Eur. Transp. Res. Rev. (2017) 9: 13
DOI 10.1007/s12544-017-0228-x

ORIGINAL PAPER

A CPN/B method transformation framework for railway safety rules formal validation

Zakaryae Boudi[1] · **Rahma Ben-Ayed**[2] · **El Miloudi El Koursi**[2] · **Simon Collart-Dutilleul**[2] · **Thomas Nolasco**[3] · **Mohamed Haloua**[1]

Received: 12 January 2015 / Accepted: 10 March 2017 / Published online: 21 March 2017

Abstract This paper presents a "CPN/B method" based process for railway systems safety analysis. Achieving interoperability through the European Rail Traffic Management System (ERTMS/ETCS) is facing difficulties in railway safety assessment due to the interaction of national and European operating specifications. These specifications have been modeled using several formalisms, which makes it is extremely hard to preserve all requirements when switching between different formalisms. However, this problem, crucial for efficient progress in railway safety research, has received very little attention in the literature. In this respect, the purpose of this contribution is to provide a methodology to demonstrate safety in railway systems by converting CPN models, widely used in modeling, into B abstract machines. It aims at enabling a stronger combination of formal design techniques and analysis tools able to cope with the real complexity of systems and automatically prove that safety properties are unambiguous, consistent and not contradictory, considering an industrial railway context.

Keywords Colored Petri nets · B method · Transformation methodology · Railway safety · ERTMS

✉ Zakaryae Boudi
Zakaryae.boudi@gmail.com

Rahma Ben-Ayed
yengui.rahma@gmail.com

El Miloudi El Koursi
el-miloudi.el-koursi@ifsttar.fr

Simon Collart-Dutilleul
simon.collart-dutilleul@ifsttar.fr

Thomas Nolasco
Thomas.nolasco@clearsy.com

Mohamed Haloua
haloua@emi.ac.ma

[1] Ecole Mohammadia d'Ingénieurs, Med V University, Rabat, Morocco

[2] French Institute of Science and Technology for Transport, Development, and Networks, IFSTTAR-COSYS-ESTAS, 20, rue Elisée RECLUS BP 317, F-59666 Villeneuve d'Ascq Cedex, France

[3] ClearSy, 62, rue de la Chaussée d'Antin, 75009 Paris, France

1 Introduction

Today, Europe has more than twenty signaling and train control systems for railway transport that are different in terms of performance and safety. Therefore, rail interoperability in Europe requires a gradual transition to a common system for the various member states, the European Rail Traffic Management System (ERTMS/ETCS).

Performing railway interoperability allowed by ERTMS requires a common understanding of requirements by all involved parties. In this context, formal rigorous models are effective tools to identify and clarify ambiguities. The PERFECT (Performing Enhanced Rail Formal Engineering Constraints Traceability) project is a French scientific project that aims to formalize railway specifications and validate various systems. Systems' models will be developed and formally analyzed to determine, when possible, the compliance between ERTMS and National railway specifications. Our work focuses on modeling of the functional and safety aspects of railway systems (ERTMS, national rules, human factor…) and their validation by the B formal method. Several studies have been conducted in this context using formal methods and techniques. We can quote the project Open-ETCS led by a European consortium.

In this regard, the transportation technology and electrification growing influence on performance and safety of systems operations, especially railways, requires from safety research, such as PERFECT research, to address the more than ever complex task of real scenario modeling. That is why railway safety systems have been modeled using different formalisms, such as Petri nets, UML and others [1–6], in order to facilitate the expression of the know-how of industrial experts who may not be familiar with mathematical formalisms. Yet, in order to link and analyze all railway models, B machines will constitute a common destination point that allows us to arrive at rigorous conclusions. One of the ultimate goals of this study is to provide a conversion methodology capable of giving B abstract machine which behave exactly like Colored Petri Nets models, and, therefore, ensure the preservation of initial expert modeling and assessment.

This paper will present at first, High Level Petri Nets and their interest in railway modeling. Afterwards, the B method and its tools will be shown before presenting the core of this contribution, which is a concrete translation method from Colored Petri Nets to B abstract machines. As the purpose of this work is to introduce the transformation framework and the way it could be used within safety rules validation processes, the next sections will detail, on the one hand, a simple railway case study in which Petri nets are used in modeling before their transformation to B abstract machines and safety analysis, and on the other hand, describe how addressing an ERTMS RBC procedure could benefit from such a translation methodology. Finally, last section sums up and shows some interesting perspectives to this work.

2 High level Petri nets modeling

Petri Nets were developed in 1960–1962 by the German mathematician and computer scientist Carl Adam Petri. They have become famous thanks to the work of American researchers who used them in the 1970s (MIT Project on Mathematics and Computation-MAC project).

These networks constitute a powerful tool for studying a wide variety of discrete event systems. They are useful for both static modeling through their structure and dynamic representation through their operating rules. Therefore, they cannot only study the architecture of systems, but also their evolution and their reaction during the simulation.

2.1 Place/transition Petri nets

A Petri Net is represented by a graph with two types of nodes: places, represented graphically by circles, and transitions that are represented by bars or boxes. Places and transitions are connected by directed arcs: arcs can only link a place to a transition or a transition to a place. A Petri Net is characterized by its initial state called initial marking. Detailed explanation is given in [7].

Using Place/transition Petri Nets to model big complex systems, such as railway systems, is very limited, due to the size and complexity of the obtained models. For this reason, it is necessary to use High-Level Petri Nets.

2.2 High level Petri nets: Colored Petri nets

In elementary Petri Nets, tokens cannot be distinguished. However, modeling real complex systems requires the possibility of transforming the nature of tokens through a transition. Thereby, a new type of Petri Nets handling tokens transformation and labeled by a first-order language was born, called High-Level Petri Nets. The first interesting class of High Level Petri Nets was the "predicate/transition" nets developed by Hartmann Genrich [8]. The next step forward was achieved by the development of Algebraic Petri Nets [9], and later the development of Colored Petri Nets introduced by Kurt Jensen [10]. In the PERFECT project, Colored Petri Nets are considered rather than the other High-Level Petri Nets forms especially for their bigger modeling power in the case of complex railway systems.

Colored Petri Nets are an extension of Petri Nets based on a functional language where the notion of typing is fundamental. They allow the association with a value named color to differentiate tokens. Color of tokens can be transformed or tested during their passage in arcs and transitions. The formal definition of a Colored Petri Net is given below [10].

Definition 1 A Colored Petri Net is a tuple $CPN = (\Sigma, P, T, A, N, C, G, E, I)$ satisfying the following requirements:

(i) $\boldsymbol{\Sigma}$ is a finite set of non-empty types, called colour sets.
(ii) $\boldsymbol{P}$ is a finite set of places.
(iii) $\boldsymbol{T}$ is a finite set of transitions.
(iv) $\boldsymbol{A}$ is a finite set of arcs such that: $P \cap T = P \cap A = T \cap A = \emptyset$.
(v) $\boldsymbol{N}$ is a node function. It is defined from A into $P \times T \cup T \times P$.
(vi) $\boldsymbol{C}$ is a colour function. It is defined from P into Σ.
(vii) $\boldsymbol{G}$ is a guard function. It is defined from T into expressions such that: $\forall t \in T : [Type\ (G(t)) = Bool \wedge Type\ (Var\ (G(t))) \subseteq \Sigma]$.
(viii) $\boldsymbol{E}$ is an arc expression function. It is defined from A into expressions such that:

$$\forall a \in A : \left[\text{Type}\ (E(a)) = C(p(a))_{MS} \wedge\ \text{Type}(Var\ (E(a))) \subseteq \Sigma\right]$$

Where $p(a)$ is the place of $N(a)$.

(ix) ***I*** is an initialization function. It is defined from P into closed expressions such that:

$$\forall p \in P : \left[\text{Type}\,(I(p)) = C(p)_{MS}\right].$$

To have more details on the above definition, the reader can refer to [10].

