to illustrate the translation process of HCPN.

The example presented in Figs. 4.13 and 4.14 is a hierarchical protocol, similar to previous "Simple Protocol". It has separate pages (subnets) for the *Sender*, the *Network*, and the *Receiver* parts. The sender sends messages which the network broadcasts to the two receivers, while each receiver send acknowledgements back to the sender through the network.

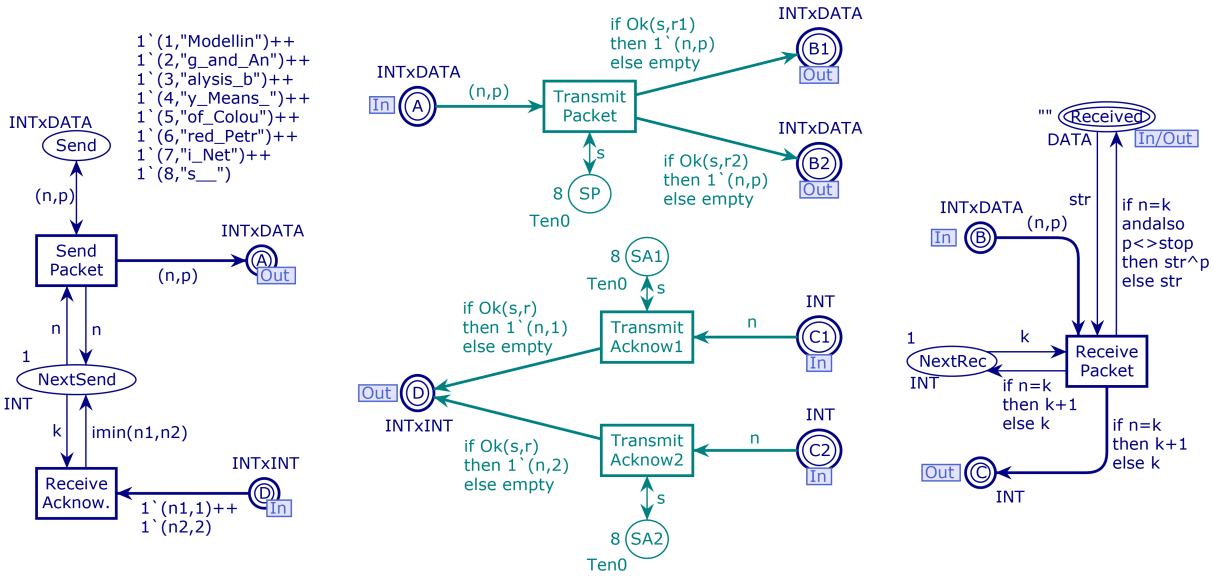
After mapping to B machines, in the root machine, all the data types are defined. There are 8 variables and 7 operations. The corresponding B machine is shown in Listing 4.9. The declaration part is similar to non-hierarchical CPNs. The original net does not contain normal transition in the root net, so the operations here are pre-defined for other transitions in the sub-nets. For example, the operation `OpArc_Sender_AA(ms)` will be recalled by operation `Op_SendPacket` in machine "Sender", the operation `OpArc_Nw_TrPck(ms_AA, ms_B1, ms_B2)` will be recalled by operation `Op_TransmitPacket` in machine "Network".



Overview

```
colset INT = int;                          colset Ten0 = int with 0..10;
colset DATA = string;                      colset Ten1 = int with 1..10;
colset INTxDATA = product INT*DATA;        var s: Ten0;
colset INTxINT = product INT*INT;          var r,r1,r2: Ten1;
var n,k,n1,n2: INT;                        fun OK(s:Ten0, r:Ten1)=(r<=s);
var p,str: DATA;
var stop: "s__";
```

Figure 4.13 – Root net of the hierarchical protocol model



(a) Sender net (b) Network net (c) Receiver net

Figure 4.14 – Sub-nets of the hierarchical protocol model

Listing 4.9 – *B*-machine for Root net

```
MACHINE HierarchicalProtocol
SETS STR = {Modellin, g_and_An, anlysis_b, y_Means, of_Colou, red_Petr, i_Net, s__}
CONSTANTS Ten0,Ten1,INTxDATA,INTxINT,val_stop
PROPERTIES
```

```
   INTxDATA = INTEGER*STR & INTxINT = INTEGER*INTEGER & val_stop : STR & val_stop = s__ &
   Ten0 <<: INT & Ten0 = 0..10 & Ten1 <<: INT & Ten1 = 1..10
VARIABLES AA, DD, B1, B2, C1, C2, Received1, Received2
INVARIANT
   AA : MS(INTxDATA) & DD : MS(INTxINT) & B1 : MS(INTxDATA) & B2 : MS(INTxDATA) &
   C1 : MS(INTEGER) & C2 : MS(INTEGER) & Received1 : MS(seq(STR)) & Received2 : MS(seq(STR)
       )
INITIALISATION
   AA := Ms_Empty(INTxDATA) || DD := Ms_Empty(INTxINT) || B1 := Ms_Empty(INTxDATA) ||
   B2 := Ms_Empty(INTxDATA) || C1 := Ms_Empty(INTEGER) || C2 := Ms_Empty(INTEGER) ||
   Received1 := Ms_Empty(seq(STR)) <+ {[]|->1} || Received2 := Ms_Empty(seq(STR)) <+
       {[]|->1}
DEFINITIONS
   "MultiSets.def"

OPERATIONS
   OpArc_Sender_AA(ms)= PRE ms : MS(INTxDATA) THEN AA:=Ms_Add(AA,ms,INTxDATA) END;

   OpArc_DD_Sender(ms)=
      PRE ms : MS(INTxINT) &
          Ms_Subset(DD,ms,INTxINT)
      THEN DD:=Ms_Less(DD,ms,INTxINT) END;

   OpArc_Nw_TrPck(ms_AA, ms_B1, ms_B2)=
   PRE ms_AA : MS(INTxDATA) & ms_B1 : MS(INTxDATA) & ms_B2 : MS(INTxDATA) &
       Ms_Subset(AA,ms_AA,INTxDATA)
   THEN AA:=Ms_Less(AA,ms_AA,INTxDATA) || B1:=Ms_Add(B1,ms_B1,INTxDATA) ||
        B2:=Ms_Add(B2,ms_B2,INTxDATA)
   END;

   OpArc_Nw_TrAck1(ms_DD, ms_C1)=
   PRE ms_DD : MS(INTxINT) & ms_C1 : MS(INTEGER) & Ms_Subset(C1,ms_C1,INTEGER)
   THEN DD:=Ms_Add(DD,ms_DD,INTxINT) || C1:=Ms_Less(C1,ms_C1,INTEGER)
   END;

   OpArc_Nw_TrAck2(ms_DD, ms_C2)=
   PRE ms_DD : MS(INTxINT) & ms_C2 : MS(INTEGER) & Ms_Subset(C2,ms_C2,INTEGER)
   THEN DD:=Ms_Add(DD,ms_DD,INTxINT) || C2:=Ms_Less(C2,ms_C2,INTEGER)
   END;

   OpArc_RecNo1(ms_B1, ms_C1, ms_Rec1_in, ms_Rec1_out)=
   PRE ms_B1 : MS(INTxDATA) & ms_C1 : MS(INTEGER) & ms_Rec1_out : MS(seq(STR)) &
       ms_Rec1_in : MS(seq(STR)) & Ms_Subset(B1,ms_B1,INTxDATA) &
       Ms_Subset(Received1,ms_Rec1_out,seq(STR))
   THEN B1:=Ms_Less(B1,ms_B1,INTxDATA) || C1:=Ms_Add(C1,ms_C1,INTEGER)||
        Received1:=Ms_Add(Ms_Less(Received1,ms_Rec1_out,seq(STR)),ms_Rec1_in,seq(STR))
   END;
```

```
  OpArc_RecNo2(ms_B2, ms_C2, ms_Rec2_in, ms_Rec2_out)=
  PRE ms_B2 : MS(INTxDATA) & ms_C2 : MS(INTEGER) & ms_Rec2_out : MS(seq(STR)) &
      ms_Rec2_in : MS(seq(STR)) & Ms_Subset(B2,ms_B2,INTxDATA) &
      Ms_Subset(Received2,ms_Rec2_out,seq(STR))
  THEN B2:=Ms_Less(B2,ms_B2,INTxDATA) || C2:=Ms_Add(C2,ms_C2,INTEGER)||
       Received2:=Ms_Add(Ms_Less(Received2,ms_Rec2_out,seq(STR)),ms_Rec2_in,seq(STR))
  END
END
```

The *B* machine of "Sender" is shown in Listing 4.10, there are 2 variables and 2 operations. The listing demonstrates how it recalls an operation in the root machine.

Listing 4.10 – *B*-machine for Sender net

```
MACHINE  Sender
INCLUDES HierarchicalProtocol
VARIABLES Send,NextSend
INVARIANT
  Send : MS(INTxDATA) & NextSend : MS(INTEGER)
INITIALISATION
  Send := Ms_Empty(INTxDATA) <+ {(1|->Modellin)|->1, (2|->g_and_An)|->1, (3|->anlysis_b)
      |->1, (4|->y_Means)|->1, (5|->of_Colou)|->1, (6|->red_Petr)|->1, (7|->i_Net)|->1,
      (8|->s__)|->1} ||
  NextSend := Ms_Empty(INTEGER) <+ {1|->1}
DEFINITIONS
  // SenderPacket
  var_SdPck == nn,pp;
    ae_rd_Sd_SdPck ==(Ms_Empty(INTxDATA) <+ {(nn|->pp)|->1});
    ae_rd_NS_SdPck ==(Ms_Empty(INTEGER) <+ {nn|->1});
    ae_SdPck_AA == (Ms_Empty(INTxDATA) <+ {(nn|->pp)|->1});
  Cdt_SdPck == (nn:INTEGER & pp:STR & Ms_Subset(Send,ae_rd_Sd_SdPck,INTxDATA) & Ms_Subset(
      NextSend,ae_rd_NS_SdPck,INTEGER));
  // ReceiveAcknow
  var_RcAck == kk,n1,n2;
    ae_NS_RcAck==(Ms_Empty(INTEGER) <+ {kk|->1});
    ae_RcAck_NS(nn)==(Ms_Empty(INTEGER) <+ {nn|->1});
    ae_DD_RcAck==(Ms_Empty(INTxINT) <+ {(n1|->1)|->1,(n2|->2)|->1});
  Cdt_RcAck == (kk:INTEGER & n1:INTEGER & n2:INTEGER & Ms_Subset(DD,ae_DD_RcAck,INTxINT) &
       Ms_Subset(NextSend,ae_NS_RcAck,INTEGER) )
OPERATIONS
  Op_SendPacket=
  SELECT #(var_SdPck).(Cdt_SdPck)
  THEN ANY var_SdPck
    WHERE Cdt_SdPck
    THEN OpArc_Sender_AA(ae_SdPck_AA)
    END
```

```
  END;

  Op_ReceiveAcknow=
  SELECT #(var_RcAck).(Cdt_RcAck)
  THEN ANY var_RcAck
    WHERE Cdt_RcAck
    THEN
      OpArc_DD_Sender(ae_DD_RcAck) ||
      IF n1>n2
      THEN NextSend:=Ms_Add(Ms_Less(NextSend,ae_NS_RcAck,INTEGER), ae_RcAck_NS(n2),
          INTEGER)
      ELSE NextSend:=Ms_Add(Ms_Less(NextSend,NextSend,INTEGER), ae_RcAck_NS(n1),INTEGER)
      END
    END
  END
END
```

The state of the each machine is shown in Table 4.11.

Table 4.11 – Component status for hierarchical protocol

AutoProved

	nPO	nPRi	nPRa	nUn	%Pr
Hierarchicalprotocol	19	0	19	0	100
Sender	6	0	6	0	100
Network	24	0	24	0	100
RecNo1	18	0	18	0	100
RecNo2	18	0	18	0	100

4.5 Extending transformation using prioritized transitions

In this section, the prioritized transitions and its mechanism are introduced into the model transformation framework.

4.5.1 Priority management

When simulating or analysing the CPN models, the priority enabling is manipulated by the tools. The prioritized enabling checking algorithm of CPN Tools is shown in Algorithm 3.1. From this algorithm we know that the priority enabling conditions for a give transition t depend on all the other transitions that have higher priority than t. That is to say, the prioritized transitions in CPN Tools have the right to check the enabling status of other transitions. However, in B language, operations have no right to access or to check the

other operations. As there is no such mechanism in B language, the priority management mechanism should be achieved inside the B machine itself. It can be divided into three parts:

Priority set

In the CPNs with prioritized transitions, all the priorities are pre-defined as a finite set of natural numbers $\mathbb{P}$. When enabling the transitions, there need a comparison of the priority values $\rho(t)$. However, in *B* machines, it is difficult to attach and get a natural number to an *operation*. So we leave the comparison function to the final priority management mechanism, in corresponding *B* machines, the definition of $\mathbb{P}$ can be translated into an enumeration set of priority IDs, which is used to distinguish different priorities.

Transition prioritized

As we may not attach a number to operations in *B* machines, we consider it as a necessary enabling condition of operations. So, we need to add a global variable to indicate the current system priority level (Pri). It value is always modified by the priority management mechanism. An operation can be priority-enabled only if Pri equals to some give value. Let a transition t_j have its priority $\rho(t_j) = pri_j$. The example of its corresponding operation is shown in Fig. 4.15.

OP_i = PRE Pri= pri_j
THEN SELECT CDT_{t_1} THEN ACT_{t_1} END
END

Figure 4.15 – An example of prioritized operation

Priority management

In order to achieve the similar priority management mechanism with AMN, we introduce an operation called "Chg_Priority". Its algorithm is shown in Algorithm 4.1. It checks the operations that are sorted by groups of different priorities, and processes them highest-priority-first until it finds the first pre-enabled operation Op_j, and it assigns the variable Pri to $\rho(Op_j)$. Then all the operations that have the same priority of $\rho(Op_j)$, can be priority-enabled by the variable Pri.

Algorithm 4.1 Algorithm of operation "Chg_Priority"

```
1: PRIORITY← {pri_1,···,pri_s}, where pri_1 > ··· > pri_s
2: procedure CHG_PRIORITY(Pri) is
3:     if Pri = Suspend then
4:         if pri_1 ∈ PRIORITY then
5:             for all Op that ρ(Op) = pri_1 do
6:                 if CheckEnabling(Op) then
7:                     return Pri = pri_1
8:             ......
9:         else if pri_s ∈ PRIORITY then
10:            for all Op that ρ(Op) = pri_s do
11:                if CheckEnabling(t) then
12:                    return Pri = pri_s
13:        elsereturn Pri = Stop
```

The content of the operation *Ctrl_Priority* is shown in Fig. 4.16. The role of this operation is a priority management system. It is only enabled when $Pri = Suspend$. Once executed, it will check all the pre-enabling conditions according to the level of priority. First it will check all the conditions for transitions with highest priority. If one of the conditions is pre-enabled, it will set Pri to the highest priority. Otherwise, it will continue to check the rest transitions with lower priority, until it finds a pre-enabled condition. If none of the transitions is pre-enabled, it will set Pri to "Stop", which means this is the dead-end of the whole system.

```
Chg_Priority = PRE Pri= Suspend
  THEN IF #VAR_1.(CDT_1) or ··· THEN Pri:=pri_t1
       ELSIF #VAR_2.(CDT_2) or ··· THEN Pri:=pri_t2
       ···
       ELSIF #VAR_n.(CDT_n) or ··· THEN Pri:=pri_tn
       ELSE Pri:=Stop
    END
  END
```

Figure 4.16 – The template of operation "Chg_Priority"

4.5.2 Framework of model transformation

For a CPN with prioritized transitions (CPN, ρ), we define that:

MACHINE M
SETS $Colset_1, Colset_2, \cdots, Colset_{k-1}, PRIORITY$
CONSTANTS $Colset_k, Colset_{k+1}, \cdots, Colset_l$
PROPERTIES
$Colset_k := EXPR_V^k \ \& \ Colset_{k+1} := EXPR_V^{k+1} \ \& \cdots \& \ Colset_l := EXPR_V^l$
VARIABLES $VP_1, VP_2, \cdots, VP_m, Pri$
INVARIANT
$VP_1 : (Colset_1)_{\mathrm{MS}} \ \& \ VP_2 : (Colset_2)_{\mathrm{MS}} \ \& \cdots \& \ VP_m : (Colset_l)_{\mathrm{MS}} \ \& \ Pri : PRIORITY$
INITIALISATION
$VP_1 := VI_1 \parallel VP_2 := VI_2 \parallel \cdots \parallel VP_m := VI_m \parallel Pri := Suspend$
DEFINITIONS
"MultiSets.def" ;
…
$\mathrm{VAR}_{t_j} == EXPR_V^{t_j}$; // For trainsition t_j
$\mathrm{E}_{p_1,t_j} == EXPR_{\mathrm{VAR}_{t_j}}$;
$\mathrm{E}_{p_2,t_j} == EXPR_{\mathrm{VAR}_{t_j}}$;
…
$\mathrm{E}_{t_j,p_1} == EXPR_{\mathrm{VAR}_{t_j}}$;
$\mathrm{E}_{t_j,p_2} == EXPR_{\mathrm{VAR}_{t_j}}$;
…
$\mathrm{GRD}_{t_j} == EXPR_{\mathrm{VAR}_{t_j}}$;
$\mathrm{CDT}_{t_j} == EXPR_{\mathrm{VAR}_{t_j}}$;
…
OPERATIONS
$Chg\_Priority$= PRE $Pri = Suspend$ THEN Pri_Comp END;

OP_{t_1}= PRE $Pri = pri_{t_1}$
THEN SELECT CDT_{t_1} THEN ACT_{t_1} END END;
…
OP_{t_n}= PRE $Pri = pri_{t_n}$
THEN SELECT CDT_{t_n} THEN ACT_{t_n} END END;
END

Figure 4.17 – The framework of a *B* machine transformed from prioritized Petri net

(i) $CPN = (P, T, A, \Sigma, V, C, G, E, I)$,

(ii) $\mathbb{P} = \{pri_1, ..., pri_s\}$,

(iii) $P = \{p_1, ..., p_m\}$,

(iv) $T = \{t_1, ..., t_n\}$,

(v) $A = \{(p,t) \mid p \in P \wedge t \in T\} \cup \{(t,p) \mid t \in T \wedge p \in P\}$,

(vi) $\Sigma = \{\sigma_1, ..., \sigma_{k-1}, \sigma_k, ..., \sigma_l\}$

Let the new β be the mapping $\beta : (CPN, \rho) \rightarrow AMN$, then the image of CPN (CPN, ρ) under β is a B machine Bm, $Bm = \beta[(CPN, \rho)]$ with the structure in Fig. 4.17. All the new added elements of Bm have the following mapping relationship:

The priority related relations are:

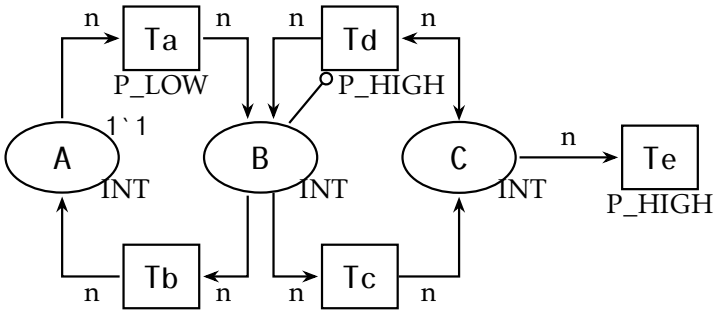
1. $PRIORITY = \beta[\mathbb{P}]$ is a set of priorities, and $Suspend, Stop \in PRIORITY$.
2. Pri is a variable that indicates the current priority level of the system.