In this context some typical cases are modeled using the CPN-tools software platform. The main architects behind the tool are Kurt Jensen, Soren Christensen, Lars M. Kristensen, and Michael Westergaard [11, 12]. This platform combines Petri Nets with the functional programming language Standard ML. Standard ML provides the possibilities of defining data types and describing data manipulation. This software has been adopted by the PERFECT project, especially for its extensive use in the previous contribution works in rail systems assessment [13], and for its perfect match with Colored Petri Nets requirements. A parallel modeling work within PERFECT, using this tool, is focused on railway scenario CPN modeling including infrastructure, regulation, interlocking and human factors. Safety properties within these models are intended to be verified using the methodology of the present paper. Figure 1 illustrates an example of a Colored Petri Net model for a simple railway movement from one track circuit to the next.

Later in this paper, an academic railway example of trains' safe movements is introduced and modeled using the CPN-tool. Then, simulating the exact coherence between the CPN model and the obtained B abstract machine will be shown.

3 B method and tools

The B method is a formal method for developing secure software. It was designed by Jean-Raymond Abrial [14]. The B method provides a formal specification and rigorous analysis of the functionality and behavior of a system. It covers all phases of the life cycle of software development from specification to implementation. Industrial applications of the B method are mainly destined to rail systems, including the commissioning of the metro line 14 (METEOR, 1998) and the automation of line 1 (2005) by the RATP and other developments autopilots subways.

The development of a B method project has two interrelated activities: writing formal texts and the proof of these texts. It is based on first-order logic mathematical notations and set theory where the system state is modeled by abstract types of preset data. It allows the modeling of static and dynamic aspects of structured software in abstract machines. The static aspect is characterized by sets, constants, properties, variables and constants, while the dynamic aspect is described by the initialization and operations. In a specification based on the model concept, the state of the modeled system is described by the set of pairs (predicate/expressions) where all predicates model the static aspect of the system. Description of state changes models the dynamic aspect. These state changes are described using three characteristic features:

- Pre-condition: defined by the set of states from which the state change is possible.
- Operation: consists of the list of changes made to the pairs (predicate expression).
- Post-condition characterizes acceptable statements as results of change.

Sometimes, in the formalism of the method B, the notion of "substitution" replaces the concepts of pre-and post-condition.

3.1 Mathematical notation and B abstract machines

Data modeling and their properties, with the B language based on mathematical notation, is essentially based on theory of sets. Nevertheless, the B theory of sets includes the notion of typing. In other words, all the elements of a set have the same type. B properties are expressed by formulas of first order predicate calculus. That is to say, in addition to the predicates constructed with conventional propositional operators (and (∧) or (∨) ...), there is equality and predicates composed of existentially quantified variables (∃) and universally quantified ones (∀).

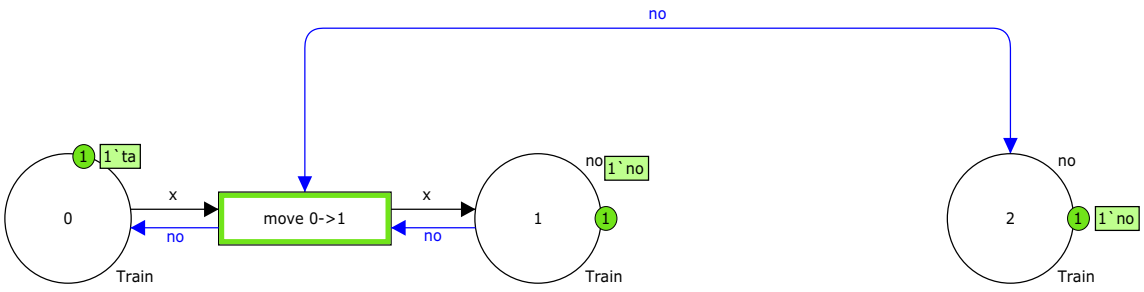


Fig. 1 Petri net model for a simple railway movement

The B method is based on the notion of abstract machines. These abstract machines can be associated to imperative programing, which describes the operations in terms of sequences of elementary instructions executed to change the program state. Each machine declares its own variables and operations and variables can only be changed by the machine's operations.

The abstract machine models a system described by a set of data or variables and operations that change their states. Figure 2 shows a generic abstract machine.

In this paragraph, each clause of an abstract machine is detailed:

MACHINE: Declaration of the name of the abstract machine and the formal parameter list (sets or values).
CONSTRAINTS: Declaration of logical properties of parameters (sets or values) of the machine.
SETS: Declaration of the definitions of machine's abstract or listed sets.
CONSTANTS: Declaration of identifiers of constants.
PROPERTIES: Declaration of logical properties of sets and constants declared with typing and evaluation of constants.
VARIABLES: Declaration of identifiers of variables.
INVARIANT: Declaration of the invariant logical properties of variables declared with typing of variables.
ASSERTIONS: Declaration of definitions of the machine's new logical assertions.
INITIALISATION: Declaration of the generalized substitution initializing the machine variables.
OPERATIONS: Declaration of machine operations in the form of a header and a body.
END: End of the machine definition.

```
MACHINE Machine
CONSTRAINTS
C
SETS
St
CONSTANTS
k
PROPERTIES
B
VARIABLES
v
DEFINITIONS
D
INVARIANT
I
ASSERTIONS
A
INITIALISATION
T
OPERATIONS
Op_y=
          PRE P
          THEN S
          END
END
```

Fig. 2 Generic B abstract machine

3.2 "ProB" and "Atelier B" tools

One interesting point of this study is the use of automatic proof and check tools of the B method, especially to visualize the behavior of B abstract machines and check safety properties of the modeled railway problem. The tools used in this context are ProB[1] animator and model checker[1], added to the "Atelier B"[2] prover. ProB is an animator and model checker for the B-Method that allows fully automatic animation of many B specifications, and can be used to systematically check a specification for a range of errors. The constraint-solving capabilities of ProB can also be used for model finding, deadlock checking and test-case generation. ProB is now being used within Siemens, Alstom, and several other companies for data validation of complicated properties.

Model checking within ProB does the verification of a system model with respect to the properties that are expected on this model. The result of this analysis is the confirmation that each property is, or is not, maintained by the model. In the latter case, and this is one of the main interests of this tool, the model checker returns a counter-example of how the property is not maintained.

Otherwise, developed by Clearsy, "Atelier B" is an industrial tool that allows for the operational use of the B Method to develop defect-free proven software. It is used to develop safety automatisms for the various subways installed throughout the world by Alstom and Siemens, and also for Common Criteria certification and the development of system models by ATMEL and STMicroelectronics. Additionally, it has been used in a number of other sectors, such as the automotive industry, to model operational principles for the onboard electronics of three car models.

Atelier B provides a theorem prover for the B models. Unlike model checking, theorem provers are not based on finite and decidable systems. They are mostly less simple to use than the model checker but used to address various problems where, for example, the model checker cannot be used. They allow us to bring proof of correctness of the program or system to be modeled.

[1] http://www.stups.uni-duesseldorf.de/ProB/index.php5/Download
[2] http://www.atelierb.eu/telecharger-latelier-b/

Accordingly, these two tools are complementary where the use of ProB suits better in the early conception phases, especially for a fast error debug, and the use of Atelier B comes later to give the proof of properties correctness. Therefore, the use of both of these tools reinforces the conclusions of the work presented in this paper.

4 Description of the transformation methodology

This section describes the proposed conversion algorithm to translate CPN models to B abstract machines. First of all, a general survey of previous contributions will be presented to point out the difficulties and limitations of the establishment of a complete and applicable transformation method. Then, the translation approach consists in defining, firstly, the transformation rules of a CPN's generic structural properties. Yet, the conversion of CPN model to B language will be complemented by the integration of the behavioral specification and dynamic properties.

4.1 Known transformation approaches and literature survey

According to [15], formalism transformation could be done under three main approaches: **mapping approach**, **pivot approach** and the **layered approach**. Mapping approach states that for each pair of formalisms a mapping is created which transforms expressions in the source formalism to expressions in the target formalism. It can be well adapted to the two specific formalisms and, therefore, involves the smallest loss of information. Concerning the pivot approach, transformation between two different formalisms is done via a chosen formalism called the pivot. The pivot formalism must be very expressive to avoid losing information in the transformation, especially if the system involves formalisms that are unlike each other. A pivot approach was conducted in [16] for Petri net transformation into DEVS formalism. However this method concerns only elementary place/transition nets because of their small amount of information. The layered approach uses a layered architecture containing languages with increasing expressiveness. This approach has been proposed in order to avoid using a very expressive language and to ensure tractable reasoning with the integrated languages. In such a layered architecture, representations can be translated into languages higher in the hierarchy without loss of information.

The method proposed in this paper is based on informed transformation [17], which is in accordance with the mapping approach in the scope of Model Driven Engineering [18]. Informed transformation stipulates that a mapping of the source formal description of a formalism to the target one exists. The transformation between formalisms uses this formal description and the mapping between them. Within Model Driven Engineering, input Petri net models conform to the metamodel presented in [19], which is based on Jensen's formal definition.

A recent previous work with the same goal and using a mapping approach is presented by BON.P in [20]. However, that conversion method was tested and it was found that the resulting machines are not accepted by the B tools due to the use of a large amount of definitions, added to their unpractical heaviness. Indeed, this method tries to define both the structural and behavioral properties of CPNs thanks to heavy definitions, and copes with these CPN modeling as a rigid graphical formalism. Consequently, the obtained machines are very hard to express or simulate and cannot be verified using B tools.

Another recent and important work to refer to is the work presented in [21], where a very close issue was considered, but this time for a mapping from Place/Transition Petri Nets to the B-language. This work constitutes a simplified version of the author's original mapping form Evaluative Petri Nets to B machines. However this paper's mapping does not cover Colored Petri Nets, the ideas presented are significantly interesting for our transformation framework in order to provide a future formal description of the present methodology.

Therefore, this paper presents and describes the framework of a new transformation method, based on a deeper analysis of the functioning and the handling of Colored Petri Nets with special focus on their component tasks. One of the basic starting principles is the ability of applying the method to a complete safety study case, as it is presented in this paper. The methodology is also likely to lead to an elaborate formal description of the transformation and open to undergo improvements. Furthermore, this new method intends to provide a transformation able to be encoded in order to develop an interactive software platform for automatic CPN conversion into B abstract machines. So, this work describes in detail the principles of the transformation and its application and will constitute the base framework for the establishment of formal representations and proof of the transformation.

The next sections detail the transformation theory, starting with basic Multisets definition, structural properties transformation rules, and finally, the description of CPN behavioral characteristics transformation according to the spirit of the B method. Afterward, the B machines behavior equivalence to the original CPN models is demonstrated, which constitute a first step toward a formal proof of the correctness of the method. The simulation of a typical railway example, which validates the syntax acceptance, supports this transformation framework and visualizes the exact matching of the machine's behavior and proves the safety invariant conservation.

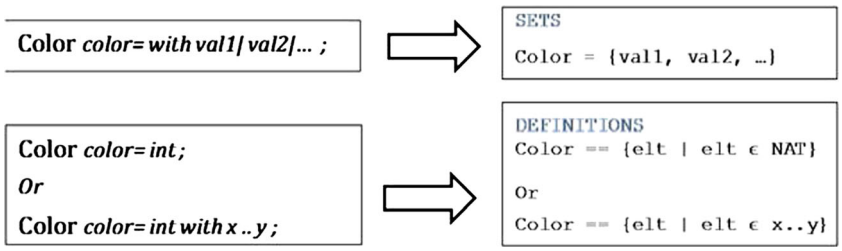


Fig. 3 Colours B translation

4.2 Multisets definition

A multiset is specified as a relationship between the base set of the multiset and natural numbers. As in [8, 13], the elements of a multiset are pairs ($ee \mapsto nn$) where ***ee*** is a member of the base set and ***nn*** is an integer which represents the coefficient (number of occurrences) of the element in the multiset.

Ms (ss) is a multiset based on ***ss*** defined as a total function from ***ss*** to all natural numbers:

$$Ms\ (ss) == ss \rightarrow NAT.$$

The empty multiset, based on ***ss***, is composed of pairs of elements of the support set related to 0. Thus, it is a total function, which for each element of the starting set combines the integer 0:

$$\text{Ms\_empty}\ (\text{ss}) == \{\text{elt} \mid \text{elt} : \text{ss} \times \{0\}\}$$

This work uses only these two Multiset definitions that will appear in all the machines obtained for each Petri net color. The following sections will detail the transformation rules.

4.3 Petri net structure transformation rules

For several reasons of formalization as the transformation and its automation, the construction of the transformation rules was based on Jensen's formal definition of Colored Petri Nets. For each point of the formal definition of Petri Net structure, an associated transformation rule is then given. The following paragraph details these rules.

(i) Σ is a finite set of non-empty types, called colour sets.

A set of the abstract machine is associated to each color of Σ. If the color is described by an enumerated set, it matches to an enumerated set in the clause "SETS" of the abstract machine B. Regarding non-enumerated sets; they correspond to definitions of the clause "DEFINITIONS" as shown in Fig. 3.

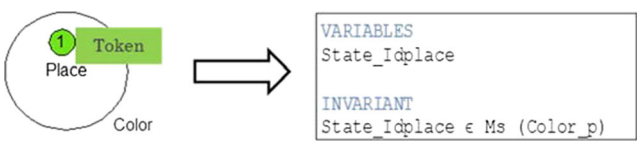


Fig. 4 Places B translation

(ii) P **is a finite set of places. (iii)** T **is a finite set of transitions. (iv)** A **is a finite set of arcs such that:** $P \cap T = P \cap A = T \cap A = \emptyset$**. (v)** N **is a node function. It is defined from** A **into** $P \times T \cup T \times P$**. (vi)** C **is a colour function. It is defined from** P **into** Σ**.**

The places of the set P are defined by their identifier and their state (marking). Therefore, they are translated by the variables ***state_Idplace***. Places are also defined by their type ***color***. This invariant property will be translated by the invariant ***state_Idplace ∈ Ms. (color)*** in the "INVARIANT" clause (Fig. 4).

A transition is defined by its identifier as well as its state (enabled or non-enabled). The structure of a transition is translated into a Boolean variable that expresses the transition's state. Of course, the name of the variable describes the identifier, and the invariant in B describes the type. Elements of transition guard are handled in the section describing Petri Net behavior transformation (Fig. 5).

Note: Arcs do not explicitly appear in the abstract machine. In fact, the goal of the translation is to provide a B abstract machine behaving exactly as the considered Petri Net. For this reason, arcs are analyzed through their function in a Petri Net which is to ensure the logical evolution of states, using expressions of arcs. Therefore, the information of arcs is not lost in the transformation process, but it will appear in the dynamic part of the machine which is covered in the section describing the behavioral transformation of a Colored Petri Net.

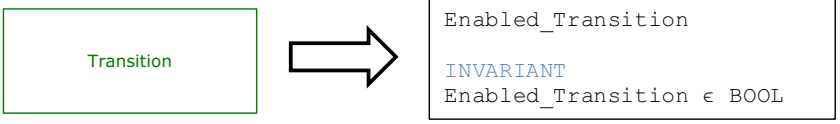
(ix) I **is an initialization function. It is defined from** P **into closed expressions such that:** $\forall p \in P : [Type\ (I(p)) = C(p)_{MS}]$**.**

The initial marking of each site is assigned to the INITIALISATION clause as is shown in Fig. 6.

Elt_color is an element of the set ***color*** and ***n*** is a natural number which expresses the occurrence of the token, according to the previous definition of the multiset ***Ms_empty***.

Note: The clauses "VARIABLES" and "INITIALISATION" will also include the declaration and

Fig. 5 Transitions B translation



the initialization of variables of type "NATURAL" describing the occurrence of each token color of a given place, denoted ***occ_eltcolor_idplace***. These variables represent only the state of the places (marking) of the Petri Net. They will mainly allow expression properties invariants and help the machines automatic proof of the Atelier B.