For each transition related elements $\forall j \in \{1..n\}$, the relations are shown below.

3. pri_{t_j} is the priority of the transition j.
4. $ACT_{t_j} = act(p_1, t_j) \wedge act(p_2, t_j) \wedge ... \wedge act(p_m, t_j) \wedge (\textbf{Pri} := \textbf{Suspend})$.

4.5.3 Mapping coloured Petri net with prioritized transitions

For a better understanding of the new transformation method, we have a toy example in Fig. 4.18, which is a CPN model with 3 places (A-C) and 5 transitions (Ta-Te). Places contain *tokens* (a multi-set) which represent the state of the system. Then initial marking is that place *A* has a token 1`1 with type INT. In such an initial marking, transition *Ta* is pre-enabled in the binding *n=1* as *A* contains a single token with value "1". In this CPN model, we have assigned priorities to transitions *Ta*, *Td*, *Te*, and other transitions without a priority inscription have the default priority. There are 3 types of priorities P_LOW, P_NORMAL and



Declarations:
colset INT = int;
var n:INT;
var P_HIGH = 100;
var P_NORMAL = 1000;
var P_LOW = 10000;

Figure 4.18 – A simple coloured Petri net model with prioritized transitions

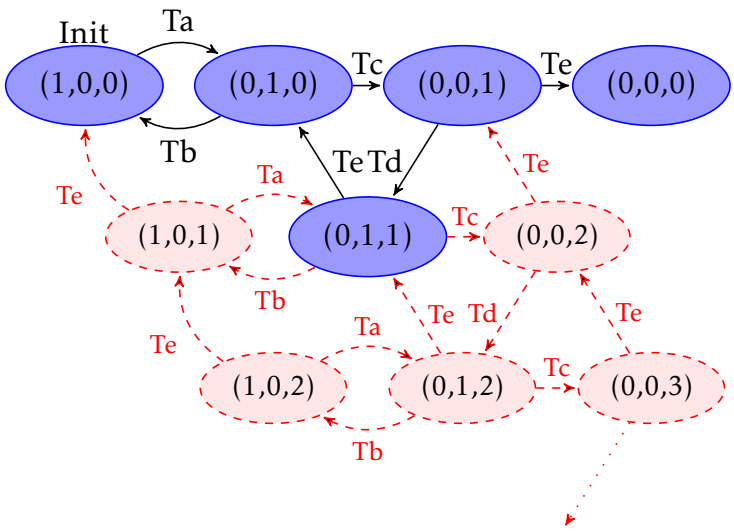


Figure 4.19 – The state-space discussion of example in Fig. 4.18

P_HIGH. The default priority is P_NORMAL. Corresponding to Definition 3.1, each priority is a natural number that is defined in the declaration table in Fig. 4.18. In CPN Tools, the priorities are defined as Positive INT variables, the higher the number, the lower the priority. According to the model declarations, priority relation is P_LOW < P_NORMAL < P_HIGH. Besides, the transition *Tb* has an *Inhibitor* arc from place *B*, which prevents *Tb* from enabling if place *B* contains a token.

The state-space of the example CPN is shown in Fig. 4.19. It is a simple net with single token colour and without value change, so we may use the representation of classical PN to illustrate its state-space. For example, the initial state $(1,0,0)$ means place *A* contains the token "1`1", place *B* and place *C* have no token. The name on the arcs are the fired transitions. The solid lines indicate the state-space of the system with priorities, while the dashed part is the system without priority. It is clear that prioritiezd transitions can exclude some unwanted occurrences, which could effectively reduce the size of state-space.

The corresponding *B* machine is shown in Listing 4.11. In this machine, we named the variables and the operations as when they are in the places and transitions. Expressions starting with "e_" are the arc expressions. The source and destination of each arc are noted within its name. For example, the arc e_AA_Ta means the arc from place *A* to transition *Ta*. Similarly, the arc e_Ta_BB means the arc from transition *Ta* to place *B*. The expression starting with "CDT_" is the enabling condition of each operation.

Listing 4.11 – *B*-machine for the prioritized coloured Petri net model

```
MACHINE  Prioritized_Transitions
SETS PRIORITY={High, Normal, Low, Suspend}
VARIABLES AA, BB, CC, Pri
INVARIANT
  AA : MS(INT) & BB : MS(INT) & CC : MS(INT) &
```

```
  Pri : PRIORITY
INITIALISATION
  AA := Ms_Empty(INT) <+ {1|->1} || BB := Ms_Empty(INT) || CC := Ms_Empty(INT) ||
  Pri := Suspend
DEFINITIONS
  "MultiSets.def"

  var_Ta == nn;
    e_AA_Ta ==(Ms_Empty(INT) <+ {nn|->1});
    e_Ta_BB ==(Ms_Empty(INT) <+ {nn|->1});
  cdt_Ta == (nn:INT & Ms_Subset(AA, e_AA_Ta, INT));

  var_Tb == nn;
    e_BB_Tb ==(Ms_Empty(INT) <+ {nn|->1});
    e_Tb_AA ==(Ms_Empty(INT) <+ {nn|->1});
  cdt_Tb == (nn:INT & Ms_Subset(BB, e_BB_Tb, INT));

  var_Tc == nn;
    e_BB_Tc ==(Ms_Empty(INT) <+ {nn|->1});
    e_Tc_CC ==(Ms_Empty(INT) <+ {nn|->1});
  cdt_Tc == (nn:INT & Ms_Subset(BB, e_BB_Tb, INT));

  var_Td == nn;
    e_rd_CC_Td ==(Ms_Empty(INT) <+ {nn|->1});
    e_Td_BB ==(Ms_Empty(INT) <+ {(nn)|->1});
  cdt_Td == (nn:INT & Ms_Subset(CC, e_rd_CC_Td, INT) & BB = Ms_Empty(INT));

  var_Te == nn;
   e_CC_Te ==(Ms_Empty(INT) <+ {(nn)|->1});
  cdt_Te == (nn:INT & Ms_Subset(CC, e_CC_Te, INT))

OPERATIONS
  Chg_Priority = PRE Pri=Suspend
  THEN IF #var_Td.(cdt_Td) or #var_Te.(cdt_Te) THEN Pri:=High
       ELSIF #var_Tb.(cdt_Tb) or #var_Tc.(cdt_Tc) THEN Pri:=Normal
       ELSIF #var_Ta.(cdt_Ta) THEN Pri:=Low END END;

  Op_Ta= PRE Pri=Low
  THEN SELECT #var_Ta.(cdt_Ta)
    THEN ANY var_Ta WHERE cdt_Ta
      THEN AA:=Ms_Less(AA, e_AA_Ta, INTEGER) || BB:=Ms_Add(BB, e_Ta_BB, INTEGER) ||
        Pri:=Suspend END END END;

  Op_Tb= PRE Pri=Normal
  THEN SELECT #var_Tb.(cdt_Tb)
    THEN ANY var_Tb WHERE cdt_Tb
      THEN BB:=Ms_Less(BB, e_BB_Tb, INTEGER) || AA:=Ms_Add(AA, e_Tb_AA, INTEGER) ||
```

```
        Pri:=Suspend END END END;

  Op_Tc= PRE Pri=Normal
  THEN SELECT #var_Tc.(cdt_Tc)
    THEN ANY var_Tc WHERE cdt_Tc
      THEN BB:=Ms_Less(BB, e_BB_Tc, INTEGER) || CC:=Ms_Add(CC, e_Tc_CC, INTEGER) ||
        Pri:=Suspend END END END;

  Op_Td= PRE Pri=High
  THEN SELECT #var_Td.(cdt_Td) THEN ANY var_Td WHERE cdt_Td
     THEN BB:=Ms_Add(BB, e_Td_BB, INTEGER) || Pri:=Suspend END END END;

  Op_Te= PRE Pri=High
  THEN SELECT #var_Te.(cdt_Te) THEN ANY var_Te WHERE cdt_Te
     THEN CC:=Ms_Less(CC, e_CC_Te, INTEGER) || Pri:=Suspend END END END

END
```

The component status for this machine is shown in Table 4.12. There are 10 proof obligations, and are automatically proved (PRa) by Atelier-B Ver.4.1.0.

Table 4.12 – Component status for the corresponding *B* machine

AutoProved

	nPO	nPRi	nPRa	nUn	%Pr
Initialisation	2	0	2	0	100
Pri_Ctrl	0	0	0	0	100
Op_Ta	2	0	2	0	100
Op_Tb	2	0	2	0	100
Op_Tc	2	0	2	0	100
Op_Td	1	0	1	0	100
Op_Te	1	0	1	0	100
Prioritized_Transitions	**10**	**0**	**10**	**0**	**100**

4.5.4 Transformation equivalence

In the previous two subsections, we have presented the formal transformation methodology and a case study. With the proposed mapping rules, it is necessary to prove that the introduced transformation is equivalent.

The proof method of system state, enabling condition and token transit are the same with those in Section 4.3.4. So in this subsection, we only proof the priority enabling condition.

Lemma 4.6. Priority enabling. For a transition t with priority $\rho(t)$ is pre-enabled by marking M, there are the following concepts:

1. $[t]_{\mathrm{EN}}$ is the pre-enabled status of t.
2. Pri is the current highest enabled priority.
3. $T_{\mathrm{EN}}(M)$ and $T_{\mathrm{UN}}(M)$ are the sets of transitions that pre-enabled or disabled by the marking M.
4. $T^{>}(t)$ and $T^{<}(t)$ are the sets of transitions that have the higher or lower priority than t.

If this transition t is priority enabled, then the corresponding condition of Definition 3.2 is expressed as:

5. $PRIORITY = \beta[\mathbb{P}]$
6. $\beta[M \xrightarrow{[t]_{\mathrm{EN}}}] = \beta[\forall t' \in T_{\mathrm{EN}}(M),\ \rho(t) \geq \rho(t')]$
   $= \beta[T^{>}(t) \subseteq T_{\mathrm{UN}}(M)] = Pri = \rho(t)$
   $= Chg\_Priority\langle t \rangle$

Although in the B machines, a different priority control algorithm is added, but it reproduces the same mechanism as that in CPNs. Thus, the consistency of priority enabling conditions is guaranteed.

Model checking

Model checking is a classical method to check equivalence of *finite-state* systems. For a given model of the system, exhaustively and automatically checking is applied to verify whether this model meets certain specifications. As both the CPNs and the B method have their own supported tools for simulation and state-space analysis. Because the case study in this paper is a small one, we can perform a comparison of their state-space. This comparison is not a proof, but it can be a good illustration for better understanding.

The state-space of the CPN model is shown in Fig. 4.19. The state-space of the B machine is calculated by ProB[1] and shown in Fig. 4.20. After initialisation, the CPN model has 5 states, while the B machine has 10 states. Each state in Fig. 4.20 represents the tokens of each place and the value of Pri. As the priority management in B machine is achieved by an operation "Chg_Priority", before any data value changes, this operation must be recalled in order to determine the highest priority of all the pre-enabled operations. So the new states caused by the operation "Chg_Priority" can be seen as transient states. From the figure, a pair of states that directly linked by an arc "Chg" have the same marking, not considering priority. The pair of states can be approximated as a single node, which is indicated by dashed ellipses. Then the new state-space is exactly the same as that of the CPN model.

[1] ProB, developed by the University of Southampton, is an animator and model checker for the B-Method. It allows animation of many B specifications, and can be used to systematically check a specification for a range of errors

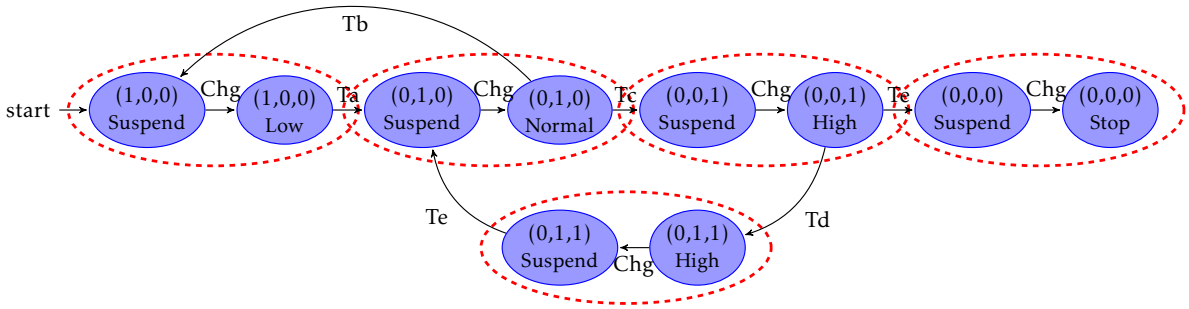


Figure 4.20 – The state-space of *B* machine "Prioritized_Transitions"

4.6 Conclusion

This chapter discusses the combination usages of different modelling languages in the context of safety critical systems. The strong motivation of such an approach is to bridge the gap of critical tasks, from a strong requirement analysis towards a valid implementation on a real system. Precisely, this approach focuses on model transformation, from CPN to *B* machine while preserving the same modelling requirements. When systems are already specified by CPNs and verified by experts, to save efforts and reduce errors, our method could transform CPN models into *B* machines. This approach may assist people in the design phase, to quickly shift from a validated solution towards a valid implementation.

As the CPNs have a fine modelling and analysis characteristic, especially an easy-to-understand graphic notation, it is more attractive for designers to use than text-based methods. However, an abstract *B* machine is a valid input of the *B* method which is able to develop into the executable implementation.

First, the detailed mapping process of non-hierarchical CPNs and AMNs are presented. Then, the transformation method is illustrated with some classical examples. This translation is compatible with various types of colour sets and the transformed *B* machines can be automatically proved by Atelier B. Based on that, hierarchy concept is introduced into the previous transformation method to carry out the mapping from HCPNs to AMNs. It could be applied to the "multi-systems" models and is illustrated by a hierarchical protocol example. Finally, we consider the transition priorities of CPNs, and create a priority management mechanism to realize the prioritized operations in abstract *B* machines.

One of the limitations of this paper is that the colour-set "list" is not so perfect. Because it is already a concrete programming data type in CPN, it is difficult to abstract all its function with set theory and first order logic. Another limitation is that the current translation method does not support timed CPNs, which are associated with a timed multi-set and new behaviour rules. Considering the timetable and time constrained protocol, it could be an interesting research task. Note that the *B* machine dealing with the time concept still needs some scientific work.

Part III

Conclusions

Conclusions and Perspectives

Conclusions

This thesis has been devoted to the model based system engineering for safety of railway critical systems. It is an interdisciplinary area of modelling of complex systems, safety-critical systems, formal methods and model-driven engineering. This thesis provides some new approaches via formal languages, which aim to aid designers in effectively ensuring railway safety, and improving the quality with system design and verification in railway industry. This study has focused on two aspects that are important for analysing the French RIS: formal specifications and a fast shift from analysis to implementation. 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 behaviours of the RISs. To bridge the gap between the specifications and the implementations, *B* method has been chosen as the solution of formal implementation development. A transformation from CPN to *B* machines is proposed, as it is necessary. It can be used for the formal implementations or the theorem-proving.

The work has been presented as follows:

Formal modelization of RIS

Railway system is always in a strong domain specific context. 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 colour features make it possible to propose a generic and compact structure which 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, which could formally specify the constructions of the fixed installations and the signalling operations of the interlocking logic. As a multi-input multi-output system, the signalling 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, which 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 model

Model transformation

The purpose of using a model transformation is to save efforts and reduce errors by automatically building the models that conform to different modelling languages. In the French railway industry, the Petri nets and the *B* method are two recognized formal methods for safety critical systems, having their own successful applications. The PNs are a mathematical modelling language for describing the distributed systems, and they offer superior graphical notations for stepwise processes. They are accepted by the French railway specialists, because they have user-friendly notations. Besides, there are already some system that are specified by PNs and CPNs. The *B* method is a software development method based on abstract machine notations and the concept of refinement. Because there are already some successful implementations of *B* method, in the French railway context, *B* proved model is accepted as a strong safety proof. Consequently, there is a need for transforming the CPN models into abstract *B* machines. This transformation can help people to quickly shift from a valid design solution to a valid implementation development process. The transformed *B*

machines can also be used for theorem-proving.

First, for non-hierarchical CPN model, we introduce a systematic mapping framework of model transformation in which all the information on a CPN can find its corresponding portion (functions or definitions) in the transformed *B* machine. The "multi-set" is the core mechanism of CPN language. We improve and extend the solution in [Bon and Dutilleul 2013], so that the transformation is compatible with various types of colour sets and the transformed B machines can be automatically proved by Atelier B without using "interactive prover". The multi-set mechanism and its related specifications are defined in an external **.def* file so that they can be accessed by all the transformed *B* machines. CPN's system marking, colour-sets, tokens are represented static part of *B* machines, while, the transitions, step enabling, token transit are represented by operations of *B* machines This transformation is compatible with various types of colour sets and the transformed B machines can be automatically proved by Atelier B. Sometimes, a CPN model specifies a scenario of multi-systems (or components). Each sub-system is an independent one, which could be treated as a signal system and be transformed into a *B* machine. Based on that, the hierarchy concept is introduced into the previous transformation method to carry out the mapping from HCPNs to AMNs. The hierarchical links between connected machines are through the "INCLUDES" clause and the accessible pre-defined operations in the upper level of machines. In addition, we introduce a set of rules for classifying different places into different machines. Finally, we consider the transition priorities of CPNs, and create a priority management mechanism to realize the prioritized operations in abstract *B* machines. Because, in CPN models, the priority enabling is manipulated by CPN tools, while in the *B* machine there are no such functions. All the priority mechanisms must be achieved inside *B* machines. So we add some new elements into the transformation framework, including a variable, a set and a priorities management operation. By the priority-based pre-condition of operations, we can realize the similar functions of prioritized transition.

Limits of model transformation

Colour set. Not all colour types are fully supported in our solutions. For example, the colour-set "list" is not so perfect. Because it is already a concrete programming data type in CPNs, it is difficult to abstract all its functions with set theory and first order logic.

Meta language inscriptions. The CPN model adopts a programming language, which is based on the functional programming language (Standard ML). This language has the advantages of defining colour types, describing data manipulation, and creating compact and parametrized models. However, most of the time it is impossible to "reverse" a well developed model of programming language into a model based on set theory and first-order logic.

Time domain. In our study, time factor has not yet been considered. So, the timed CPNs are not supported,which is associated with a timed multi-set and new behaviour rules. Considering the timetable and time constrained protocol, it could be an interesting research task. Note that the B machine dealing with the time concept still needs some scientific work.

Perspectives

Based on the results given by this thesis, several perspectives should be considered:

In theory

- **Extend the model framework**: The current solution is only a basic extendible framework. There are many factors that are not considered in this framework. Further study may perform the following works: introducing failure modes, timing aspects, simulation of given scenarios.

- **Combined analysis with ERTMS**: Combination analysis and global safety verification can be performed by integrating several existing models. The ERTMS is a general standard, while each RIS is compliant with different infrastructure conditions. With some existing ERTMS rule models [Hörste and Schnieder 1999], we can check the safety consistency of ERTMS rules and given local conditions. There is an example of combination research on ERTMS in [Goverde, Corman, and D'Ariano 2013], which use standard UIC compression method for the scheduling conditions, and compute the capacity consumption via Monte Carlo simulation.

- **Extend the transformation framework**: For HCPNs, we only consider the "multi-systems" solution in this thesis. The other solution, so called "single-system", corresponding to a large single system with multiple nets. In order to continuously enhance the compatibility of our transformation, this feature should be considered as necessary.

In application

- **PERFECT project**: This thesis takes place in the PERFECT project, which studies safe implementation of ERTMS in France. The management of railway signalling is based on local rules pertaining to each country. Generally, the current solution of ERTMS implementation is to use ERTMS-equipped trains on existing lines. This makes it difficult to evaluate the ERTMS-based system in term of safety.

  The first step is to build formal models of each sub-systems, then to carry out models analysis to formally prove each sub-system. Some research of first step of the project can be found in papers, such as the B formal validation of ERTMS operating rules

[Ben-ayed, Collart-Dutilleul, Bon, Idani, et al. 2014; Ben-ayed, Collart-Dutilleul, Bon, Ledru, et al. 2014] and the modelling of French interlocking system (in my thesis).

The second step is to integrate each model into one formal model. It will then be possible to analyse in details the influence of implementing European laws on the original systems. To prepare the model combination, we need to unify all the specifications, so the model transformation technique is studied, such as UML to B [Ben-ayed, Bon, and Collart-Dutilleul 2014] and CPN to B (in my thesis).

The integration work is an interesting perspective of both the work of Ben-Ayed and SUN. In the future, the integration work will be performed, some scenarios and simulations may be tested on this global model.

- **Semi-automated tool of model transformation**: The model transformation from CPNs to AMNs seems to be tedious laborious. It is inefficient and error-prone, not easy to check. To maintain high efficiency and low error rate, we need to develop a software to assist the transformation process, in order to minimize the human participation rate.

Résumé étendu en français

Le développement et l'application des langages formels sont un défi à long terme pour la science informatique. Un enjeu particulier est l'acceptation par l'industrie. Malgré des succès industriels avérés, la dissémination des méthodes formelles dans l'industrie ferroviaire est encore limitée. Cette thèse présente une approche pour la modélisation et la vérification des postes d'aiguillage français en utilise les méthodes formelles pour assurer efficacement la sécurité ferroviaire. Les méthodes formelles étudiées dans cette thèse sont les réseaux de Petri colorés (RdPCs) et la méthode *B*. Les RdPCs sont appropriés pour spécifier les systèmes de contrôles/commandes à événements discrets, alors que la méthode *B* offre la démonstration automatique de théorèmes et permet d'refiner les modèles de specification abstraite jusqu'à l'implémentation finale.

Motivation de la recherche

Jusqu'à présent, il existe plus de 20 types de systèmes de signalisation et de contrôle de vitesse différents dans toute l'Union européenne. Chaque système est autonome et non-interopérable avec les autres de sorte que les trains qui circulent dans plusieurs pays doivent être équipés de tous les systèmes à bord. Cela demande un effort d'ingénierie et d'intégration supplémentaire, augmente les coûts totaux et réduit la performance du transport ferroviaire. Pour mettre fin à une telle situation de segmentation et pour promouvoir le marché ferroviaire européen des passagers et du fret, le système de signalisation ERTMS (European Rail Traffic Management System) a été introduit pour créer une nouvelle génération de systèmes de signalisation et de systèmes de contrôle de la vitesse afin de disposer d'un réseau sans soudure au niveau européen.

En raison des contraintes financières et organisationnelles, il est impossible d'installer ERTMS sur l'ensemble du réseau dans un courte période. La plupart des compagnies nationales préfèrent déployer progressivement ERTMS. Pour les lignes existantes, il existe une solution de « fonctionnement mixte » qui est une stratégie dans lequel la signalisation en bordure de voie est équipé à la fois d'ERTMS et des systèmes conventionnels. Jusqu'ici, il n'y a pas eu beaucoup d'implementation d'ERTMS ce qui signifie qu'il est impossible d'évaluer ce nouveau système de manière traditionnelle. Au cours de la mise en œuvre d'ERTMS, une évaluation de la cohérence globale est nécessaire. La spécification et l'analyse de systèmes

d'enclenchement ferroviaires (SEF) en est un élément important. Afin de maintenir la sécurité de haut niveau avec le cadre déterministe, un projet nommé « PERFECT » a été lancé par l'IFSTTAR pour développer la spécification et la vérification de la sécurité des SEFs français dans le cadre des règles nationales et de l'influence de la mise en œuvre d'ERTMS sur les systèmes nationaux.

Mon travail de thèse est une des phases fondamentales de ce projet. Il se concentre sur l'approche formelle des SEFs français basé sur le poste à relais à commande informatique (PRCI). L'objectif général de cette thèse est d'étudier l'hypothèse que les languages formelles peuvent améliorer la qualité de la conception et la vérification des systèmes dans l'industrie ferroviaire. L'étude vise à fournir une méthodologie pour une évaluation complète de la cohérence des deux aspects suivants : les règles de fonctionnement des systèmes de signalisation locales et les exigences de sécurité supplémentaires (ERTMS).

Contexte de la recherche

Si nous voulons obtenir l'acceptation et la reconnaissance dans l'industrie ferroviaire française, il nous semble préférable de « parler », dès la phase de conception, le même langage qu'eux ou au moins un langage similaire.

Langage formel de modélisation

Le GRAphe Fonctionnel de Commande Étape/Transition (GRAFCET) est un mode de représentation et d'analyse d'un automatisme particulièrement bien adapté aux systèmes à évolution séquentielle. Le GRAFCET est largement utilisé dans les domaines de l'industrie et de la recherche notamment dans l'industrie française. Par exemple, des grandes entreprises comme Alstom, Renault, Siemens, Peugeot et Michelin l'utilisent largement ou le reconnaissent comme une norme interne. Cependant, avec la nécessité accrue de sécurité imposée par les normes internationales, le GRAFCET a été critiqué en raison de son absence de fondements formels permettant d'assurer les exigences d'exactitude et de sécurité.

Afin d'être proche des habitudes d'utilisation de l'industrie et de profiter des méthodes formelles, nous considérons que les réseaux de Petri (RdPs) sont un outil satisfaisant pour la modélisation du système ferroviaire français. Les formalismes de notation des RdPs et du GRAFCET sont si proches que l'ingénieur qui est familier avec le GRAFCET comprendra aisément les modèles RdP. Autrement dit, si un système est spécifié par des RdPs, il peut être validé à la fois par des outils des RdPs et par les ingénieurs experts. En conséquence, les RdPs sont le langage formel qui nous semble le plus appropriée pour poursuivre nos recherches. Actuellement, les RdPs et les RdPCs sont acceptés par certaines industries françaises telles que la Société Nationale des Chemins de Fer français (SNCF) [Antoni 2012b; Buchheit et al. 2011; Lalouette et al. 2010].

Nous pouvons également citer une application à grande échelle, le métro d'Oslo [Bjørk 2006; Hagalisletto et al. 2007; Moen and Yu 2004; Yu 2004], qui intègre les RdPCs dans le développement de son système de simulation et les outils pour l'analyse des horaires de trains. Un des retours d'expérience important est que les RdPCs sont un bon langage de spécification pour la communication. En effet, le groupe de recherche a collaboré avec les ingénieurs en chef de l'infrastructure des chemin en fer et le département de la circulation par ce moyen et bien qu'ils n'étaient pas des spécialistes en RdP, ils ont compris les modèles et ont apporté des amélioration significatives.

Langage formel de démonstration de théorèmes

La vérification de modèles RdPC est basées sur la vérification de modèle qui se confronte au problème bien connu de « l'explosion de conbinatoire ». Afin d'être capable de vérifier un système à grande échelle, nous avons besoin d'utiliser un autre langage formel qui permet la démonstration de théorèmes.

Dans le context de l'industrie ferroviaire française, nous pensons bien évidemment à la méthode *B*. Non seulement la méthode *B* permet de modéliser formellement un système mais elle permet également de prouver que tous les comportements du système respectent les propriétés. Des projets industriels tels que le système SACEM[2], le SAET-METEOR[3] et le CdG-VAL[4] utilisent le langage *B* dans leur processus de développement. Ces succès d'ingénierie ont réussi à convaincre de la fiabilité de la méthode *B*, parce que le code final généré à partir de la specification en machine *B* est considéré comme sûr et prouvé. Donc, dans le contexte ferroviaire français, un modèle *B* prouvée est acceptée comme une preuve forte de la sécurité [Boulanger 2013a,b].

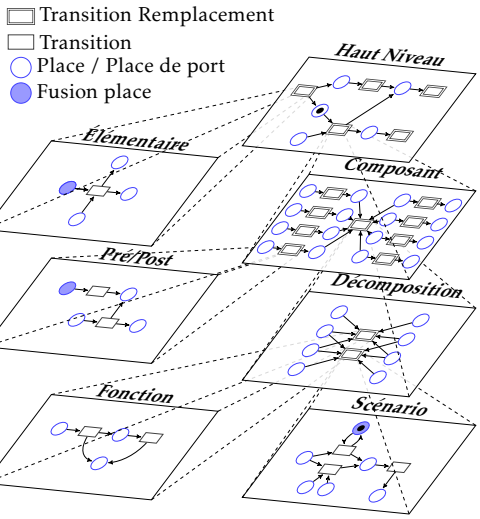
Modélisation formelle pour les systèmes d'enclenchement ferroviaires via les réseaux de Petri colorés

Le cadre de modélisation d'un SEF peut être divisée en trois parties : les opérations de signalisation, les installations fixes et les trains. les opérations de signalisation sont un ensemble de règles de fonctionnement et de procédures de contrôle d'un système d'enclenchement. Les installations fixes comprennent des cantons, des aiguillages, des feux de signalisation et d'autres installations automatiques. Les trains se déplacent sur les itinéraires d'enclenchement et sont supervisés par les conditions de la route et les instructions d'opérations.

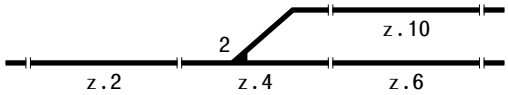
Les opérations de signalisation du SEF sont un système à entrées et sorties multiples. Leurs modes de fonctionnement nécessitent des fonctions de différents niveaux. Par conse-

[2]Système d'Aide à la Conduite, l'Exploitation et la Maintenance
[3]Système d'Automatisation de l'Exploitation des Trains - METro Est-Ouest Rapide
[4]Véhicule Automatique Léger de l'aéroport Charles de Gaulle

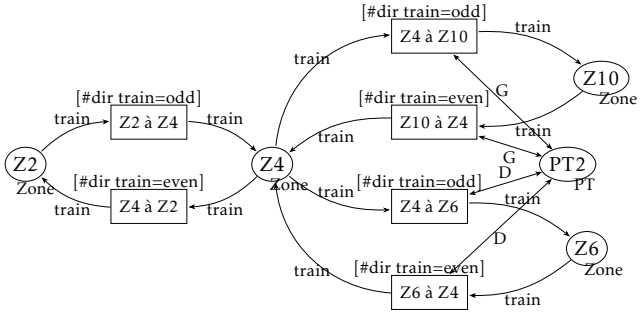
quent, pour la modélisation, la structure hiérarchique nous semble la plus compatible avec ces exigences. Le modèle doit intégrer les différentes fonctions de la description du système et contenir des vues de modularité comme représenté sur la figure 5.1(a). Les RdPCHs nous semble répondre à ces exigences et permettent la mise en place du modèle simple.



(a) La structure du modèle hiérarchique de fonctionnement de signalisation



(b) Démo des Cantons avec un aiguillage



(c) Modèle RdPC correspondante

Figure 1 – Une approche géographique de la modélisation de SEF

La solution la plus courante de la modélisation des installations fixes est l'approche géographique. Cette approche peut être considérée comme la modélisation du placement géographique des éléments physiques du SEF. Sa structure géographique nous permet de découper le réseau ferroviaire en composants indépendants et distribués qui peuvent être modélisées individuellement.Normalement, un itinéraire du SEF est constitué de plusieurs composants similaires : des cantons, des aiguillages et des feux de signalisation en bordure de itinéraire. Les figures 5.1(b) et5.1(c), montre un schéma d'aiguillage et et son modèle de RdPC correspondant. Chaque place représente un canton et chaque transition modélise le déplacement d'un train entre deux cantons. L'aiguillage est représenté par une place unique qui stocke les informations de connexion en cours.

La modélisation géographique impose un modèle par gare. Pour modéliser une ligne complète, il est donc nécessaire de juxtaposer les différents models bout à bout. Ces contraintes multiplient les risques d'erreur. Nous avons constaté que les gares qui sont équipés avec le même type de SEF suivent les mêmes règles nationales. Les seules différences sont les configurations de leurs installations fixes. Par consequent, il nous semble que la structure attendue du modèle peut être à la fois générique et paramétrée permettant ainsi de modéliser, à partir du même modèle, les diverses configurations possibles. Pour pouvoir utiliser une structure générique, on doit donc s'affranchir de l'approche géographique de la modélisation. Les informations géographiques doivent alors être représenté dans les jetons. La figure 2 illustre cette généricité. Ce réseau simple modélise le schema de la figure 5.1(b)

par un RdPC paramétré et est equivalent au réseau de la figure5.1(c). La place « Connexions des cantons » modélise les contraintes de circulation des trains. À chaque franchissement de la transition, la valeur du jeton « train » sera actualisée en fonction des éléments de liaison activés.

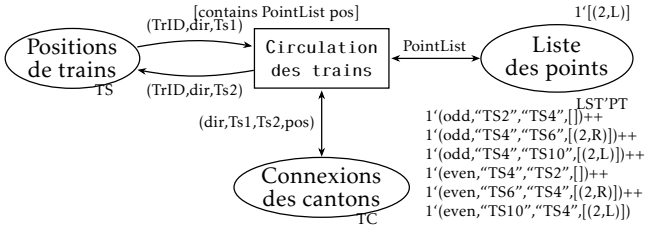


Figure 2 – Exemple de représentation généralisée

Dans les systèmes à relais, chaque événement, comme par exemple les commandes externes ou les actions de commutation internes, entraîne un changement d'état du circuit. Lors de la modélisation de tels systèmes, il exist des problèmes d'actions synchrones et de conditions « temporaires ». Nous avons conçu un mécanisme pour résoudre ces problèmes. Il est réalisé en introduisant des règles à base d'événements par une « place événement » dans les modèles ordinaires RdPC. Un modèle événementiel est un modèle RdP avec des transitions prioritaires et des conditions d'événements stockées dans la place événement (type de fusion). Ceci transforme les transitions connectées en « transitions dirigées événement » et permet aux événements interne/externe d'influencer la progression des jetons.

Comme les transitions à base d'événements expriment tous les changements internes, il y aura par conséquent une augmentation de l'espace d'état. Nous introduisons donc une méthode d'abstraction pour minimiser la taille de l'espace d'état du système. Du point de vue de l'analyse, les états internes ne sont pas utiles, car ils ont moins de valeur que les états stables. En effet, chaque état interne est un petit changement à l'intérieur des systèmes à relais. Lorsque le système termine tous les changements dans un chemin de l'espace d'état, le système atteint un nouvel état d'équilibre. Cela signifie une réponse complète à l'entrée externe (commande de contrôle ou actions des trains). Ainsi, les états d'événement-stable et les états d'absence d'événement sont plus concis pour décrire l'accessibilité d'un système de sécurité. Un exemple est représenté sur la figure 3.

Transformation de modèles réseau de Petri coloré en machine *B*

Cette étude est basée sur[Bon and Dutilleul 2013]. Nous introduisons un cadre plus explicite de la transformation, simplifions les opérations pour les structures multi-ensemble et rendons la traduction compatible avec les types non-numériques de jeux de couleurs. Ensuite, nous réformons les spécifications du multi-ensemble afin d'assurer que les machines B transformées peuvent être automatiquement prouvées par l'Atelier B.