4.4 Petri net behavior transformation rules

The behavior of a Petri Net corresponds to the evolution of its places states (markings). This evolution is governed by the crossing of enabled transitions, following the instructions of expressions of arcs and guards.

Of course, a transition is enabled if and only if all incoming places (with incoming arcs) have a number of tokens with the type indicated on the incoming arcs and if the predicate of the guard is true. The definition of a transition state (enabled or non-enabled) will be translated by an operation ***Op_Enabled_Idtransion***. This will aim to change the Boolean variable describing the state of the transition ***Enabled_Idtransion***. Regarding transition crossing (firing), it consists in transforming the states of adjacent places, according to the guidelines of the guard and outgoing arcs to outgoing places, and subtracting consumed tokens in incoming places. Crossing a transition will be translated by a second operation ***Op_Fired_Idtransion***.

Note: The behavioral transformation is characterized by the introduction of two operations for each transition, when the use of only one operation ***(op_Fired_Idtransition)*** might be absolutely sufficient in most cases. The main idea behind the operation ***Op_Enabled_Idtransion*** is to leave open the use of the Boolean variables ***enabled_Idtransition*** in expressing safety invariants related to transition or events. In fact, sometimes, it is difficult to establish invariants using variables corresponding to places or tokens. If the transformation is used in critical software development for example, this operation could be omitted or removed in the refinement phases.

Therefore, the evolution of Colored Petri Nets will be translated into B through operations in the kind of substitutions with precondition "***PRE***" and "***SELECT***" substitutions that will represent the evolution of places and transitions states which are already declared as variables. The use of "***SELECT***" in the ***op_Fired_Idtransition*** is justified by the fact that the firing of the transition may involve different tokens (colors) choices according to the elements of the "color set" related to the incoming and outgoing places (all tokens in those places are necessarily one of these elements, i.e. *token* ∈ X where X ∈ Σ).

(vii) *G* is a guard function. It is defined from *T* into expressions such that: $\forall\ t \in T : [Type(G(t)) = Bool \wedge Type(Var(G(t))) \subseteq \Sigma]$. **(viii) *E* is an arc expression function. It is defined from *A* into expressions such that:** $\forall a \in A : [Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$ **where *p*(*a*) is the place of N(a).**

The predicate part of the guard expressions and incoming arcs expressions will be translated by a predicate in the ***PRE*** section of the substitution with precondition in the operation ***Op_Enabled_Idtransion***. The ***THEN*** section will include the action on Boolean variable ***Enabled_Idtransion:=TRUE*** (Fig. 7).

On the other hand, the guard may also have an action part. Expressions of outgoing arcs can be considered as actions as well. These actions (guards and outgoing arcs) aim to change the state of outgoing places, thus changing the state of the corresponding variables in B abstract machine, and possibly other variables according to the Petri Net. They will be translated in the "***THEN"*** section of the substitution with selection in ***Op_Fired_Idtransion*** operation. The change of variables corresponding to states of places is done by an overload in the B machine. This part will always contain the substitution ***Op_Enabled_Idtransion:= FALSE*** so that ***Op_Enabled_Idtransion*** operation is no longer enabled. The "***SELECT"*** (and ***WHEN***) part of this second operation will include the following predicate: ***Enabled_Idtransion = TRUE & Predicate_P***. the ***Predicate_P*** gives an indication of the selected token and the decrement condition of occurrences variables (Fig. 8).

At this point, the Colored Petri Net conversion methodology into B abstract machines was described. The following sections will emphasizes the behavioral equivalence of the input CPN models and the obtained B

Fig. 6 Initial marking translation

```
INITIALISATION
State_Idplace:= Ms_empty(color) <+ {(elt_color |-> n),...}
```

```
Op_Enabled_Idtransition=

     PRE Predicate
     THEN Enabled_idtransition:= TRUE
     END
```

Fig. 7 Enabled transition Operation in the B machine

machines, before showing the use of this method through an academic railway example.

Note: the predicates in the obtained B machine could be enriched in order to facilitate the proving task without affecting the behavior of the machines or the quality of the transformation.

4.5 Behavioral equivalence

According to Jensen's formal definition of the behavior of a Colored Petri Net:

(1) **A transition is enabled ⇔ there are enough tokens of the correct values specified on arcs on each input-place and the guard evaluates to true.**
(2) **When a transition is fired ⇔ a multi-set of tokens is removed from each input-place and a multi-set of tokens is added to each output-place.**

To ensure that the transformation keeps the elements of the initial model, these two basic equivalences have to be verified in terms of their element's corresponding components in the B abstract machine:

(1) An operation ***Op_Enabled_Idtransion*** is enabled ⇔ the predicate of the clause PRE in ***Op_Enabled_Idtransion*** is true.
(2) An operation ***Op_Fired_Idtransion*** is executed ⇔ the substitution in the clause THEN in ***Op_Fired_Idtransion*** contains a multisets overload and is executed.

The demonstration of these two equivalences, presented hereafter, is direct and can be done using the basic B abstract machine definitions. In this work part of the PERFECT project, such a demonstration supports the correctness of the transformation and provides materials for the construction of its formal proof.

```
Op_Fired_Idtransition=

     SELECT Enabled_idtransition= TRUE & Predicate1
     THEN Substituions 1
     WHEN Enabled_idtransition= TRUE & Predicate2
     THEN Substituions 2
     …
     END
```

Fig. 8 Firing transition translation

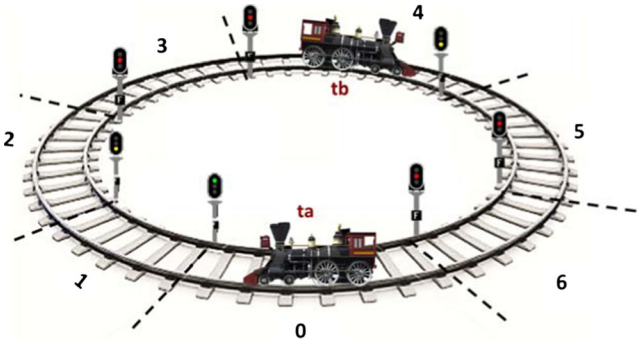


Fig. 9 Schematic representation of the study case

Demonstration 1 For the first equivalence, let us suppose that: An operation ***Op_Enabled_Idtransion*** is enabled and let us prove that the predicate of the clause ***PRE*** in ***Op_Enabled_Idtransion*** is true (direct implying): by construction, the substitutions with precondition operation (containing "***PRE***" and "***THEN***") is enabled if and only if the substitution after ***PRE*** is true. So, the predicate of the clause ***PRE*** in ***Op_Enabled_Idtransion*** is true.

Let us suppose now that the predicate of the clause PRE in ***Op_Enabled_Idtransion*** is true and let us prove that the operation ***Op_Enabled_Idtransion*** is enabled (reverse implying): since the predicate of the operation clause ***PRE*** is true, the ***THEN*** substitution can be executed, and that means, by construction, that the operation ***Op_Enabled_Idtransion*** is enabled.

Demonstration 2 For the second equivalence, let us suppose that: an operation ***Op_Fired_Idtransion*** is executed and let us prove that the substitution in the clause ***THEN*** in ***Op_Fired_Idtransion*** contains a multisets overload and is executed (direct implying): the transformation method specifies that the change of variables corresponding to states of places is done by an overload in the B machine. So this implication is proved directly.

Let us suppose now that the substitution in the clause ***THEN*** in ***Op_Fired_Idtransion*** contains a multisets overload and is executed and let us prove that operation ***Op_Fired_Idtransion*** is executed (reverse implying): by definition, an operation is executed if and only if the ***THEN*** substitution of this operation is executed. Therefore, this implication is directly proved.

5 Railway study case

As well as the rigorous construction of the transformation from CPNs to B machines in the context of railway safety,

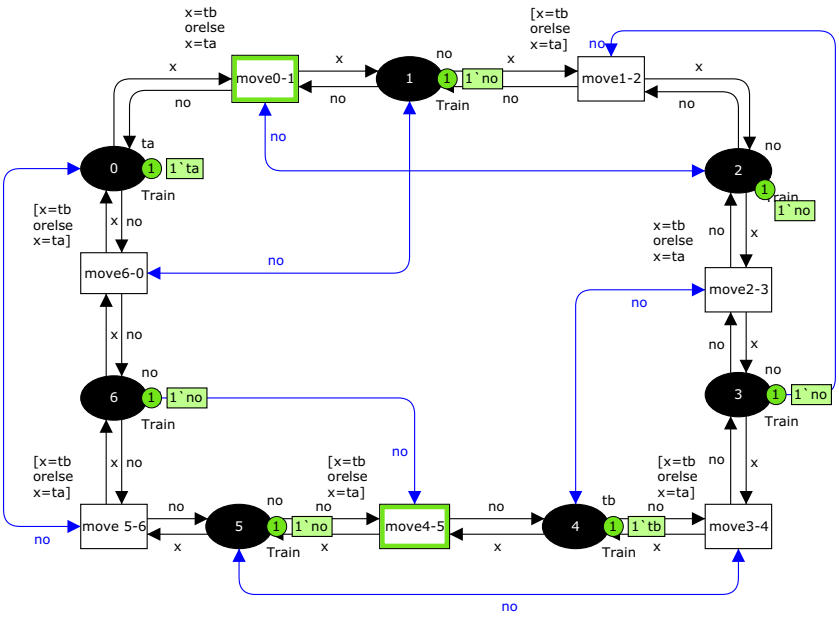


Fig. 10 Study case Petri net model

its applicability of the validation process is crucial and extremely important, especially for the PERFECT project results requirements. That is why, in parallel to the development of an elaborate and formal transformation, based on the framework presented here and comprising the description of transformation rules, the early application on

```
MACHINE
RdP_BOUDI_Complet
SETS
train = {ta,tb,no}
VARIABLES
occ_ta_0, occ_ta_1, occ_ta_2,occ_ta_3, occ_ta_4,
occ_ta_5,occ_ta_6,
occ_tb_0, occ_tb_1, occ_tb_2,occ_tb_3, occ_tb_4,
occ_tb_5,occ_tb_6,
occ_no_0, occ_no_1, occ_no_2,occ_no_3, occ_no_4,
occ_no_5,occ_no_6,
enabled_from0to1,enabled_from1to2,enabled_from2to3,enabled_fro
m3to4,
enabled_from4to5,enabled_from5to6,enabled_from6to0,
state_0,state_1,state_2,state_3, state_4,state_5,state_6
INVARIANT
occ_ta_0:NATURAL & occ_ta_1:NATURAL & occ_ta_2:NATURAL &
occ_ta_3:NATURAL & occ_ta_4:NATURAL &
occ_ta_5:NATURAL & occ_ta_6:NATURAL &
occ_tb_0:NATURAL & occ_tb_1:NATURAL & occ_tb_2:NATURAL &
occ_tb_3:NATURAL & occ_tb_4:NATURAL &
occ_tb_5:NATURAL & occ_tb_6:NATURAL &
occ_no_0:NATURAL & occ_no_1:NATURAL & occ_no_2:NATURAL &
occ_no_3:NATURAL & occ_no_4:NATURAL &
occ_no_5:NATURAL & occ_no_6:NATURAL &
enabled_from0to1 : BOOL & enabled_from1to2 : BOOL &
enabled_from2to3 : BOOL &
enabled_from3to4 : BOOL & enabled_from4to5 : BOOL &
enabled_from5to6 : BOOL &
enabled_from6to0 : BOOL &
state_0:Ms(train)& state_1:Ms(train)& state_2:Ms(train)&
state_3:Ms(train)& state_4:Ms(train)& state_5:Ms(train)&
state_6:Ms(train)
DEFINITIONS
Ms(ss)== ss --> NAT;
Ms_empty(ss)=={elt|elt : ss×{0}}
INITIALISATION
occ_ta_0:=1 || occ_ta_1:=0 || occ_ta_2:=0 || occ_ta_3:=0
||occ_ta_4:=0 || occ_ta_5:=0 || occ_ta_6:=0
||occ_tb_0:=0 || occ_tb_1:=0 || occ_tb_2:=0 || occ_tb_3:=0
||occ_tb_4:=1 || occ_tb_5:=0 || occ_tb_6:=0
||occ_no_0:=0 || occ_no_1:=1 || occ_no_2:=1 || occ_no_3:=1
||occ_no_4:=0 || occ_no_5:=1 || occ_no_6:=1
||enabled_from0to1:=FALSE ||enabled_from1to2:=FALSE
||enabled_from2to3:=FALSE
||enabled_from3to4:=FALSE ||enabled_from4to5:=FALSE
||enabled_from5to6:=FALSE
||enabled_from6to0:=FALSE
||state_0:=Ms_empty(train)<+ {(ta |->1)}
||state_1:=Ms_empty(train)<+ {(no |->1)}
||state_2:=Ms_empty(train)<+ {(no |->1)}
```

Fig. 11 Extract of the first part of the obtained B machine

```
OPERATIONS
Op_Enabled_from0to1=
PRE ((ta |-> 1):state_0 or (tb |-> 1):state_0) &
(no |-> 1):state_1 & (no |-> 1):state_2
THEN enabled_from0to1 := TRUE
END;
Op_Enabled_from1to2=
PRE ((ta |-> 1):state_1 or (tb |-> 1):state_1) &
(no |-> 1):state_2 & (no |-> 1):state_3
THEN enabled_from1to2:= TRUE
END;
Op_Enabled_from2to3=
PRE ((ta |-> 1):state_2 or (tb |-> 1):state_2) &
(no |-> 1):state_3 & (no |-> 1):state_4
THEN enabled_from2to3:= TRUE
END;
Op_Enabled_from3to4=
PRE ((ta |-> 1):state_3 or (tb |-> 1):state_3) &
(no |-> 1):state_4 & (no |-> 1):state_5
THEN enabled_from3to4:= TRUE
END;
Op_Enabled_from4to5=
PRE ((ta |-> 1):state_4 or (tb |-> 1):state_4) &
(no |-> 1):state_5 & (no |-> 1):state_6
THEN enabled_from4to5:= TRUE
END;
Op_Enabled_from5to6=
PRE ((ta |-> 1):state_5 or (tb |-> 1):state_5) &
(no |-> 1):state_6 & (no |-> 1):state_0
THEN enabled_from5to6:= TRUE
END;
Op_Enabled_from6to0=
PRE ((ta |-> 1):state_6 or (tb |-> 1):state_6) &
(no |-> 1):state_0 & (no |-> 1):state_1
THEN enabled_from6to0:= TRUE
END;
Op_Fired_from0to1=
SELECT enabled_from0to1 = TRUE & (ta |-> 1):state_0 &
occ_ta_0>=1 & occ_no_1>=1
THEN state_0 := state_0 <+ {(ta |->(0)),(no |->(1))}
||state_1 := state_1 <+ {(ta |->(1)),(no |->(0))}
||enabled_from0to1:= FALSE
||occ_ta_0:=occ_ta_0-1 || occ_ta_1:=occ_ta_1+1 ||
occ_no_0:=occ_no_0+1 || occ_no_1:=occ_no_1-1
WHEN enabled_from0to1 = TRUE & (tb |-> 1):state_0 &
occ_tb_0>=1 & occ_no_1>=1
THEN state_0 := state_0 <+ {(tb |->(0)),(no |->(1))}
```

Fig. 12 Extract of the operation part of the obtained B machine

small case studies figures in the work agenda. In this section, a complete process is shown, from the Colored Petri Net modeling step to the validation of safety properties by B machine analysis, which is applied to a theoretical railway case study. The correctness of transformation is also checked using "ProB" and "CPN-tools" simulation tools.

5.1 Description of the railway case study

The studied example consists of a closed railway network composed of seven elementary portions numbered 0 to 6 including two trains *ta* and *tb* (Fig. 9). Train movements are specified by the following general rules (constraints):

- C1: the network is composed of consecutive elementary portions called track circuits (CdV).
- C2: trains run on that network in a given direction.
- The functioning of this e railway is governed by two other safety rules:
- C'1: there cannot be two trains on the same track circuit.
- C'2: there must always be a free track circuit between two trains.