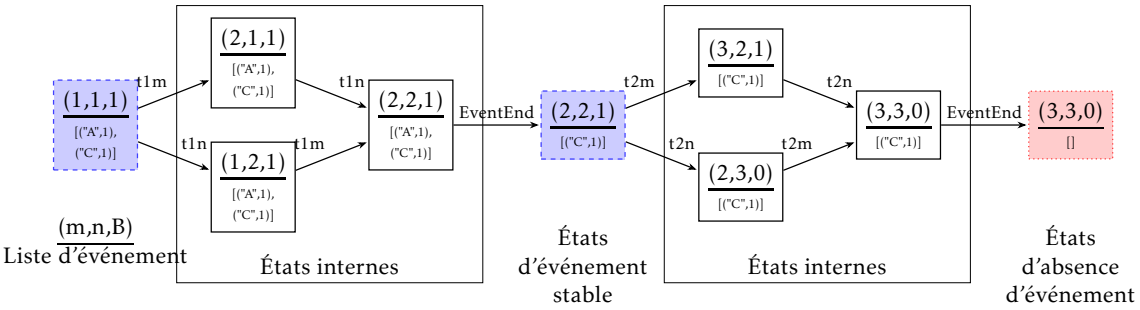


Figure 3 – Règles de simplification de l'état de l'espace du système

D'abord, nous étudions la transformation du modèle RdPC non-hiérarchique en machine *B*. Une version simplifiée du cadre de la transformation de modèle est mise en évidence dans le tableau 1, pour faire correspondre la structure « place-transition » du RdPC au formalisme du langage *B*.

Table 1 – Une version simplifiée du cadre modèle de transformation

Élément du RdPC	⇒	Notation de la machine *B*
Couleurs Σ		SETS and/or CONSTANTS
Places + Marquages $M(p)$		VARIABLES
Couleurs→Place C		INVARIANT
Transitions T		OPERATIONS
Liaison b		DEFINITIONS
Multi-ensembles		"MultiSets.def"

L'une des caractéristiques importantes du RdPC est l'existence de différents types de données. Un marquage du RdPC correspond à un multi-ensemble qui contient des informations de nombre et de couleur de jeton. Chaque action des transitions est associée aux modifications de marquage. Ainsi, les spécifications de multi-ensemble doivent être introduites dans le cadre de la machine *B*. Cependant, il est imprudent et inutile de prédéfinir tous les types de multi-ensembles dans une seule machine abstraite. Afin d'être plus adaptables et flexibles, nous introduisons un ensemble de définitions paramétrées, qui pourrait associer les numéros de jetons avec leurs couleurs et permettre la modification de jetons via des opérations. Ces définitions sont écrites dans un fichier « MultiSets.def » qui est directement inclus dans les machines abstraites comme indiqué dans le listing 1.

Listing 1 – Spécifications de multi-ensemble et de leurs opérations

```
DEFINITIONS
MS(ss)==(ss<->NATURAL);
Ms_Empty(ss)== {elt|elt: ss * {0}};
Ms_Subset(ms1,ms2,ss)== !elt.(elt:ss => ms1(elt)>=ms2(elt));
Ms_Add(ms1,ms2,ss) == (%ee.(ee:ss & ms1(ee):ran(ms1) & ms2(ee):ran(ms2) & ms1(ee)>=0 & ms2(
    ee)>=0 | ms1(ee) + ms2(ee)));
```

```
Ms_Less(ms1,ms2,ss)== (%ee.(ee:ss & ms1(ee):ran(ms1) & ms2(ee):ran(ms2) | ms1(ee) - ms2(ee)
    ))
```

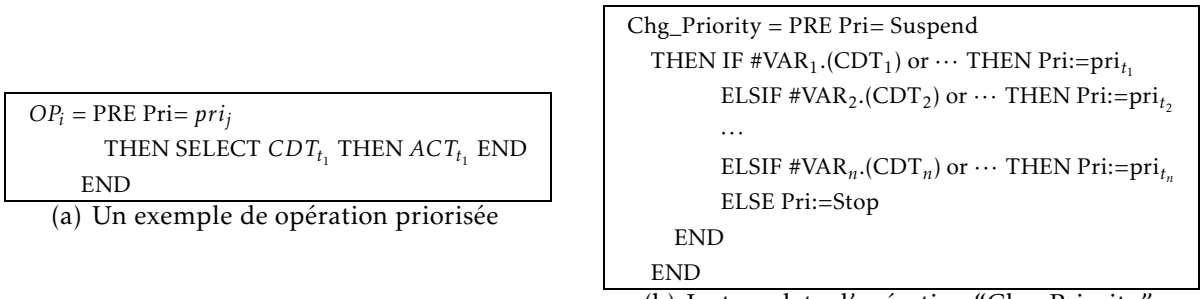
Après cela, nous présentons un schema global de preuve d'équivalence de la transformation de modèle. Nous analysons à la fois les attributs statiques et les comportements dynamiques pour démontrer l'équivalence de la transformation. Après le succès de la preuve de notre transformation, nous générons les graphes d'espace d'états à la fois du modèle RdPC et du modèle *B* généré. Nous comparons ces deux espaces d'états afin d'obtenir une preuve supplémentaire.

Le concept de hiérarchie du RdPC peut principalement résoudre deux types de problèmes. Tout d'abord, les systèmes-uniques qui sont un seul grand système constitué de plusieurs systèmes non-autonomes. Ensuite, les multi-systèmes (ou systèmes composites), qui sont un système intégré ou composé de plusieurs systèmes autonomes. Dans cette thèse, nous ne traitons que des multi-systèmes. Pour être précis, un modèle RdPCH d'un grand système sera traduit en un ensemble de machines abstraites. Chaque réseau unique aura son image après la transformation. Pour une telle transformation, la clause la plus appropriée du langage *B* est la clause « INCLUDES ». Elle permet à la machine incluante de lire l'ensemble des constituants des machines incluses.

La différence majeure par rapport à la transformation de RdPC non-hiérarchique tiens dans les transmissions de données par la structure hiérarchique. Dans les réseaux de Petri colorés hiérarchies (RdPCHs), la transmission de données entre différent modules est réalisée par des « places composées » qui sont localisées dans des modules différents mais qui partagent les mêmes valeurs. Néanmoins, une variable ne peut être définie qu'une seule fois dans les machines *B* générées. Cela signifie que chaque machine *B* ne peut définir que ses propres variables locales. Les autres ne serons accessibles qu'en utilisant des machines incluses. Si une variable doit être modifiée par d'autres machines, cela ne peut être fait que par des opérations prédéfinies. Nous utilisons cette méthode afin de simuler la fonction des places composées. L'emplacement de la place est dépendant de sa nature : si c'est une place normale, elle sera définie dans sa propre machine *B*, si c'est une place de fusion, elle sera définie dans la machine *B* racine et si c'est une place de port, elle sera définie dans la machine *B* du niveau supérieur.

Enfin, nous introduisons les transitions prioritaires dans notre méthode de transformation. Dans le RdPC avec des transitions prioritaire, toutes les priorités sont prédéfinis comme un ensemble fini de nombres naturels $\mathbb{P}$. Avant la validation des transitions, il y a besoin d'une comparaison des valeurs de priorité $\rho(t)$. Dans les machines *B* correspondantes, la définition de $\mathbb{P}$ peut être traduit par un ensemble d'énumération des identités utilisé pour distinguer les différentes priorités. Nous ne pouvons pas directement assigner un numéro à une opération dans les machines *B*, par conséquent nous considérons le numéro de priorité comme une condition nécessaire des opérations (figure 5.4(a)), et nous modélisons la

fonction de comparaison par une opération de gestion de priorité (figure 5.4(b)).



(a) Un exemple de opération priorisée

(b) Le template d'opération "Chg_Priority"

Figure 4 – Mangement de priorité

Conclusions

Cette thèse propose de nouvelles approches qui visent à aider les concepteurs à assurer efficacement la sécurité de chemin de fer et à améliorer la qualité de la conception du système et de sa vérification par l'utilisation des langages formels. Cette étude s'est concentrée sur deux aspects importants pour l'analyse du SEF français: les spécifications formelles et une évolution rapide de l'analyse à la mise en œuvre.

Modélisation formelle des systèmes d'enclenchement

Dans le domaine ferroviaire français, le SEF du type PRCI est la plus répandu. D'abord, pour ces systèmes, nous introduisons une approche de modélisation intuitive qui pourrait spécifier formellement les constructions des installations fixes et les opérations de signalisation de la logique du SEF. La partie de signalisation du SEF est modélisée par une décomposition verticale alors que la partie de l'installation fixe est modélisée de façon géographique. Puis, avec la puissance de la modélisation des RdPC, une solution générale est proposé par l'introduction d'un patron de modèle qui est un modèle sous une forme paramétrée. La structure de « place/transition » représente un ensemble de règles fonctionnelles RIS. La formation logique de chemin de fer est représenté par des jetons. Enfin, pour la partie de bas niveau du SEF, les « circuits logiques à relais », un concept basé sur les événements est introduit afin de mieux décrire les interactions internes et de réaliser une synchronisation relative.

Modélisation formelle des systèmes d'enclenchement

La transformation des RdPCs en machines B peut aider les concepteurs sur la route de l'analyse à application. Tout d'abord, pour le modèle du RdPC non-hiérarchique, nous introduisons un modèle de transformation par un cadre de correspondance systématique

dans lequel toutes les informations sur un modèle RdPC peuvent trouver leur partie correspondante dans la machine *B* générée. Le « multi ensembles » est le mécanisme central des RdPCs. Nous améliorons et étendons la solution de [Bon and Dutilleul 2013], de tell sorte que la nouvelle transformation soit compatible avec différentes couleurs de jetons. Cette amélioration permet aux machines *B* générées d'être automatiquement prouvés par l'Atelier B. Ensuite, le concept de hiérarchie est introduit dans la méthode de transformation pour effectuer la correspondance du RdPCH. Les liens hiérarchiques entre les machines connectées sont exprimées par la clause « INCLUDES » et les opérations prédéfinies dans les machines du niveau supérieur. En outre, nous introduisons un ensemble de règles de classification des places dans différentes machines. Enfin, nous considérons les priorités de transition de RdP et créons un mécanisme de gestion de priorité pour réaliser les opérations prioritaires dans les machines abstraites. Quelques nouveaux éléments sont ajoutés dans le cadre de la transformation, y compris une variable globale, un ensemble et une opération de gestion des priorités. Par la condition préalable des opérations basée sur la priorité, nous pouvons réaliser les fonctions similaires à celle des transitions priorisées.

Perspectives

Pour la parti théorique, le cadre du modèle et le cadre de la transformation de modèles sont extensibles, en effet, un certain nombre de facteurs ne sont pas pris en compte. Dans les recherches futures, on peut introduire le mode de défaillance dans le modèle RdPC ou encore considérer la transformation de « système-unique ».

Du point du vue pratique, cette thèse se place dans le projet PERFECT qui étudie la mise en œuvre de la sécurité de l'ERTMS en France. Les règles de fonctionnement de l'ERTMS sont spécifiés par la langage *B* dans certains recherches [Ben-ayed, Collart-Dutilleul, Bon, Idani, et al. 2014; Ben-ayed, Collart-Dutilleul, Bon, Ledru, et al. 2014]. Le travail d'intégration des travaux de Ben-Ayed est une perspective intéressante.

Bibliography

[Aalst and Odijk 1995] Aalst, W. M. van der and Odijk, M. (1995). "Analysis of railway stations by means of interval timed coloured Petri nets". In: *Real-Time Systems* 9(3), pp. 241–263 (cit. on p. 24).

[Aanæs and Thai 2012] Aanæs, M. and Thai, H. P. (2012). "Modelling and verification of relay interlocking systems". supervisor: Associate Professor Anne Haxthausen. MA thesis. Asmussens Alle, Building 305, DK-2800 Kgs. Lyngby, Denmark: Technical University of Denmark, DTU Informatics (cit. on p. 21).

[Abbott 2006] Abbott, R. (2006). "Complex systems + systems engineering = complex systems engineering". In: *Conference on Systems Engineering Research* (cit. on p. 12).

[Abrial 1996] Abrial, J.-R. (1996). *The B-book: assigning programs to meanings*. New York, NY, USA: Cambridge University Press (cit. on pp. 15, 22, 102).

[Abrial 2010] Abrial, J.-R. (2010). *Modeling in Event-B: system and software engineering*. Cambridge University Press (cit. on pp. 15, 22).

[Ahmad 2013] Ahmad, M. (2013). "Modeling and verification of functional and non functional requirements of ambient, self adaptative systems". PhD thesis. Toulouse: Université Toulouse le Mirail - Toulouse II (cit. on p. 17).

[Amrani, Combemale, et al. 2014] Amrani, M., Combemale, B., Lúcio, L., Selim, G. M. K., Dingel, J., Le Traon, Y., Vangheluwe, H., and Cordy, J. R. (2014). "Formal verification techniques for model transformations: a tridimensional classification". In: *Journal of object technology*, pp. 1–44 (cit. on p. 25).

[Amrani, Lúcio, et al. 2012] Amrani, M., Lúcio, L., Selim, G., Combemale, B., Dingel, J., Vangheluwe, H., Le Traon, Y., and Cordy, J. R. (2012). "A tridimensional approach for studying the formal verification of model transformations". In: *ICST 2012, 5th International conference on software testing, verification and validation*. Montreal, QC (cit. on p. 25).

[Antoni 2009a] Antoni, M. (2009a). "Formal validation method and tools for french computerized railway interlocking systems". In: *International journal of railway* 2(3), pp. 99–106 (cit. on p. 25).

[Antoni 2009b] Antoni, M. (2009b). "Formal validation method for computerized railway interlocking systems". In: *CIE 2009, International conference on computers industrial engineering*, pp. 1532–1541 (cit. on p. 25).

[Antoni 2009c] Antoni, M. (2009c). "Validation d'automatismes ferroviaires de sécurité à base de réseaux de Petri". PhD thesis. Braunschweig, Allemagne: Technischen Universität Carolo-Wilhelmina zu Braunschweig (cit. on pp. 25, 30, 31, 33, 38, 44).

[Antoni 2012a] Antoni, M. (2012a). "Formal validation method and tools for computerized interlocking system". In: *FM industry day*, pp. 1–44 (cit. on pp. 25, 33).

[Antoni 2012b] Antoni, M. (2012b). "Méthode de validation formelle d'un poste d'aiguillage informatique". In: *Recherche transports sécurité* 28(2), pp. 101–118 (cit. on pp. 25, 56, 160).

[Antoni and Ammad 2007] Antoni, M. and 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 (cit. on p. 25).

[Antoni and Ammad 2008] Antoni, M. and 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 (cit. on p. 25).

[Atlas 2005] Atlas, I. (2005). "KM3: Kernel MetaMetaModel". In: *Rapport technique, LINA & INRIA, Nantes*, p. 36 (cit. on p. 26).

[Attiogbe 2009] Attiogbe, C. (2009). "Semantic embedding of Petri nets into Event-*B*". In: *IFM 2009, Integration of model-based formal methods tools* (cit. on pp. 26, 102, 109).

[Bacherini et al. 2006] Bacherini, S., Fantechi, A., Tempestini, M., and Zingoni, N. (2006). "A story about formal methods adoption by a railway signaling manufacturer". In: *FM 2006, formal methods*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 179–189 (cit. on pp. 18, 20).

[Badeau and Amelot 2005] Badeau, F. and Amelot, A. (2005). "Using *B* as a high level programming language in an industrial project: Roissy VAL". In: *ZB 2005, formal specification and Development in Z and B*. Springer, pp. 334–354 (cit. on p. 23).

[Banci, Fantechi, and Gnesi 2004] Banci, M., Fantechi, A., and Gnesi, S. (2004). "The role of formal methods in developing a distributed railway interlocking system". In: *FORMS/FORMAT 2004*, pp. 220–230 (cit. on p. 72).

[BAppSc 1998] BAppSc, C. W. J. (1998). "Modelling and analysis of railway network control logic using coloured Petri Nets". PhD thesis. University of South Australia (cit. on p. 24).

[Bar-Yam 2003] Bar-Yam, Y. (2003). *Dynamics of complex systems*. Westview Press (cit. on p. 11).

[Basagiannis, Katsaros, and Pombortsis 2006] Basagiannis, S., Katsaros, P., and Pombortsis, A. (2006). "Interlocking control by distributed signal boxes: design and verification with the SPIN model checker". In: *ISPA 2006, 4th International Symposium on Parallel and Distributed Processing and Applications*. Sorrento, Italy, pp. 317–328 (cit. on p. 21).

[Bause 1997] Bause, F. (1997). "Analysis of Petri nets with a dynamic priority method". In: *Application and theory of Petri nets*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 215–234 (cit. on p. 57).

[BEA-TT and AET 2009] BEA-TT and AET (2009). *Rapport d'enquête techniquesur la collision ferroviaire survenue le 11 octobre 2006 à la frontière franco-luxembourgeoise à Zoufftgen (Moselle)*. Tech. rep. (cit. on p. 3).

[Behm, Benoit, et al. 1999] Behm, P., Benoit, P., Faivre, A., and Meynadier, J.-M. (1999). "Météor: a successful application of *B* in a large project". In: *Petri nets: central models and their properties*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 369–387 (cit. on pp. 16, 22).

[Behm, Desforges, and Meynadier 1998] Behm, P., Desforges, P., and Meynadier, J.-M. (1998). "MÉTÉOR: an industrial success in formal development". In: *B 1998, Recent advances in the development and use of the B method*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 26–26 (cit. on p. 22).

[Ben-ayed, Bon, and Collart-Dutilleul 2014] Ben-ayed, R., Bon, P., and Collart-Dutilleul, S. (2014). "Checking the European railways traffic management system (ERTMS) operating rules using UML and the *B* method". In: *Computers in railways XIV: railway engineering design and optimization* 135, p. 139 (cit. on p. 157).

[Ben-ayed, Collart-Dutilleul, Bon, Idani, et al. 2014] Ben-ayed, R., Collart-Dutilleul, S., Bon, P., Idani, A., and Ledru, Y. (2014). "B formal validation of ERTMS/ETCS railway operating rules". In: *Abstract State Machines, Alloy, B, TLA, VDM, and Z*. Springer, pp. 124–129 (cit. on pp. 157, 167).

[Ben-ayed, Collart-Dutilleul, Bon, Ledru, et al. 2014] Ben-ayed, R., Collart-Dutilleul, S., Bon, P., Ledru, Y., and Idani, A. (2014). "Modélisation et validation formelle des règles d'exploitation ferroviaires". In: *Approches formelles dans l'assistance au développement de logiciels*, 15p (cit. on pp. 157, 167).