5.2 Railway case study modeling using CPNs

The Colored Petri Net of the previous railway example is presented in the following Fig. 10. Each track circuit is modeled using a place of type "Train" where "Train" is a set containing the tokens *ta* and *tb*, and the train movements are represented by transitions.

Note that the initial marking is represented with the token "***ta***" in the place "***0***", "***tb***" in the place "***4***". The other places contain the token "***no***" which denotes the non-presence of a train in the track circuit modeled by the place.

5.3 The obtained B abstract machine after transformation

Following the transformation rules defined earlier, a B abstract machine is obtained for which the following Fig. 11 shows an extract of declaration and initialization parts.

An extract of the operations part is illustrated in Fig. 12.

5.4 Simulation of the exact correspondence between the obtained machine and the initial Petri net

Before the introduction of the safety invariants to validate and achieve the purpose of this study, a first check of the obtained B abstract machine is conducted by the "Atelier B" prover and the ProB model checking and animation platform. **The "Atelier B" prover has generated 174 proof obligations and has proved 174 one (a 100% rate)**. Furthermore, with the prover, if we demonstrate that the system has a rate of 100%, the typing of the machine is then correct. The ProB animation, added to this proof, will show that the obtained machine behaves correctly like the Petri Net model. For this purpose, this paper will compare the evolution of state spaces of the CPN model and the B machine for the same sequence of events.

At first, Fig. 13 illustrates the state space of CPN tools describing the initial marking and its reachable states.

This state corresponds to the ProB state space of the machine after its initialization presented in Fig. 14.

At this stage, since the paper cannot present all states, only the firing of the transition "move 0–1" et "move1–2", giving access to the state shown in Fig. 15. It is important to note that this comparison is done only for support to the transformation rules and demonstrations. A formal description and proof is to be constructed within the project. However, this illustration might help the reader to have a better understanding to the transformation basic rules.

After this firing, the only enabled transition is "move 4–5", and the places "0", "1", "2", "3", "4", "5" and "6" contain respectively the tokens "no", "no", "ta", "no", "tb", "no" and "no", as it is shown in Fig. 16 of CPN tools state space. The following figure of ProB animator shows that the same state is obtained after executing the operations corresponding to the two transitions (Fig. 17).

The simulation continues using the ProB animator until all the states are reached. This section visualized then that the

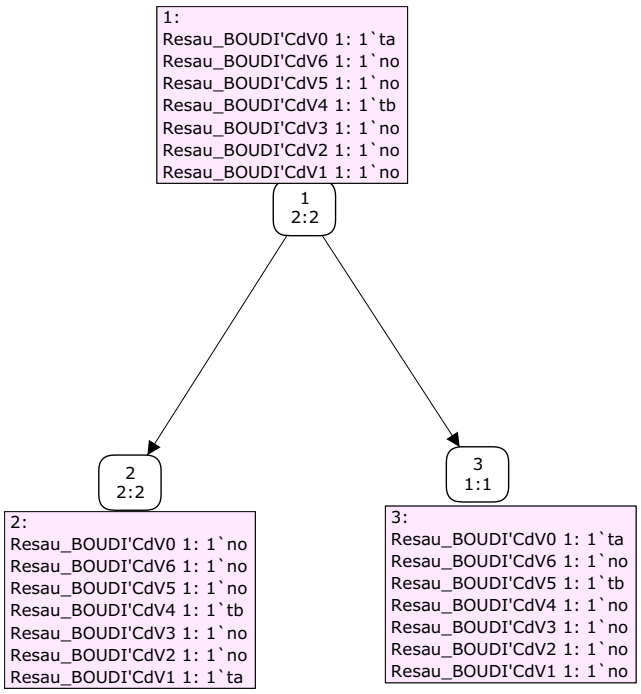


Fig. 13 CPN tools state space for the initial marking

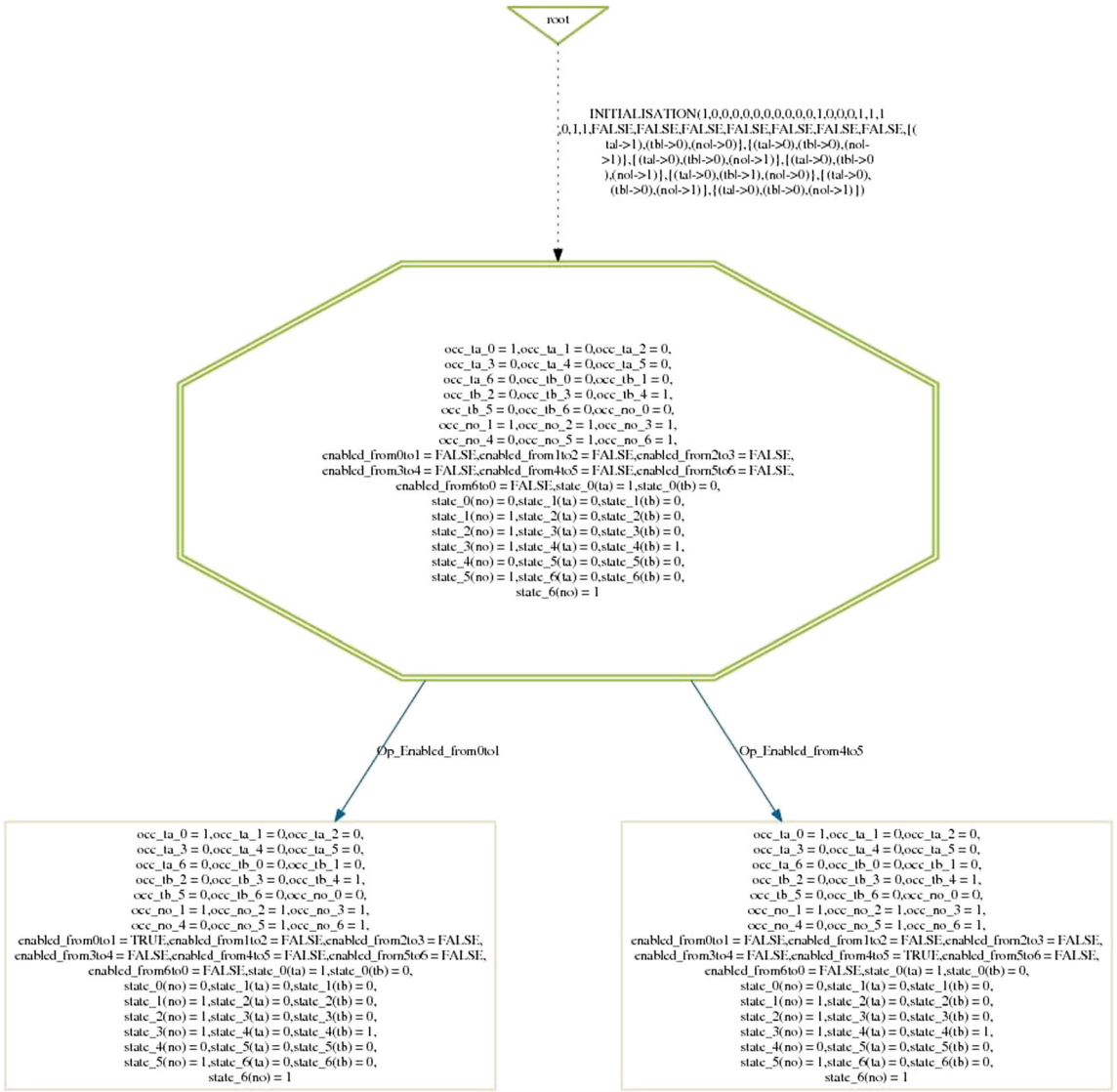


Fig. 14 ProB state space view at initialization

obtained abstract machine behaves exactly like the transformed Colored Petri Net and keeps all the modeled requirements, which is reinforcing our conclusions to use the transformation rules presented in this paper as a base framework for the construction of all the aspects of the transformation, such as its formalization, formal proof and automation. Note that in the industrial practice, the use of simulation tools and small applications during the early phases of a project is very important in order to ensure the right orientation of the efforts while progressing.