[Berger, Middelraad, and Smith 1992] Berger, J., Middelraad, P., and Smith, A. J. (1992). "EURIS, european railway interlocking specification". In: *IRSE proceedings*. UIC, Commission, pp. 70–82 (cit. on p. 20).

[Bernardeschi et al. 1998] Bernardeschi, C., Fantechi, A., Gnesi, S., Larosa, S., Mongardi, G., and Romano, D. (1998). "A formal verification environment for railway signaling system design". In: *Formal methods in system design* 12(2), pp. 139–161 (cit. on p. 20).

[Berryman and Campbell 2010] Berryman, M. J. and Campbell, P. (2010). "Some complex systems engineering principles". In: *Australias preeminent systems engineering and test evaluation conference 2010*, pp. 1–11 (cit. on p. 11).

[Best and Koutny 1992] Best, E. and Koutny, M. (1992). "Petri net semantics of priority systems". In: *Theoretical Computer Science* 96(1), pp. 175–215 (cit. on pp. 57, 63–65, 187, 192, 193).

[Bézivin et al. 2006] Bézivin, J., Büttner, F., Gogolla, M., Jouault, F., Kurtev, I., and Lindow, A. (2006). "Model transformations ? transformation models !" In: *Model driven engineering languages and systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 440–453 (cit. on p. 26).

[Bidoit and Mosses 2004] Bidoit, M. and Mosses, P. D. (2004). *CASL user manual*. Ed. by Bidoit, M. and Mosses, P. D. Vol. 2900. Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg (cit. on p. 16).

[Billington 1991] Billington, J. (1991). "Many-sorted High-level Nets". In: *High-level Petri nets: theory and application*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 123–136 (cit. on p. 24).

[Bjørk 2006] Bjørk, J. (2006). "Executing large scale colored Petri nets by using Maude". PhD thesis. Oslo, Norway: University of Oslo (cit. on pp. 24, 161).

[Bjørner 1978] Bjørner, D. (1978). *The Vienna development method: the meta-language*. Ed. by Bjørner, D. and Jones, C. B. Vol. 61. Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg (cit. on p. 15).

[Bjørner 2003] 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 (cit. on p. 18).

[Bjørner 2012a] Bjørner, D. (2012a). "A survey of domain engineering". In: *APSEC 2012, 19th Asia-Pacific software engineering conference* 2, pp. 145–145 (cit. on p. 22).

[Bjørner 2012b] Bjørner, D. (2012b). "A survey of formal methods in software development". In: *APSEC 2012, 19th Asia-Pacific Software Engineering Conference* (cit. on p. 22).

[Bjørner 2014] Bjørner, D. (2014). "Domain engineering a basis for safety critical software". In: *ASSC 2014, Australian system safety conference*, pp. 26–28 (cit. on p. 22).

[Bjørner and Havelund 2014] Bjørner, D. and Havelund, K. (2014). "40 years of formal methods". In: *Petri nets: central models and their properties*. Cham: Springer International Publishing, pp. 42–61 (cit. on pp. 15, 16, 22).

[Bon 2000] Bon, P. (2000). "Du cahier des charges aux spécifications formelles : une méthode basée sur les réseaux de Petri de haut niveau". PhD thesis. Université des Sciences et Techniques de Lille (cit. on pp. 26, 109).

[Bon, Collart-Dutilleul, and Sun 2013] Bon, P., Collart-Dutilleul, S., and 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 (cit. on p. 4).

[Bon and Dutilleul 2013] Bon, P. and Dutilleul, S. C. (2013). "From a solution model to a *B* model for verification of safety properties". In: *Journal of universal computer science* 19(1), pp. 2–24 (cit. on pp. 6, 7, 26, 102, 109, 155, 163, 167).

[Bonacchi and Fantechi 2014] Bonacchi, A. and Fantechi, A. (2014). "On the validation of an interlocking system by model-checking". In: *Formal methods for industrial critical systems*. Cham: Springer International Publishing, pp. 94–108 (cit. on p. 21).

[Bonacchi, Fantechi, et al. 2014] Bonacchi, A., Fantechi, A., Bacherini, S., Tempestini, M., and Cipriani, L. (2014). "Validation of railway interlocking systems by formal verification, A case study". In: *Software engineering and formal methods*. Cham: Springer International Publishing, pp. 237–252 (cit. on pp. 20, 21).

[Boulanger 2013a] Boulanger, J.-L. (2013a). *Formal methods: industrial use from model to the code*. ISTE. Wiley (cit. on pp. 101, 161).

[Boulanger 2013b] Boulanger, J.-L. (2013b). *Industrial use of formal methods: formal verification*. ISTE. Wiley (cit. on pp. 101, 161).

[Boulanger 2014] Boulanger, J.-L., ed. (2014). *Formal methods applied to complex systems*. Implementation of the *B* method. Hoboken, NJ, USA: John Wiley & Sons, Inc. (cit. on pp. 23, 105).

[Bowen and Stavridou 1993] Bowen, J. and Stavridou, V. (1993). "Safety-critical systems, formal methods and standards". In: *Software engineering journal* 8(4), pp. 189–209 (cit. on p. 22).

[Buchheit et al. 2011] 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* (cit. on pp. 25, 56, 160).

[Calegari and Szasz 2013] Calegari, D. and Szasz, N. (2013). "Verification of model transformations: a survey of the state-of-the-art". In: *Electronic notes in theoretical computer science* 292, pp. 5–25 (cit. on pp. 25, 121).

[Cansell and Méry 2012] Cansell, D. and Méry, D. (2012). "Foundations of the *B* method". In: *Computing and informatics* 22(3-4), pp. 221–256 (cit. on pp. 103, 104, 109).

[Cavada et al. 2009] Cavada, R., Cimatti, A., Mariotti, A., Mattarei, C., Micheli, A., Mover, S., Pensallorto, M., Roveri, M., Susi, A., and Tonetta, S. (2009). "Supporting requirements validation: the EuRailCheck tool". In: *ASE 2009, 24th IEEE/ACM international conference on automated software engineering*, pp. 665–667 (cit. on p. 19).

[Chen 2006] Chen, C. (2006). "CiteSpace II: detecting and visualizing emerging trends and transient patterns in scientific literature". In: *Journal of the American society for information science and technology* 57(3), pp. 359–377 (cit. on p. 14).

[Chen, Hu, et al. 2012] Chen, C., Hu, Z., Milbank, J., and Schultz, T. (2012). "A visual analytic study of retracted articles in scientific literature". In: *Journal of the American society for information science and technology* 64(2), pp. 234–253 (cit. on p. 14).

[Chen, Ning, and Xu 2007] Chen, L., Ning, B., and Xu, T. (2007). "Research on modeling and simulation of vehicle-on-board Automatic Train Protection subsystem of Commu-

nication Based Train Control system". In: *ICVES 2007, IEEE international conference on vehicular electronics and safety*, pp. 1–5 (cit. on p. 25).

[Chen, He, and Huang 2011] Chen, X., He, Y., and Huang, H. (2011). "An approach to automatic development of interlocking logic based on Statechart". In: *Enterprise information systems* 5(3), pp. 273–286 (cit. on p. 20).

[Cheng and Yang 2009] Cheng, Y.-H. and Yang, L.-A. (2009). "A fuzzy Petri nets approach for railway traffic control in case of abnormality: evidence from taiwan railway system". In: *Expert systems with applications* 36(4), pp. 8040–8048 (cit. on p. 24).

[Cimatti, Corvino, et al. 2012] Cimatti, A., Corvino, R., Lazzaro, A., Narasamdya, I., Rizzo, T., Roveri, M., Sanseviero, A., and Tchaltsev, A. (2012). "Formal verification and validation of ertms industrial railway train spacing system". In: *Petri nets: central models and their properties*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 378–393 (cit. on p. 23).

[Cimatti, Giunchiglia, et al. 1998] Cimatti, A., Giunchiglia, F., Mongardi, G., Romano, D., Torielli, F., and Traverso, P. (1998). "Formal verification of a railway interlocking system using model checking". In: *Formal aspects of computing* 10(4), pp. 361–380 (cit. on p. 20).

[Cimatti, Roveri, and Susi 2008] Cimatti, A., Roveri, M., and Susi, A. (2008). *ETCS requirements specification and validation: the methodology*. Tech. rep. ERA/2007/ERTMS/02. European Railway Agency (cit. on p. 19).

[Clarke Jr, Grumberg, and Peled 1999] Clarke Jr, E. M., Grumberg, O., and Peled, D. A. (1999). *Model checking*. MA, USA: MIT Press Cambridge (cit. on p. 16).

[Clavel et al. 2007] Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., and Talcott, C. (2007). *All about maude - a high-performance logical framework*. Vol. 4350. Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg (cit. on p. 16).

[ClearSy 2011] ClearSy (2011). *Type Checker - Error Message Manual*. English. Version Version 3.6. CLEARSY. 87 pp. January 29, 2009 (cit. on pp. 103, 132).

[ClearSy 2014] ClearSy (2014). *B Language Reference Manual*. English. Version Version 1.8.7. CLEARSY. 231 pp. (cit. on pp. 103, 104, 130).

[Collart-Dutilleul et al. 2014] Collart-Dutilleul, S., Bon, P., El-Koursi, E., and 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 (cit. on pp. 4, 52, 75).

[Combemale et al. 2009] Combemale, B., Crégut, X., Garoche, P.-L., and Thirioux, X. (2009). "Essay on semantics definition in MDE. an instrumented approach for model verification". In: *Journal of software* 4(9), pp. 943–958 (cit. on pp. 25, 26).

[Czarnecki and Helsen 2006] Czarnecki, K. and Helsen, S. (2006). "Feature-based survey of model transformation approaches". In: *IBM systems journal* 45(3), pp. 621–645 (cit. on pp. 25, 26, 101).

[DaSilva, Dehbonei, and Mejia 1992] DaSilva, C., Dehbonei, B., and Mejia, F. (1992). "Formal specification in the development of industrial applications: subway speed control system". In: *FORTE 1992, 5th IFIP conference on formal description techniques for distributed systems and communication protocols*. Perros-Guirec, North-Holland: North-Holland Publishing Co., pp. 199–213 (cit. on p. 19).

[Decknatel 1999] Decknatel, G. (1999). "Modelling train movement with hybrid Petri nets". In: *FMRail workshop*, pp. 11–12 (cit. on p. 24).

[Decknatel and Schnieder 1998] Decknatel, G. and Schnieder, E. (1998). "Modelling railway systems with hybrid Petri nets". In: *International conference on automation of mixed process*. Reims, France, pp. 309–315 (cit. on p. 24).

[Defossez, Collart-Dutilleul, and Bon 2011] Defossez, F., Collart-Dutilleul, S., Bon, P., et al. (2011). "A formal model of requirements". In: *Open transportation journal* 5, pp. 60–70 (cit. on p. 102).

[Diaconescu and Futatsugi 1998] Diaconescu, R. and Futatsugi, K. (1998). "CafeOBJ report: the language, proof techniques, and methodologies for object-oriented algebraic specification". In: *World scientific publishing* (cit. on p. 16).

[Dijk et al. 1998] Dijk, F. v., Fokkink, W., Kolk, G., Ven, P. v. d., and Vlijmen, B. v. (1998). "EURIS, a specification method for distributed interlockings". In: *Computer safety, reliability and security* 1516(Chapter 23), pp. 296–305 (cit. on p. 20).

[Dincel, Eriş, and Kurtulan 2013] Dincel, E., Eriş, O., and Kurtulan, S. (2013). "Automata-based railway signaling and interlocking system design". In: *IEEE antennas and propagation magazine* 55(4), 308–318 (cit. on p. 21).

[DO 178B 2010] DO 178B (2010). *Software considerations in airborne systems and equipment certification*. Norm (cit. on p. 14).

[EN 50128 2011] EN 50128 (2011). *Railway applications – communication, signalling and processing systems – software for railway control and protection systems*. Norm (cit. on pp. 4, 14, 19).

[EN 50128 2001] EN 50128 (2001). *Railway applications – communication, signalling and processing systems – software for railway control and protection systems*. Norm (cit. on p. 19).

[Erbin and Soulas 2003] Erbin, J. and Soulas, C. (2003). "Twenty years of experiences with driverless metros in France". In: *VWT19*, pp. 1–33 (cit. on p. 102).

[Eriş and Mutlu 2010] Eriş, O. and Mutlu, İ. (2010). "Design of signal control structures using formal methods for railway interlocking systems". In: *Control, automation, robotics and vision*, pp. 776–780 (cit. on p. 21).

[FAA 2014] FAA (2014). *FAA systems engineering manual*. 1.0. USA: Federal Aviation Administration (cit. on p. 12).

[Fantechi 2012a] Fantechi, A. (2012a). "Distributing the challenge of model checking interlocking control tables". In: *Leveraging applications of formal methods, verification and validation. applications and case studies*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 276–289 (cit. on p. 21).

[Fantechi 2012b] Fantechi, A. (2012b). "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 (cit. on p. 18).

[Fantechi 2014a] Fantechi, A. (2014a). "On validating software at layout changes in interlocking systems". In: *FORMS 2014, Symposium on formal methods for railway operation and control systems*, pp. 198–212 (cit. on pp. 20, 21).

[Fantechi 2014b] Fantechi, A. (2014b). "Twenty-five years of formal methods and railways: what next?" In: *Software engineering and formal methods*. Cham: Springer International Publishing, pp. 167–183 (cit. on pp. 18, 19).

[Fantechi, Flammini, and Gnesi 2014] Fantechi, A., Flammini, F., and Gnesi, S. (2014). "Formal methods for railway control systems". In: *International journal on software tools for technology transfer* 16(6), pp. 643–646 (cit. on p. 18).

[Fantechi, Fokkink, and Morzenti 2012] Fantechi, A., Fokkink, W., and Morzenti, A. (2012). "Some trends in formal methods applications to railway signaling". In: *Formal Methods*

for Industrial Critical Systems. Hoboken, NJ, USA: John Wiley & Sons, Inc., pp. 61–84 (cit. on p. 18).

[Fanti, Giua, and Seatzu 2006] Fanti, M. P., Giua, A., and Seatzu, C. (2006). "Monitor design for colored Petri nets: an application to deadlock prevention in railway networks". In: *Control engineering practice* 14(10), pp. 1231–1247 (cit. on p. 24).

[Ferbeck 1981] Ferbeck, D. (1981). "Le systeme VAL applique au metro de Lille". In: *REV GEN CHEM FER* (cit. on p. 23).

[Ferrari, Fantechi, et al. 2009] Ferrari, A., Fantechi, A., Bacherini, S., and Zingoni, N. (2009). "Modeling guidelines for code generation in the railway signaling context". In: *NFM 2009, 1st NASA formal methods symphosium*. Moffet Field, CA, USA., pp. 166–170 (cit. on p. 20).

[Ferrari, Grasso, et al. 2010] Ferrari, A., Grasso, D., Magnani, G., Fantechi, A., and Tempestini, M. (2010). "The Metrô Rio ATP case study". In: *Formal methods for industrial critical systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–16 (cit. on p. 21).

[Ferrari, Magnani, et al. 2011] Ferrari, A., Magnani, G., Grasso, D., and Fantechi, A. (2011). "Model checking interlocking control tables". In: *FORMS/FORMAT 2010, 6th Symposium on formal methods for automation and safety in railway and automotive systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 107–115 (cit. on pp. 21, 23).

[Feuser and Peleska 2014] Feuser, J. and Peleska, J. (2014). "Dependability in open proof software with hardware virtualization—The railway control systems perspective". In: *Science of computer programming* 91, pp. 188–215 (cit. on p. 19).

[France 2010] France, R. (2010). "Bridging the formal techniques and model-driven engineering divide". In: *NSF Microsoft workshop on usable verification*, pp. 1–5 (cit. on p. 17).

[France and Rumpe 2007] France, R. and Rumpe, B. (2007). *Model-driven development of complex software: a research roadmap*. IEEE Computer Society (cit. on p. 17).

[Genrich 1987] Genrich, H. J. (1987). "Predicate/transition nets". In: *Petri nets: central models and their properties*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 207–247 (cit. on p. 24).

[George 1991] George, C. (1991). "The RAISE specification language a tutorial". In: *VDM 1991, Formal software development methods*. Berlin/Heidelberg: Springer Berlin Heidelberg, pp. 238–319 (cit. on p. 21).

[Ghazel 2009] Ghazel, M. (2009). "Using stochastic Petri nets for level-crossing collision risk assessment". In: *IEEE transactions on intelligent transportation systems* 10(4), 668–677 (cit. on p. 24).

[Ghazel 2014] Ghazel, M. (2014). "Formalizing a subset of ERTMS/ETCS specifications for verification purposes". In: *Transportation research part C: emerging technologies* 42, pp. 60–75 (cit. on p. 19).

[Giua and DiCesare 1993] Giua, A. and DiCesare, F. (1993). "GRAFCET and Petri nets in manufacturing". In: *Intelligent manufacturing:* London: Springer London, pp. 153–176 (cit. on pp. 54, 57).

[Giua and Seatzu 2008] Giua, A. and Seatzu, C. (2008). "Modeling and supervisory control of railway networks using Petri nets". In: *Automation science and engineering* 5(3), pp. 431–445 (cit. on p. 24).

[Gnesi et al. 2000] Gnesi, S., Lenzini, G., Latella, D., Abbaneo, C., Amendola, A., and Marmo, P. (2000). "An automatic SPIN validation of a safety critical railway control system". In: *International conference on dependable systems and networks (includes FTCS-30 30th Annual international symposium on fault-tolerant computing and DCCA-8)*, pp. 119–124 (cit. on p. 20).

[Goverde, Corman, and D'Ariano 2013] Goverde, R. M. P., Corman, F., and D'Ariano, A. (2013). "Railway line capacity consumption of different railway signalling systems under scheduled and disturbed conditions". In: *Journal of rail transport planning & management* 3(3), pp. 78–94 (cit. on p. 156).

[Grégory, Nicolae, et al. 2010] Grégory, B., Nicolae, B., Lalouette, J., and Max, W. (2010). "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* (cit. on p. 25).

[Grégory, Olaf, et al. 2013] Grégory, B., Olaf, M., Nicolae, B., and Nadia, A. (2013). "Dependability assessment of large railway systems". In: *RAMS 2013, Reliability and maintainability symposium*, pp. 1–6 (cit. on p. 21).

[Groote, Vlijmen, and Koorn 1995] Groote, J. F., Vlijmen, S. F. M. van, and Koorn, J. W. C. (1995). "The safety guaranteeing system at station Hoorn-Kersenboogerd". In: *COMPASS 1995, 10th Conference on computer assurance, systems integrity, software safety and process security*. Gaithersburg, MD, pp. 57–68 (cit. on p. 20).

[Gruner et al. 2013] Gruner, S., Haxthausen, A. E., Maibaum, T., and Roggenbach, M. (2013). "Towards a formal methods body of knowledge for railway control and safety systems". In: *FM-RAIL-BOK Workshop 2013* (cit. on p. 15).

[Guiho and Hennebert 1990] Guiho, G. and Hennebert, C. (1990). "SACEM software validation (experience report)". In: *Software engineering*. Nice, pp. 186–191 (cit. on p. 22).

[Hagalisletto et al. 2007] Hagalisletto, A. M., Bjørk, J., Yu, I. C., Yu, I. C., and Enger, P. (2007). "Constructing and refining large-scale railway models represented by Petri nets". In: *Systems, man, and cybernetics, part C: applications and reviews* 37(4), pp. 440–460 (cit. on pp. 24, 161).

[Hammoudi, Vale, and Lopes 2008] Hammoudi, S., Vale, S., and Lopes, D. (2008). "Vers un processus semi-automatique de la transformation dans l'approche MDA". In: *4èmes journées sur l'ingénierie dirigée par les modèles*, pp. 1–16 (cit. on p. 25).

[Hartonas-Garmhausen et al. 2000] Hartonas-Garmhausen, V., Campos, S., Cimatti, A., Clarke, E., and Giunchiglia, F. (2000). "Verification of a safety-critical railway interlocking system with real-time constraints". In: *Science of computer programming* 36(1), pp. 53–64 (cit. on p. 20).

[Hase 2011] Hase, K.-R. (2011). "'Open proof' for railway safety software - a potential way-out of vendor lock-in advancing to standardization, transparency, and software security". In: *FORMS/FORMAT 2010, formal methods for automation and safety in railway and automotive systems symposium*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 5–38 (cit. on p. 19).

[Haxthausen 2010] Haxthausen, A. E. (2010). *An introduction to formal methods for the development of safety-critical applications*. Tech. rep. Technical University of Denmark, pp. 1–32 (cit. on pp. 15, 23).

[Haxthausen 2011] Haxthausen, A. E. (2011). "Towards a framework for modelling and verification of relay interlocking systems". In: *Petri nets: central models and their properties*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 176–192 (cit. on p. 21).

[Haxthausen 2014] Haxthausen, A. E. (2014). "Automated generation of formal safety conditions from railway interlocking tables". In: *International journal on software tools for technology transfer* 16(6), pp. 713–726 (cit. on p. 21).

[Haxthausen, Kjær, and Le Bliguet 2011] Haxthausen, A. E., Kjær, A. A., and Le Bliguet, M. (2011). "Formal development of a tool for automated modelling and verification of relay

interlocking systems". In: *Petri nets: central models and their properties*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 118–132 (cit. on p. 21).

[Haxthausen, Le Bliguet, and Kjær 2010] Haxthausen, A. E., Le Bliguet, M., and Kjær, A. A. (2010). "Modelling and verification of relay interlocking systems". In: *Petri nets: central models and their properties*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 141–153 (cit. on p. 21).

[Haxthausen, Peleska, and Haxthausen 2003] Haxthausen, A. E., Peleska, J., and Haxthausen, A. E. (2003). "Generation of executable railway control components from domain-specific descriptions". In: *FORMS 2003, Symposium on formal methods for railway operation and control systems* (cit. on p. 20).

[Haxthausen, Peleska, and Pinger 2014] Haxthausen, A. E., Peleska, J., and Pinger, R. (2014). "Applied bounded model checking for interlocking system designs". In: *Parallel and distributed processing and applications*. Ed. by Counsell, S. and Nunez, M. Gewerbestrasse 11, Cham, CH-6330, Switzerland: Springer INT Publishing AG, 205–220 (cit. on p. 21).

[Hede 2012] Hede, K. (2012). "Formal modelling and verification of railway time tables". supervisor: Associate Professor Anne Haxthausen. MA thesis. Asmussens Alle, Building 305, DK-2800 Kgs. Lyngby, Denmark: Technical University of Denmark, DTU Informatics (cit. on p. 21).

[Hennebert and Guiho 1993] Hennebert, C. and Guiho, G. (1993). "SACEM: a fault tolerant system for train speed control". In: *FTCS 1993, 23th International symposium on fault-tolerant computing*. Toulouse, France, pp. 624–628 (cit. on p. 22).

[Hielscher et al. 1998] Hielscher, W., Urbszat, L., Reinke, C., and Kluge, W. (1998). "On modelling train traffic in a model train system". In: *Workshop and tutorial on practical use of coloured Petri nets and Design/CPN*. Aarhus Denmark (cit. on p. 24).

[Hoare 1985] Hoare, C. A. R. (1985). *Communicating sequential processes*. Prentice Hall International (cit. on p. 16).

[Holloway, Krogh, and Giua 1997a] Holloway, L. E., Krogh, B. H., and Giua, A. (1997a). "A survey of Petri net methods for controlled discrete event systems". In: *Discrete event dynamic systems* 7(2), pp. 151–190 (cit. on pp. 24, 53).

[Holloway and Krogh 1994] Holloway, L. and Krogh, B. (1994). "Controlled Petri nets: A tutorial survey". English. In: *11th International conference on analysis and optimization of systems discrete event systems*. Ed. by Cohen, G. and Quadrat, J.-P. Vol. 199. Lecture Notes in Control and Information Sciences. Springer Berlin Heidelberg, pp. 158–168 (cit. on p. 87).

[Holloway, Krogh, and Giua 1997b] Holloway, L., Krogh, B., and Giua, A. (1997b). "A survey of Petri net methods for controlled discrete event systems". English. In: *Discrete event dynamic systems* 7(2), pp. 151–190 (cit. on p. 87).

[Holzmann 2003] Holzmann, G. (2003). *The SPIN model checker: primer and reference manual*. Addison-Wesley Professional (cit. on p. 16).

[Hörste and Schnieder 1999] Hörste, M. M. zu and Schnieder, E. (1999). "Formal modelling and simulation of train control systems using Petri nets". In: *Petri Nets: central models and their properties*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1867–1867 (cit. on pp. 19, 24, 156).

[Huang, Weng, and Zhou 2010] Huang, Y.-S., Weng, Y.-S., and Zhou, M. (2010). "Critical scenarios and their identification in parallel railroad level crossing traffic control systems". In: *IEEE transactions on intelligent transportation systems* 11(4), 968–977 (cit. on p. 24).

[Huber, Jensen, and Shapiro 1991] Huber, P., Jensen, K., and Shapiro, R. M. (1991). "Hierarchies in coloured Petri nets". In: *Advances in Petri Nets 1990*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 313–341 (cit. on p. 57).

[IEC 1988] IEC (1988). *IEC 60848: Preparation of function charts for control systems*. Tech. rep. International Electrotechnical Commission (cit. on p. 54).

[IEC 61508 2010] IEC 61508 (2010). *Functional safety of electrical/electronic/programmable electronic safety-related systems (E/E/PE, or E/E/PES)*. Norm (cit. on p. 14).

[IEEE 1990] IEEE (1990). *IEEE standard computer dictionary. a compilation of IEEE standard computer glossaries*. Norm (cit. on p. 30).

[Iliasov, Lopatkin, and Romanovsky 2013] Iliasov, A., Lopatkin, I., and Romanovsky, A. (2013). "The SafeCap project on railway safety verification and capacity simulation". In: *Software Engineering for Resilient Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 125–132 (cit. on p. 19).

[Iliasov, Palacin, and Romanovsky 2014] Iliasov, A., Palacin, R., and Romanovsky, A. (2014). "Advanced modelling, simulation and verification for future traffic regulation optimisation". In: *Software engineering for resilient systems*. Cham: Springer International Publishing, pp. 131–138 (cit. on p. 19).

[Iliasov and Romanovsky 2012] Iliasov, A. and Romanovsky, A. (2012). "SafeCap domain language for reasoning about safety and capacity". In: *Workshop on dependable transportation systems/recent advances in software dependability* (cit. on p. 19).

[INCOSE 2012] INCOSE (2012). *Systems engineering handbook: A guide for system life cycle processes and activities*. 3.2.2. San Diego, CA, USA: INCOSE 2012, International council on systems engineering (cit. on p. 12).

[IOS/IEC 2002] IOS/IEC (2002). *High-level Petri nets - concepts, definitions and graphical notation*. Final draft international standard, version 4.7.1 (cit. on p. 187).

[ITU-T 2011] ITU-T (2011). "ITU-T Rec. Z.120 (02/2011) Message Sequence Chart (MSC)". In: pp. 1–146 (cit. on p. 16).

[James, Moller, Nguyen, Roggenbach, Schneider, Treharne, et al. 2014] James, P., Moller, F., Nguyen, H. N., Roggenbach, M., Schneider, S., Treharne, H., Trumble, M., and Williams, D. (2014). "Verification of scheme plans using CSP||B". In: *Software engineering and formal methods*. Cham: Springer International Publishing, pp. 189–204 (cit. on p. 21).

[James 2014] James, P. (2014). "Designing domain specific languages for verification and applications to the railway domain". PhD thesis (cit. on p. 21).

[James, Lawrence, et al. 2014] James, P., Lawrence, A., Moller, F., Roggenbach, M., Seisenberger, M., Setzer, A., Kanso, K., and Chadwick, S. (2014). "Verification of solid state interlocking programs". In: *Software engineering and formal methods*. Ed. by Counsell, S. and Nunez, M. Springer INT Publishing AG, 253–268 (cit. on pp. 20, 21).

[James, Moller, Nguyen, Roggenbach, Schneider, and Treharne 2014a] James, P., Moller, F., Nguyen, H. N., Roggenbach, M., Schneider, S., and Treharne, H. (2014a). "Decomposing scheme plans to manage verification complexity". In: *FORMS/FORMAT 2014, 10th Symposium on formal methods for automation and safety in railway and automotive systems*, pp. 1–9 (cit. on p. 21).

[James, Moller, Nguyen, Roggenbach, Schneider, and Treharne 2014b] James, P., Moller, F., Nguyen, H. N., Roggenbach, M., Schneider, S., and Treharne, H. (2014b). "On modelling and verifying railway interlockings: tracking train lengths". In: *Science of computer programming*, pp. 1–22 (cit. on p. 21).

[James, Moller, Nguyen, Roggenbach, Schneider, and Treharne 2014c] James, P., Moller, F., Nguyen, H. N., Roggenbach, M., Schneider, S., and Treharne, H. (2014c). "Techniques for

modelling and verifying railway interlockings". In: *International journal on software tools for technology transfer (J.STTT)* 16(6), pp. 685–711 (cit. on p. 21).

[James and Roggenbach 2011] James, P. and Roggenbach, M. (2011). "Automatically verifying railway interlockings using SAT-based model checking". In: *AVoCS 2010, 10th International workshop on automated verification of critical systems*. Ed. by Bendispoto, J., Leuschel, M., and Roggenbach, M. Electronic Communications of the EASST (cit. on pp. 21, 23).

[Janhsen, Lemmer, Horste, et al. 1997] Janhsen, A., Lemmer, K., Horste, M. M. zu, and 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, signalling, telecommunications and non-conventional systems*. Florence, pp. 101–118 (cit. on p. 68).

[Janhsen, Lemmer, Ptok, et al. 1997] Janhsen, A., Lemmer, K., Ptok, B., and Schnieder, E. (1997). "Formal specification of the European train control system". In: *Proceedings of transportation systems*. Chania, pp. 1215–1220 (cit. on pp. 68, 69).

[Jansen, Meyer Zu Hörste, and Schnieder 1998] Jansen, L., Meyer Zu Hörste, M., and 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*. Aarhus, Denmark: Citeseer, pp. 103–115 (cit. on pp. 19, 68, 69).

[Jensen 1981] Jensen, K. (1981). "Coloured Petri nets and the invariant-method". In: *Theoretical computer science* 14(3), pp. 317–336 (cit. on pp. 24, 52, 57).

[Jensen 1987] Jensen, K. (1987). "Coloured Petri nets". In: *Petri nets: central models and their properties*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 248–299 (cit. on pp. 24, 52, 57, 187).

[Jensen 1996] Jensen, K. (1996). *Coloured Petri nets: basic concepts, analysis methods and practical use*. Monographs in theoretical computer science. an EATCS series. Berlin, Heidelberg: Springer Berlin Heidelberg (cit. on pp. 57, 59).

[Jensen and Kristensen 2009] Jensen, K. and Kristensen, L. M. (2009). *Coloured Petri nets: modelling and validation of concurrent systems*. Berlin, Heidelberg: Springer Science & Business Media (cit. on pp. 57, 58, 113, 187, 188).

[Jung 2000] Jung, B. (2000). "Die methode und werkzeuge GRACE". In: *Fortschritt berichte-vdi reihe 12 verkehrstechnik fahrzeugtechnik*, pp. 173–184 (cit. on p. 20).

[Jurack and Taentzer 2010] Jurack, S. and Taentzer, G. (2010). "A Component concept for typed graphs with inheritance and containment structures: long version". In: *ICGT 2010, 5th International conference on graph transformation*, pp. 1–22 (cit. on p. 25).

[Kaakai, Hayat, and El Moudni 2007] Kaakai, F., Hayat, S., and El Moudni, A. (2007). "A hybrid Petri nets-based simulation model for evaluating the design of railway transit stations". In: *Simulation modelling practice and theory* 15(8), pp. 935–969 (cit. on p. 24).

[Kanso, Moller, and Setzer 2009] Kanso, K., Moller, F., and Setzer, A. (2009). "Automated verification of signalling principles in railway interlocking systems". In: *Electronic notes in theoretical computer science* 250(2), pp. 19–31 (cit. on p. 23).

[Knight 2002] Knight, J. C. (2002). "Safety critical systems: challenges and directions". In: *ICSE 2002, 24th International conference on software engineering*. ACM, pp. 547–550 (cit. on p. 13).

[König and Einer 2003] König, N. H. and Einer, S. (2003). "The Euro-Interlocking formalized functional requirements approach (EIFFRA)". In: *FORMS 2003, 4th Symposium on*

formal methods for railway operation and control systems. Budapest. L'Harmattan Hongrie (cit. on p. 20).

[Korečko, HudáK, and ŠIMOŇÁK 2007] Korečko, Š., HudáK, Š., and ŠIMOŇÁK, S. (2007). "Petri net-like treatment of B-machine behaviour". In: *Journal of information, control and management systems* 5(2) (cit. on pp. 26, 102, 109).

[Korečko and Sobota 2014] Korečko, Š. and Sobota, B. (2014). "Petri nets to B-language transformation in software development". In: *Acta polytechnica hungarica* 11(6) (cit. on pp. 26, 102, 109).

[Kossiakoff et al. 2011] Kossiakoff, A., Sweet, W. N., Seymour, S. J., and Biemer, S. M. (2011). *Systems engineering principles and practice*. Kossiakoff/Systems Engineering 2E. Hoboken, NJ, USA: John Wiley & Sons, Inc. (cit. on p. 12).

[Kühne et al. 2009] Kühne, T., Mezei, G., Syriani, E., Vangheluwe, H., and Wimmer, M. (2009). "Systematic transformation development". In: *MPM 2009, 3rd International workshop on multi-paradigm modeling*, pp. 1–11 (cit. on p. 26).

[Lalouette et al. 2010] 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*, (cit. on pp. 25, 56, 160).

[Lano and Haughton 1996] Lano, K. and Haughton, H. (1996). *Specification in B: an introduction using the B toolkit*. World Scientific (cit. on p. 103).

[Lara and Vangheluwe 2009] Lara, J. de and Vangheluwe, H. (2009). "Automating the transformation-based analysis of visual languages". In: *Formal aspects of computing* 22(3-4), pp. 297–326 (cit. on p. 26).

[Larrucea, Combelles, and Favaro 2013] Larrucea, X., Combelles, A., and Favaro, J. (2013). "Safety-critical software [guest editors' introduction]". In: *IEEE Software* 30(3), pp. 35–37 (cit. on p. 13).

[Le Bouar 2003] Le Bouar, P. (2003). "Interlocking SNCF functional requirements description". In: *Euro-Interlocking project* (cit. on p. 20).

[Lecomte 2008] Lecomte, T. (2008). "Safe and reliable metro platform screen doors control/command systems". In: *FM 2008, formal methods*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 430–434 (cit. on p. 23).

[Lecomte, Servat, and Pouzancre 2007] Lecomte, T., Servat, T., and Pouzancre, G. (2007). "Formal methods in safety-critical railway systems". In: *BSFM 2007, Brazilian symposium on formal methods*, pp. 1–9 (cit. on pp. 22, 23).

[Lei, Zhang, et al. 2013] Lei, L., Zhang, Y., Shen, X., Lin, C., and Zhong, Z. (2013). "Performance analysis of device-to-device communications with dynamic interference using stochastic Petri nets". In: *IEEE transactions on wireless communications* 12(12), pp. 6121–6141 (cit. on p. 25).

[Lei, Tanglong, and Jian 2000] Lei, T., Tanglong, C., and Jian, X. (2000). "Analysis of the concurrent model of train station based on Petri net". In: *International workshop on autonomous decentralized systems*. Chengdu, China, pp. 92–96 (cit. on p. 24).

[Leuschel et al. 2011] Leuschel, M., Falampin, J., Fritz, F., and Plagge, D. (2011). "Automated property verification for large scale *B* models with ProB". In: *Formal Aspects of Computing* 23(6), pp. 683–709 (cit. on p. 23).

[Leveson 1995] Leveson, N. G. (1995). *Safeware: system safety and computers*. Addison-Wesley (cit. on p. 13).

[Lijie et al. 2012] Lijie, C., Tao, T., Xianqiong, Z., and Schnieder, E. (2012). "Verification of the safety communication protocol in train control system using colored Petri net". In: *Reliability engineering & system safety* 100, pp. 8–18 (cit. on p. 19).

[Lopes et al. 2006] Lopes, D., Hammoudi, S., Bézivin, J., and Jouault, F. (2006). "Mapping specification in MDA: from theory to practice". In: *Interoperability of enterprise software and applications*. London: Springer London, pp. 253–264 (cit. on p. 25).