5.5 Safety constraints validation by B tools

In the case study, two safety rules are to be proved by B method tools: There cannot be two trains on the same track circuit and there must always be a free track circuit between two trains.

At first, it is necessary to express these safety rules by an invariant predicate. An invariant predicate which guarantees at the same time that only one train occupies a track circuit and that there is always a free track circuit between two trains is:

$$
\begin{aligned}
&(occ\_ta\_0 + occ\_tb\_0 + occ\_ta\_1 + occ\_tb\_1 <= 1)\&\\
&(occ\_ta\_1 + occ\_tb\_1 + occ\_ta\_2 + occ\_tb\_2 <= 1)\&\\
&(occ\_ta\_2 + occ\_tb\_2 + occ\_ta\_3 + occ\_tb\_3 <= 1)\&\\
&(occ\_ta\_3 + occ\_tb\_3 + occ\_ta\_4 + occ\_tb\_4 <= 1)\&\\
&(occ\_ta\_4 + occ\_tb\_4 + occ\_ta\_5 + occ\_tb\_5 <= 1)\&\\
&(occ\_ta\_5 + occ\_tb\_5 + occ\_ta\_6 + occ\_tb\_6 <= 1)\&\\
&(occ\_ta\_6 + occ\_tb\_6 + occ\_ta\_0 + occ\_tb\_0 <= 1)
\end{aligned}
$$

Thereby, the use of the model check of ProB which stops when all operations are covered proves that this invariant is respected by the abstract machine.

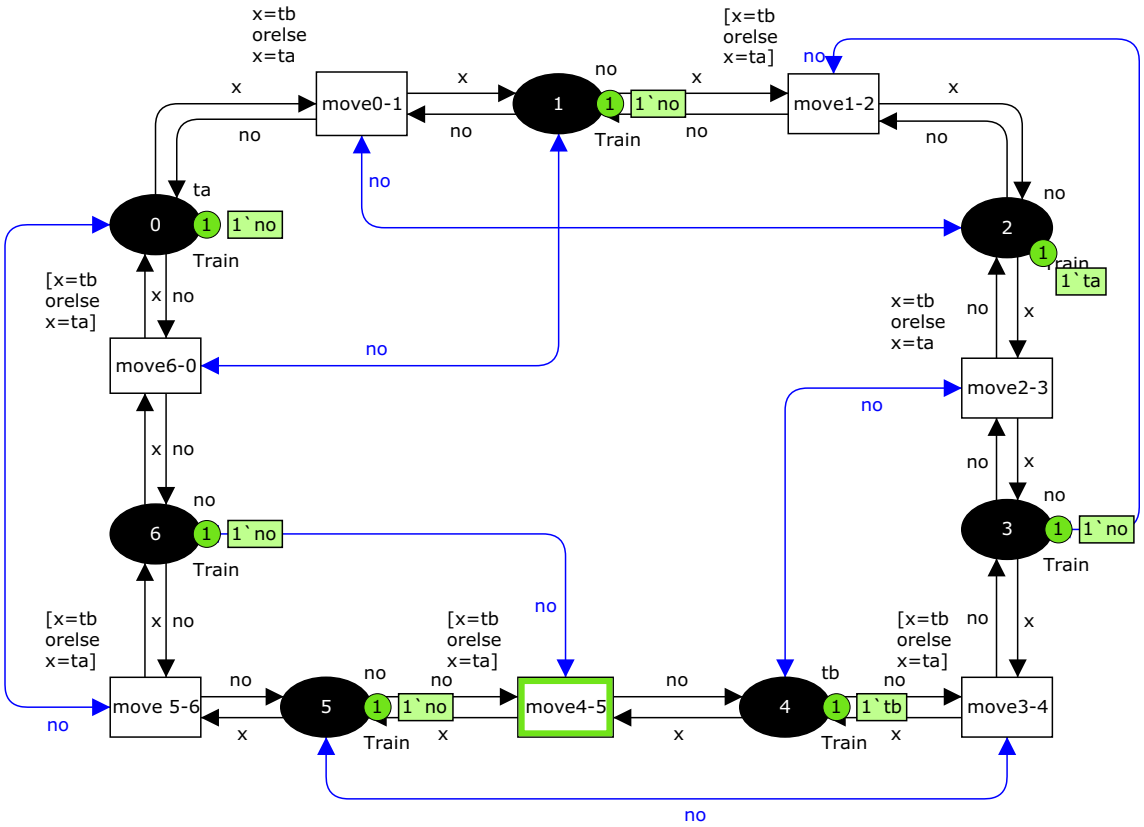
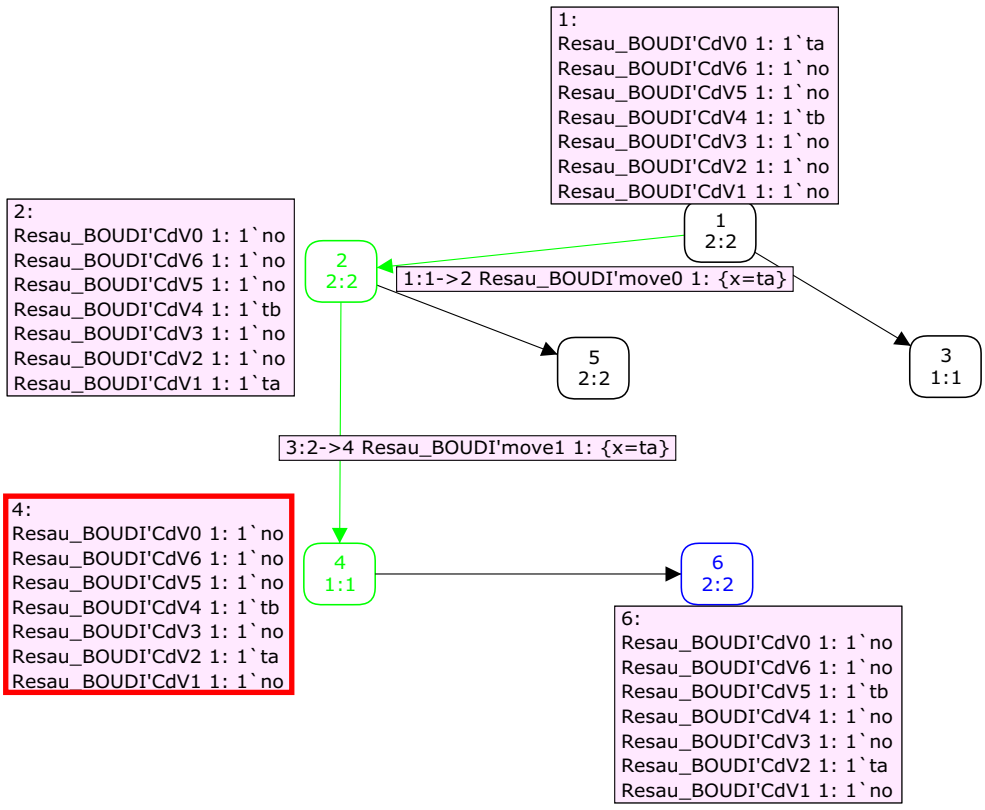