[Lúcio et al. 2014] Lúcio, L., Amrani, M., Dingel, J., Lambers, L., Salay, R., Selim, G. M. K., Syriani, E., and Wimmer, M. (2014). "Model transformation intents and their properties". In: *Software & systems modeling*, pp. 1–38 (cit. on pp. 25, 101).

[Lukman and Mernik 2008] Lukman, T. and Mernik, M. (2008). "Model-driven engineering and its introduction with metamodeling tools". In: *9th International PhD workshop on systems and control young generation viewpoint*. Izola, Slovenia, pp. 1–6 (cit. on p. 17).

[Maximova, Ehrig, and Ermel 2010] Maximova, M., Ehrig, H., and Ermel, C. (2010). "Formal relationship between Petri Net and graph transformation systems based on functors between M-adhesive categories". In: *PNGT 2010, 4th International workshop on Petri nets and graph transformation* (cit. on p. 26).

[Mens and Van Gorp 2006] Mens, T. and Van Gorp, P. (2006). "A taxonomy of model transformation". In: *Electronic notes in theoretical computer science* 152, pp. 125–142 (cit. on p. 26).

[Milner 1980] Milner, R., ed. (1980). *A calculus of communicating systems*. Vol. 92. Lecture notes in computer science. Berlin, Heidelberg: Springer Berlin Heidelberg (cit. on p. 16).

[Milner 1997] Milner, R. (1997). *The definition of standard ML: revised*. MIT press (cit. on p. 59).

[Moen and Yu 2004] Moen, A. and Yu, I. C. (2004). "Large scale construction of railroad models from specifications". In: *IEEE International conference on systems, man and cybernetics*, pp. 6212–6219 (cit. on pp. 24, 161).

[Moller et al. 2013] Moller, F., Nguyen, H. N., Roggenbach, M., Schneider, S., and Treharne, H. (2013). "Defining and model checking abstractions of complex railway models using CSP||B". In: *Petri nets: central models and their properties*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 193–208 (cit. on p. 23).

[Morley 1994] Morley, M. J. (1994). "Safety in railway signalling data: A behavioural analysis". In: *Higher order logic theorem proving and its applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 464–474 (cit. on p. 20).

[Murata 1989] Murata, T. (1989). "Petri nets: properties, analysis and applications". In: *Proceedings of the IEEE* 77(4), pp. 541–580 (cit. on pp. 24, 53).

[Murphy 2007] Murphy, E. (2007). "The application of ERTMS/ETCS systems". In: *IRSE 2007, Australasia technical meeting on institution of railway signal engineers* (cit. on p. 51).

[NASA 2004] NASA (2004). *NASA software safety guidebook*. National aeronautics and space administration: NASA Technical Standard (cit. on p. 13).

[Object Management Group 2003] Object Management Group (2003). *Meta object facility (MOF) 2.0 core specification*. 2.0 (cit. on p. 26).

[Ostroff 1989] Ostroff, J. S. (1989). *Temporal logic for real time systems*. John Wiley & Sons, Inc. (cit. on p. 16).

[Pachl 2002] Pachl, J. (2002). *Railway operation and control*. VTSTD rail publishing (cit. on p. 43).

[Patin, Pouzancre, and Servat 2006] Patin, F., Pouzancre, G., and Servat, T. (2006). "A formal approach in the implementation of a safety system for automatic control of 'platform doors'". In: *4th Annual conference on system engineering* (cit. on p. 23).

[Peterson 1981] Peterson, J. L. (1981). *Petri net theory and the modeling of systems*. Upper Saddle River, NJ, USA: Prentice Hall PTR (cit. on p. 53).

[Petri 1962] Petri, C. A. (1962). "Kommunikation mit Automaten". ger. PhD thesis. Darmstadt, Germany: Universität Hamburg (cit. on pp. 24, 53).

[Petri 1966] Petri, C. A. (1966). "Communication with Automata". In: *second edition: New York: Griffiss Air Force Base, technical report RADC-TR-65–377* 1 (cit. on pp. 16, 53).

[Rastocny, Janota, and Zahradnik 2004] Rastocny, K., Janota, A., and Zahradnik, J. (2004). "The use of UML for development of a railway interlocking system". In: *Integration of software specification techniques for applications in engineering*. Heidelberger Platz 3, d-14197 Berlin, Germany: Springer-Verlag Berlin, 174–198 (cit. on p. 20).

[Reisigs and Rozenberg 1998] Reisigs, W. and Rozenberg, G. (1998). "Informal introduction to Petri nets". In: *Lectures on Petri nets I: basic models* (cit. on p. 53).

[Ren and Zhou 1995] Ren, X. and Zhou, M. (1995). "Tactical scheduling of rail operations: a Petri net approach". In: *1995 IEEE international conference on systems, man and cybernetics. intelligent systems for the 21st century*, pp. 3087–3092 (cit. on p. 24).

[René and Alla 1992] René, D. and Alla, H. (1992). *Petri nets and grafcet - tools for modelling discrete event systems*. Prentice Hall (cit. on p. 56).

[René and Alla 1997] René, D. and Alla, H. (1997). "Du grafcet aux réseaux de Petri". In: *Ouvrage ISBN13: 978-2-86601-325* 7 (cit. on p. 56).

[Rétiveau 1987] Rétiveau, R. (1987). *La signalisation ferroviaire*. Presse de l'école nationale des Ponts et Chaussées (cit. on pp. 45, 71, 75, 83, 84).

[Robinson 1997] Robinson, K. (1997). "The *B* method and the *B* toolkit". In: *Algebraic Methodology and Software Technology*. Springer, pp. 576–580 (cit. on p. 103).

[Rosaria et al. 2003] Rosaria, E., Armando, L., Pietro, M., and Angela, S. (2003). "Formal verification of ertms euroradio safety critical protocol". In: *FORMS 2003, formal methods for railway operation and control systems*. L'Harmattan, Budapest, Hongrie, pp. 1–9 (cit. on p. 19).

[Sabatier et al. 2012] Sabatier, D., Burdy, L., Requet, A., and Guéry, J. (2012). "Formal proofs for the NYCT line 7 (flushing) modernization project". In: *Abstract State Machines, Alloy, B, VDM, and Z*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 369–372 (cit. on p. 23).

[Santiago et al. 2012] Santiago, I. n., Jiménez, Á., Vara, J. M., De Castro, V., Bollati, V. A., and Marcos, E. (2012). "Model-driven engineering as a new landscape for traceability management: A systematic literature review". In: *Information and software technology* 54(12), pp. 1340–1356 (cit. on p. 17).

[Schmidt 2006] Schmidt, D. C. (2006). "Guest Editor's Introduction: Model-Driven Engineering". In: *Computer* 39(2), pp. 25–31 (cit. on p. 17).

[Schneider, Treharne, and Wehrheim 2012] Schneider, S., Treharne, H., and Wehrheim, H. (2012). "The behavioural semantics of Event-*B* refinement". In: *Formal aspects of computing* 26(2), pp. 251–280 (cit. on p. 16).

[Seidewitz 2003] Seidewitz, E. (2003). "What models mean". In: *IEEE Software* 20(5), pp. 26–32 (cit. on p. 17).

[Steinberg, Budinsky, and Paternostro 2009] Steinberg, D., Budinsky, F., and Paternostro, M. (2009). *EMF: eclipse modeling framework 2.0*. Addison-Wesley Professional (cit. on p. 26).

[Storey 1996] Storey, N. R. (1996). *Safety critical computer systems*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. (cit. on p. 13).

[Sun, Collart-Dutilleul, and Bon 2014] Sun, P., Collart-Dutilleul, S., and 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 (cit. on pp. 5, 52, 68, 75, 102).

[Sun, Collart-Dutilleul, and Bon 2015] Sun, P., Collart-Dutilleul, S., and 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 (cit. on p. 76).

[Syriani and Ergin 2012] Syriani, E. and Ergin, H. (2012). "Operational semantics of UML activity diagram: an application in project management". In: *MoDRE 2012, IEEE model-driven requirements engineering workshop*, pp. 1–8 (cit. on p. 26).

[The RAISE Language Group 1992] The RAISE Language Group (1992). *The RAISE specification language*. Prentice Hall (cit. on p. 21).

[The RAISE Language Group 1995] The RAISE Language Group (1995). *The RAISE development method*. Prentice Hall Int. (cit. on p. 21).

[UIC-ETF 2014] UIC-ETF (2014). *UIC safety database report 2014*. Tech. rep. 16 rue Jean Rey - F-75015 Paris: UIC Safety Unit, p. 36 (cit. on p. 3).

[Umamageswaran, Pandey, and Wilsey 1999] Umamageswaran, K., Pandey, S. L., and Wilsey, P. A. (1999). "Interval Temporal Logic". In: *Formal semantics and proof techniques for optimizing VHDL models*. Boston, MA: Springer US, pp. 65–68 (cit. on p. 16).

[Union Technique de l'Electricitè 1982] Union Technique de l'Electricitè (1982). *Function chart "GRAFCET" for the description of logic control systems*. French standard NF C03-190 (cit. on p. 53).

[Varró, Gyapay, et al. 2006] Varró, D., Gyapay, S. V., Ehrig, H., Prange, U., and Taentzer, G. (2006). "Termination analysis of model transformations by Petri nets". In: *Graph transformations*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 260–274 (cit. on p. 26).

[Varró and Pataricza 2003] Varró, D. and Pataricza, A. (2003). "Automated formal verification of model tranformations". In: *CSDUML 2003*, pp. 63–78 (cit. on p. 26).

[Wang and Bai 2010] Wang, F. and 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 (cit. on p. 25).

[Westergaard and Verbeek 2011] Westergaard, M. and Verbeek, H. M. W. (2011). "Efficient implementation of prioritized transitions for high-level Petri nets". In: *PNSE 2011, Petri Nets and software engineering. international workshop*. Ed. by Duvigneau, M., Moldt, D., and Hiraishi, K. Vol. 723. CEUR Workshop Proceedings. CEUR-WS.org, pp. 27–41 (cit. on pp. 57, 58, 188).

[Westergaard and Verbeek 2013] Westergaard, M. and Verbeek, H. M. W. (2013). *Efficient implementation of simulation of prioritized transitions for high-level Petri nets*. Tech. rep. BPM-13-24. BPM center report (cit. on pp. 57, 58, 64, 65, 187, 188, 192, 193).

[Wikipedia 2003] Wikipedia (2003). *Life-critical system — Wikipedia, The Free Encyclopedia*. [Online; accessed 31-January-2015] (cit. on p. 13).

[Wimmer, Kappel, and Kusel 2009] Wimmer, M., Kappel, G., and Kusel, A. (2009). "Right or wrong ? – verification of model transformations using colored Petri nets". In: *DSM 2009, 9th OOPSLA Workshop on Domain-Specific Modeling*, pp. 1–6 (cit. on p. 26).

[Wimmer, Kusel, Reiter, et al. 2009] Wimmer, M., Kusel, A., Reiter, T., Retschitzegger, W., Schwinger, W., and Kappel, G. (2009). "Lost in translation ? transformation nets to the rescue !" In: *Information systems: modeling, development, and integration*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 315–327 (cit. on p. 26).

[Wimmer, Kusel, Schoenboeck, Kappel, et al. 2009] Wimmer, M., Kusel, A., Schoenboeck, J., Kappel, G., Retschitzegger, W., and Schwinger, W. (2009). "Reviving QVT relations: model-based debugging using colored Petri nets". In: *Model driven engineering languages and systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 727–732 (cit. on p. 26).

[Wimmer, Kusel, Schoenboeck, Reiter, et al. 2009] Wimmer, M., Kusel, A., Schoenboeck, J., Reiter, T., Retschitzegger, W., and Schwinger, W. (2009). "Let's play the token game – model transformations powered by transformation nets". In: *Workshop on Petri nets and software engineering*. Paris (cit. on p. 26).

[Winter, Johnston, et al. 2006] Winter, K., Johnston, W., Robinson, P., Strooper, P., and Berg, L. van den (2006). "Tool support for checking railway interlocking designs". In: *In Proceedings of the th Aus- tralian Workshop on Safety Critical Systems and Software*, pp. 101–107 (cit. on p. 21).

[Winter and Robinson 2003] Winter, K. and Robinson, N. J. (2003). "Modelling large railway interlockings and model checking small ones". In: *ACSC 2003, 26th Australasian computer science conference*. Adelaide, South Australia, pp. 309–316 (cit. on p. 21).

[Woodcock and Davies 1999] Woodcock, J. and Davies, J. (1999). "Using Z: specification, refinement, and proof". In: *Pren- tice Hall International Series in Computer Science* (cit. on p. 15).

[Woodcock, Larsen, et al. 2009] Woodcock, J., Larsen, P. G., Bicarregui, J., and Fitzgerald, J. (2009). "Formal methods : practice and experience". In: *ACM computing surveys (CSUR)* 41(4), pp. 1–36 (cit. on p. 22).

[Wu and Zhou 2004] Wu, N. and Zhou, M. (2004). "Modeling and deadlock control of automated guided vehicle systems". In: *IEEE/ASME transactions on mechatronics* 9(1), pp. 50–57 (cit. on p. 24).

[Xu and Tang 2007] Xu, T. and 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 (cit. on p. 25).

[You, Li, and Xia 2012] You, J., Li, J., and Xia, S. (2012). "A survey on formal methods using in software development". In: *ICISCE 2012, International conference on information science and control engineering*. Institution of Engineering and Technology, pp. 2.40–2.40 (cit. on p. 15).

[Yu 2004] Yu, I. C. (2004). "A layered approach to automatic construction of large scale Petri nets". PhD thesis. Oslo, Norway: University of Oslo (cit. on pp. 24, 161).

[Zafar 2009] Zafar, N. A. (2009). "Formal specification and validation of railway network components using Z notation". In: *IET Software* 3(4), p. 312 (cit. on p. 20).

[Zafar, Khan, and Araki 2012] Zafar, N. A., Khan, S. A., and Araki, K. (2012). "Towards the safety properties of moving block railway interlocking system". In: *International journal of innovative computing information and control* 8(8), 5677–5690 (cit. on p. 20).

[Zaytoon and Villermain-Lecolier 1999] Zaytoon, J. and Villermain-Lecolier, G. (1999). "Grafcet: methodological and formal issues". In: *Advances in manufacturing*. London: Springer London, pp. 101–114 (cit. on pp. 54, 55).

[Zhou and Hansen 2004] Zhou, C. and Hansen, M. R. (2004). *Duration calculus: a formal approach to real-time systems*. Monographs in theoretical computer science an EATCS series. Berlin, Heidelberg: Springer Berlin Heidelberg (cit. on p. 16).

[Zhu et al. 2012] Zhu, L., Yu, F. R., Ning, B., and 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 (cit. on p. 25).

[Zimmermann and Hommel 2003] Zimmermann, A. and 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. (Cit. on p. 25).

Appendix A

Definition of coloured Petri nets and related concepts

The Appendix provides a formal definition of coloured Petri net and the definitions of the related concepts which are utilized throughout the thesis. This appendix begin with introducing some background knowledge regarding the CPN definitions. Then we introduce the concept of set and multi-set used in the thesis, which is based on the standard [IOS/IEC 2002]. Next, the definition of non-hierarchical CPN and HCPN, adopted from [Jensen and Kristensen 2009], is presented. Finally, we introduce the Transition priorities. In order to keep the definitions self consistent, we adapt and integrate the definitions of Best and Koutny 1992; Westergaard and Verbeek 2013 into Kurt Jensen's definition framework.

Background

CPN can be represented in two different ways [Jensen 1987]:

1. *Graphically*, by expressing in directed bipartite graphs with place, transition, arcs, initial markings and complementary inscriptions.
2. *Algebraically*, by providing a many tuple mathematical definition of sets and functions

In Jensen's theory both forms are equivalent, and he provide a translation procedure between them [Jensen 1987]. Later the extended (hierarchical and time concepts) graphical and algebraic forms are provided in [Jensen and Kristensen 2009].

Sets and multi-sets

Sets

In mathematics, a set is a collection of numbers or distinct objects.

Let $N = \{0, 1, 2, \cdots\}$ denote the set of natural numbers. Let $bool = \{true, false\}$. Then we have the following relations:

- $true \in bool$
- $\emptyset \subseteq bool$

- $bool \subseteq bool$
- $\{true\} \subset bool$
- $\{true\} \cup \{false\} = bool$
- $3 \notin bool$
- $N_a = \{4, 2, 1, 3\} = \{1, 2, 3, 4\} = \{1, 2, 2, 3, 3, 4, 1, 1\}$
- $N_a = N_b$ if and only if $N_a \subseteq N_b$ and $N_a \supseteq N_b$.

Multi-sets

Definition A.1. A multi-set B over a basis set A is a maplet function:

$$B : A \to N$$

In the CPN's framework, the function B associates a natural number with each of the basis elements. A multi-set can be noted as a symbolic sum of basis elements scaled by their numbers: $B = \sum_{a \in A} B(a) * a$. For example, if $A = (x, y)$ then $B = 2x + 3y$ is the symbolic sum representation of the multi-set $\{(x, 2), (y, 3)\}$.

Set of multi-sets

The colour set of a place in CPNs is a set of multi-sets.

Definition A.2. Let C be the set of all multi-set over A, where $C = [A \to N]$ and Let $B_1, B_2 \in C$, then the following relations exists:

- $B_1 = B_2 \iff \forall a \in A, B_1(a) = B_1(b)$
- $B_1 \leq B_2 \iff \forall a \in A, B_1(a) \leq B_1(b)$
- for $n \in N, B = n * B_1 \iff \forall a \in A, B_1(a) = n * B_1(b)$

Definition of coloured Petri nets

Non-hierarchical coloured Petri nets

This basis definitions are adopt from the book of "*Coloured Petri nets: modelling and validation of concurrent systems*" [Jensen and Kristensen 2009] and the papers of Westergaard and Verbeek [Westergaard and Verbeek 2011, 2013].

Definition A.3. A non-hierarchical CPN is a nine-tuple $CPN = (P, T, A, \Sigma, V, C, G, E, I)$, where:

(i) P is a finite set of **places**.

(ii) T is a finite set of **transitions**, $P \cap T = \phi$.

(iii) $A \subseteq P \times T \cup T \times P$ is a finite set of **arcs**, .

(iv) Σ is a finite set of non-empty **colour sets**.

(v) V is a finite set **typed variables**, $Type[v] \in \Sigma$ for all variables $v \in V$.

(vi) $C : P \to \Sigma$ is a **colour set function** that assigns a colour set to each place

(vii) $G : T \rightarrow EXPR_v$ is a **guard function**, that assigns a guard to each transition, $\forall t \in T$ such that $Type[G(t)] = Bool$ and $Type[Var(G(t))] \subseteq \Sigma$.

(viii) $E : A \rightarrow EXPR_v$ is an **arc expression function**, that assigns an expression to each arc, $\forall a \in A$ that $Type[E(a)] = C(p)_{\text{MS}}$ and $Type[Var(E(t))] \subseteq \Sigma$.

(ix) $I : P \rightarrow EXPR_\phi$ is an **initialisation function**, that assigns an initialisation expression to each place, $\forall p \in P$ that $Type[I(p)] = C(p)_{\text{MS}}$.

Enabling and Occurrence rules of non-hierarchical coloured Petri nets

To formally define the behaviour of CPNs, some semantic concepts and notations should be introduced.

Definition A.4. For a CPN, there exists the following concepts:

(i) A **marking** M is a mapping from places into multi-set of tokens, for $\forall p \in P : M(p) \in C(p)_{\text{MS}}$,

(ii) The **initial marking** M_0 is defined as $\forall p \in P : M_0(p) = I(p)\langle\rangle$.

(iii) The **variables of a transition** t is denoted as $Var(t) \subseteq V$ and $Var(t) = \{v \mid v \in Var(G(t)) \vee \exists a \in A : v \in Var(E(a))\}$.

(iv) A **binding** of a transition t is a function b that maps each variable $v \in Var(t)$ into a value $b(v) \in Type[v]$. The set of all bindings for t is denoted by $B(t)$.

(v) A **binding element** is a couple (t, b) where $t \in T$ and $b \in B(t)$. The set of all binding elements in a CPN is denoted as BE.

(vi) An **arc expression** is $E(p, t)$ or $E(t, p)$ denote the arc expression on the input or output arc from p to t. In an enabled binding element (t, b), the multi-set of tokens removed from an input place p when t occurs in a binding b is given by $E(p, t)\langle b\rangle$, and $E(t, p)\langle b\rangle$ is the multi-set of tokens added to an output place p.

The behaviours of a CPN are the transfer of tokens, which are based on the enabling rules of the binding elements, there are represented as following.

Definition A.5. In the marking M, a binding element $(t, b) \in BE$ is called **enabled** if and only if the following two conditions are satisfied:

(i) $G(t)\langle b\rangle = ture$

(ii) $\forall p \in P : E(p, t)\langle b\rangle \ll= M(p)$.

After the firing the enabled transition t, the new marking of the system is:

(iii) $\forall p \in P : M'(p) = (M(p) - -E(p, t)\langle b\rangle) + +E(t, p)\langle b\rangle$.

The operator "++" takes two multi-sets as arguments and returns the union (the sum). In the same way, the operator "−−" the subtraction of two multi-sets.

The enabling and occurrence of steps can be summarised in a more general form, in which the Definition A.5 is a special case of the definition below.

Definition A.6. A step $Y \in BE_{\mathrm{MS}}$ is **enabled** in a marking M if and only if the following two properties are satisfied:

(i) $\forall (t,b) \in Y : G(t)\langle b \rangle = ture$

(ii) $\forall p \in P : \overset{++}{\underset{\mathrm{MS}}{\sum_{(t,b)\in Y}}} E(p,t)\langle b \rangle \ll= M(p).$

After the firing the enabled the step Y, the new marking M' of the system is:

(iii) $\forall p \in P : M'(p) = (M(p) -- \overset{++}{\underset{\mathrm{MS}}{\sum_{(t,b)\in Y}}} E(p,t)\langle b \rangle ++ \overset{++}{\underset{\mathrm{MS}}{\sum_{(t,b)\in Y}}} E(t,p)\langle b \rangle.$

When a step Y occurs in marking M_1, producing a new marking M_2, it can be denoted as $M_1 \xrightarrow{Y} M_2$.

Definition A.7. Let an infinite occurrence sequence be $M \xrightarrow{Y_1} M_1 \xrightarrow{Y_2} M_2 \xrightarrow{Y} \cdots$, then the set of marking reachable from the marking M is denoted as $\mathbb{R}(M)$.

Module of hierarchy

The hierarchy of HCPNs is represented by modules, each module constituting a non-hierarchical CPN as defined in Definition A.3. This means that each module can have its local colour set definitions and declarations of typed variables. Meanwhile, it should contain the information of hierarchical links. The definition of module in a HCPN model is based on the descriptions below.

Definition A.8. A **CPN module** is a four-tuple $CPN_M = (CPN, T_{sub}, P_{port}, PT)$, where:

(i) $CPN = (P, T, A, \Sigma, V, C, G, E, I)$ is a **non-hierarchical CPN**.

(ii) $T_{sub} \subseteq T$ is a set of **substitution transitions**.

(iii) $P_{port} \subseteq T$ is a set of **port places**.

(iv) $PT : P_{port} \to \{\text{IN, OUT, I/O}\}$ is a **port type function** that assigns a port type to each port place.

Hierarchical coloured Petri nets

A HCPN model consists of a finite set S of sub-modules. Each sub-module $s \in S$ is defined in the Definition A.8. Such modules can exchange tokens via *port-socket relations*. The **port places** P_{port} in the sub-modules and the **socket place** P_{sock} in the upper level modules are related. The **socket type function** $ST : P_{sock} \to \{IN, OUT, IO\}$ that assigns a sock type to each P_{sock}. It is also possible for modules to exchange tokens via **fusion sets**, which allow places in different modules to be glued together into one compound place across the hierarchical structure of the model. Then, the definition of a HCPN model is described as below.

Definition A.9. A **HCPN** is a four-tuple $CPN_H = (S, SF, PS, FS)$, where:

(i) S is a finite set of **sub-modules**. $s = ((P^s, T^s, A^s, \Sigma^s, V^s, C^s, G^s, E^s, I^s), P^s_{port}, T^s_{sub}, PT^s)$. It should be noted that, for all $s_1, s_2 \in S$ and $s_1 \neq s_2$ such that $(P^{s_1} \cup T^{s_1}) \cap (P^{s_2} \cup T^{s_2}) = \phi$.

(ii) $SF : T_{sub} \to S$ is a **sub-module function** that assigns a **sub-module** to each substitution transition. It is required that the module hierarchy be acyclic.

(iii) PS is a **port-socket relation function** that assigns a **port-socket relation** $PS(t) \subseteq P_{sock}(t) \times P_{port}^{SF(t)}$ to each substitution transition t. It is required that $ST(p) = PT(p')$, and $I(p)\langle\rangle = I(p')\langle\rangle$ for $\forall(p,p') \in PS(t)$ and $\forall t \in T_{sub}$

(iv) $FS \subseteq 2^P$ is a set of non-empty **fusion sets** such that $C(p) = C(p')$ and $I(p)\langle\rangle = I(p')\langle\rangle$ for $\forall p, p' \in fs$ and $\forall fs \in FS$.

The definition below summarises the definition of the module hierarchy.

Definition A.10. The **module hierarchy** for a HCPN is a direct graph $MH = (N_{MH}, A_{MH})$, where

(i) $N_{MH} = S$ is the set of **nodes**.

(ii) $A_{MH} = \{(s_1, t, s_2) \in N_{MH} \times T_{sub} \times N_{MH} \mid t \in T_{sub}^{s_1} \wedge s_2 = SF(t)\}$ is the set of arc.

(iii) The roots of MH are called **prime modules**, and the set of all prime modules is denoted S_{PM}.

(iv) For a module $s \in S$, the set of **module instances** is $MI^s = \{(s', t_1 t_2 \cdots t_n, s) \in S_{PM} \times T_{sub}^* \times S \mid s' \xrightarrow{t_1 t_2 \cdots t_n} s$ is a path in $MH\}$, The set of all instance of modules is denoted MI, where T_{sub}^* is to denote the set of all finite sequences of substitution transition. If the sequence is empty it will be denoted by ε.

(v) The **instance hierarchy** of CPN_H is directed graph $IH = (N_{IH}, A_{IH})$, where $N_{IH} = MI$ is the set of **nodes**, and $A_{IH} = \{((s', t_1 t_2 \cdots t_n, s_1), t, (s', t_1 t_2 \cdots t_n, s_2)) \in N_{IH} \times T_{sub} \times T_{IH}\}$ is the set of **arcs**.

Instances and compound places

The following definition describes the definition of place and transition instances, and of compound places.

Definition A.11. Let $CPN_H = (S, SF, PS, FS)$ be a HCPN and let $IH = (N_{IH}, A_{IH})$ be the instance hierarchy of CPN_H.

(i) The set of all **place instances** $PI_p = \{(p, s^*) \mid s^* \in MI^s\}$. The set of all **transition instances** $TI_t = \{(t, s^*) \mid s^* \in MI^s\}$. The set of all place instances and all transition instances is denoted by PI and TI.

(ii) The **place instance relation** $\sim_{cp} PI \times PI$ is the smallest equivalence relation containing all those pairs $((p_1, s_1^*), (p_2, s_2^*))$ that satisfy at least one of the following two conditions:

(a) A fusion set $fs \in FS$ exists, such that $p_1, p_2 \in fs$.

(b) An arc $(s_1^*, t, s_2^*) \in A_{IH}$ exists, such that $p_1, p_2 \in PS(t)$

(iii) The equivalence classes determined by $\sim_{cp}$ are called **compound places**. The set of compound places is denoted $[PI]$. For a place instance p^* we use $[p^*]$ to denote the compound place to which p^* belongs.

The concepts of variables of transitions, bindings, binding elements are defined in a way similar to that for non-hierarchical CPN models.

Definition A.12. For a HCPN, the following concepts exist:

(i) A **marking** is a function M that maps each compound place $[p^*]$ into a multi-set of tokens $M([p^*]) \in C(p)_{\text{MS}}$, where (p, s^*) is any place instance belonging to $[p^*]$.

(ii) The **initial marking** M_0 is defined by $M_0([p^*]) = I(p)\langle\rangle$.

(iii) The **variables of a transition** t^* of a transition t are denoted as $Var(t^*)$ and defined by $Var(t^*) = Var(t)$.

(iv) A **binding** of a transition t^* of a transition $t \in T - T_{sub}$ is a function b that maps each variable $v \in Var(t^*)$ into a value $b(v) \in Type[v]$. The set of all bindings for a transition instance t^* is denoted by $B(t^*)$.

(v) A **binding element** is a couple (t^*, b) that $t^* \in T$ is a transition instance of a transition $t \in T - T_{sub}$ and $b \in B(t^*)$. $BE(t)$ denotes the set of all binding elements for a transition instance t^*, $BE(t^*) = \{(t^*, b) \mid b \in B(t^*)\}$. The set of all binding elements in a CPN is denoted as BE.

Enabling and Occurrence rules of hierarchical coloured Petri nets

The enabling and occurrence of a binding element is defined as follow.

Definition A.13. A binding element $(t^*, b) \in BE_{\text{MS}}$ is **enabled** in a marking M if and only if the following two conditions are satisfied:

(i) $G(t^*)\langle b\rangle = ture$

(ii) $\forall p_{cp} \in [PI] : \overset{++}{\underset{\text{MS}}{\sum}}_{(t^*,b)\in BE_{\text{MS}}, p^* \in P_{cp}} E(p^*, t^*)\langle b\rangle \leq M(p_{cp})$

If (t^*, b) is enabled in M, it may occur, the new marking M' defined as:

$$\forall p_{cp} \in [PI] : M'(p_{cp}) = (M(p_{cp}) - - \overset{++}{\underset{\text{MS}}{\sum}}_{(t^*,b)\in BE_{\text{MS}}, p^* \in P_{cp}} E(p^*, t^*)\langle b\rangle) + + \overset{++}{\underset{\text{MS}}{\sum}}_{(t^*,b)\in BE_{\text{MS}}, p^* \in P_{cp}} E(t^*, p^*)\langle b\rangle.$$

Transition priorities

For low-level PNs, static priorities are defined in Best and Koutny 1992, which are *relational static* priorities. For high-level PNs, CPN Tools support the *absolute static* priority concept, which is described in Westergaard and Verbeek 2013.

As we are using the Kurt Jensen's definition form in our research, while the publication above using other forms of PN definitions. To normalize the all the definition, we adapt and integrate the definitions of Best and Koutny 1992; Westergaard and Verbeek 2013 into Kurt Jensen's definition framework. Then, we have the definitions as below:

Definition A.14. Static Priority (**Def. 3.1** in Best and Koutny 1992,**Def. 3** in Westergaard and Verbeek 2013) A static priority system is a pair (CPN, ρ), which:

1. $CPN = (P, T, A, \Sigma, V, C, G, E, I)$ is a CPN.
2. $\rho : T \to \mathbb{P}$ is a **priority function** that assigns a priority to each transition.

3. $\mathbb{P}$ is a finite set of natural numbers, called the **priorities** of the transitions.

The priorities in Definition A.14 are global and do not depend on the binding of the transition. Intuitively, if $\rho(t) < \rho(t')$ then t' has priority over t. Precisely, it means that t' has the priority to be enabled than t. When dealing with such priority systems, we say that if a transition is enabled according to Definition A.5, it is **pre-enabled**. Then, we can have the new enabling rules for priority systems as follows:

Definition A.15. Enabling with Priority (**Def. 3.3** in Best and Koutny 1992,**Def. 4** in Westergaard and Verbeek 2013). A transition $t \in T$ is enabled in the marking M if it is pre-enabled and no transition t' with $\rho(t) < \rho(t')$ is pre-enabled.

The algorithm for checking enabling with priority in CPN Tools is shown in Algorithm A.1. First, It sorts all transitions according to priority and processes them highest-priority-first until it reaches t. If it finds a pre-enabled transition with higher priority than t, it returns false. If it does not find a pre-enabled transition with higher priority than t it returns the pre-enable status of t.

Algorithm A.1 Algorithm for checking enabling with priority (**Algorithm 3** in [Westergaard and Verbeek 2013])

```
1: SortedTransitions← PrioritySort(T)
2: After any transitions or initialisation
3: procedure CheckEnablingPriority(t) is
4:     for all t' ∈ SortedTransitions do
5:         if ρ(t') > ρ(t) then
6:             if CheckEnabling(t') then
7:                 return false
8:         else
9:             return CheckEnabling(t)
```

Model based system engineering for safety of railway critical systems

Abstract

Development and application of formal languages are a long-standing challenge within the computer science domain. One particular challenge is the acceptance of industry. Although many successful stories have demonstrated the applicability of formal methods to industrial practice, their use within industry is still limited. This thesis presents some model-based methodologies for modelling and verification of the French railway interlocking systems (RIS), which aims to use formal methods to effectively ensure railway traffic safety.

This thesis mainly addresses two issues. The first one is the modellization of interlocking system by coloured Petri nets (CPNs). Next, a generic and compact modelling framework is introduced, in which the interlocking rules are modelled in a hierarchical structure while the railway layout is modelled in a geographical perspective. Then, a modelling pattern is presented. It is a parameterized model which respects the French national rules. It is a general reusable solution that can be applied in different stations. Then, an event-based concept is brought into the modelling process of low level part of RIS to have better description of internal interactions of low level relay-based interlocking logic.

The second issue is the transformation of coloured Petri nets into *B* machines, which can assist designers on the way from analysis to implementation. A mechanism describing the multi-sets and their behaviours in *B* machines is introduced to allow the following systematic translations. Firstly, a detailed mapping methodology from non-hierarchical CPNs to abstract *B* machine notations is presented. Then the hierarchy and the transition priority of CPNs are successively integrated into the mapping process, in order to enrich the adaptability of the transformation. This transformation is compatible with various types of colour sets and the transformed *B* machines can be automatically proved by Atelier B.

All these works at different levels contribute towards a global safe analysis framework.

Keywords: railway interlocking system, discrete event systems, CPN, modelling methodology

---

Ingénierie de modèle pour la sécurité des systèmes critiques ferroviaires

Résumé

Le développement et l'application des langages formels sont un défi à long terme pour la science informatique. Un enjeu particulier est l'acceptation par l'industrie. Malgré des succès industriels avérés, la dissémination des méthodes formelles dans la pratique industrielle ferroviaire est encore limitée. Cette thèse présente une approche pour la modélisation et la vérification des postes d'aiguillage français, qui utilise les méthodes formelles pour assurer efficacement la sécurité ferroviaire.

Cette thèse se concentre sur deux questions. La première est la modélisation du système d'enclenchement par les réseaux de Petri colorés (RdPC). Un cadre de modélisation générique et compact est introduit, dans lequel les règles d'enclenchement sont modélisées dans une structure hiérarchique, tandis que les installations sont modélisées dans une perspective géographique. Ensuite, un patron de modèle est présenté. C'est un modèle paramétré qui intègre les règles nationales françaises. C'est une solution générale et réutilisable qui peut être appliquée pour différentes gares. Puis, un concept basé sur l'événement est présenté dans le processus de modélisation des parties basses des postes d'aiguillage. Ce dernier fournit une meilleure description des interactions internes de la logique d'enclenchement à relais.

La deuxième question est la transformation des RdPCs en machines B, qui va aider les concepteurs sur la route de l'analyse à application. Des mécanismes décrivant les multi-ensembles et leurs comportements dans les machines *B* sont introduits. Il permettent la transformation systématique. Tout d'abord, une méthodologie détaillée, s'appuyant sur une table de correspondance, du RdPCs non-hiérarchiques vers les notations *B* est présentée. Ensuite, la hiérarchie et la priorité des transitions du RdPC sont successivement intégrées dans le processus de mapping, afin d'enrichir les possibilités de types de modèles en entrées de la transformation. Cette transformation est compatible avec différents types d'ensemble de couleurs, et la structure des machines *B* produites par la transformation permet la preuve automatique intégrale par l'Atelier B.

L'ensemble de ces travaux , chacun à leur niveau, contribuent à renforcer l'efficacité d'un cadre global d'analyse sécuritaire.

Mots clés : système d'enclenchement ferroviaire, systèmes à événements discrets, RdPC, méthode de modélisation

---

IFSTTAR, COSYS/ESTAS
20, rue Élisée Reclus – BP 70317 – F-59666 Villeneuve d'Ascq Cedex – France