Fig. 15 Petri net after firing "move0–1" and 'move1–2″ transitions

Fig. 16 CPN tools state space after firing "move0–1" and 'move1–2″ transitions



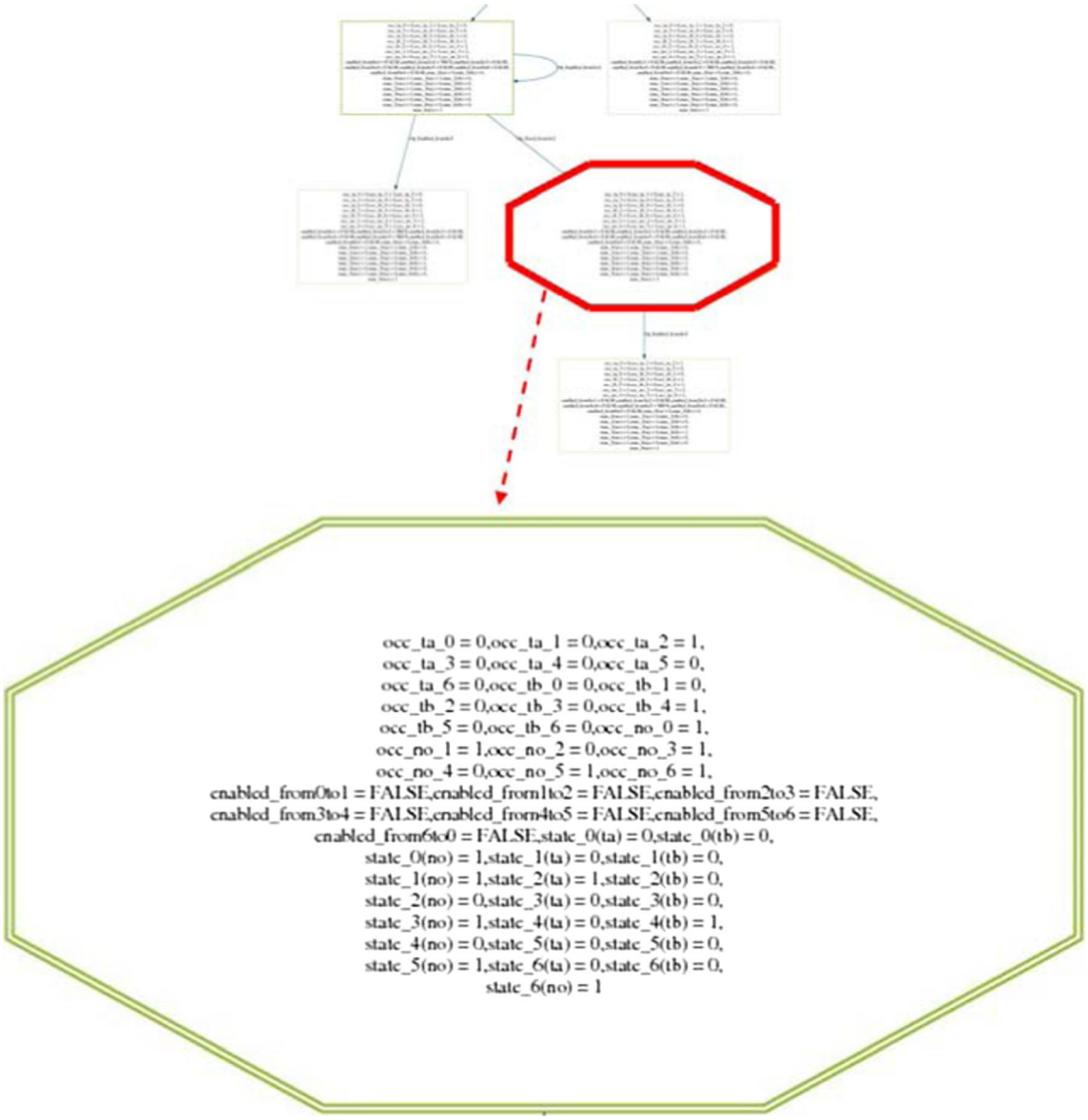


Fig. 17 ProB state space view after operations execution

"Atelier B" proof reached a **rate of 100%** after having added the previous invariant and concretized the implicit conditions of the B machine operations, such as completing the implicit conditions expressed by "***state_Idplace = Ms_empty(train)"*** with the precondition "***occ_eltcolor1_idplace =0 & … & occ_eltcolorN_idplace=0***", which does not affect the behavior of the obtained B machine.

6 An example of ERTMS case study – RBC handover

Of course, the present methodology has many possible applications the within development and verification processes in train control systems such as ERTMS, mainly because of the wide use of formal techniques in those environments, especially state diagrams, which can easily be remodeled using Petri Nets, and the B method. For example, one common procedure to be implemented by Radio Block Centers

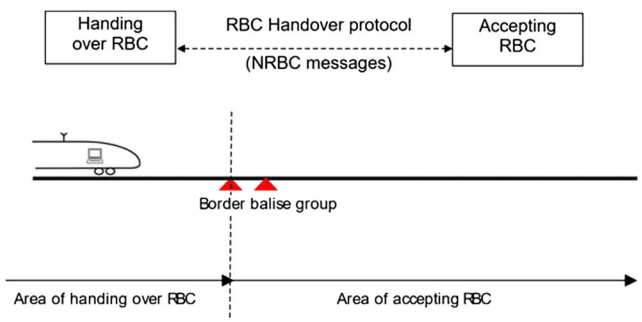


Fig. 18 RBC/RBC Handover

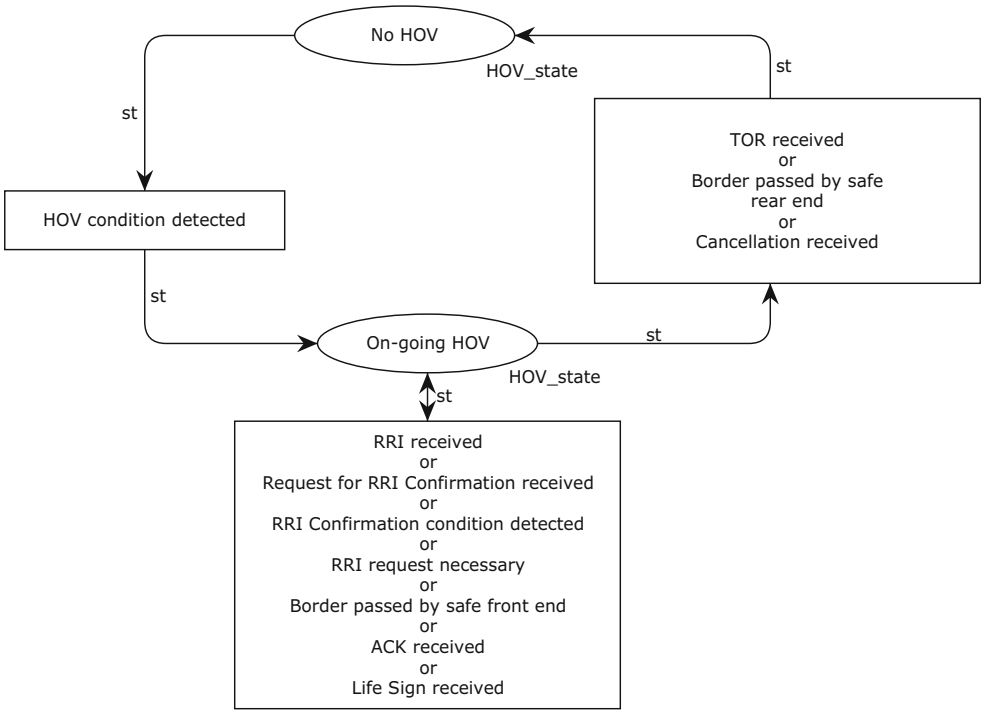


Fig. 19 Handing over RBC Petri Net

(RBCs) is the Handover. In fact, the RBC Handover (HOV), as described in Subset-039 [22], is about the communication between two RBCs when a given train is changing its corresponding RBC to the other (Fig. 18). This communication consists of neighbor RBC (NRBC) messages as shown in the following fig. [22].

In particular, the Subset-039 [22] details the states and events of one RBC/RBC handover transaction using two possible states: "No RBC/RBC handover is in progress" and "An RBC/RBC handover is on-going. The RBC has the role of Handing over RBC". For our study, we suggest the following Petri Net model (Fig.19) as an equivalent to the state diagram describing the transaction in the Subset.

The following table describes the various events involved in the RBAC/RBC Handover transaction (Table 1).

One clear advantage of transforming the Petri Net model describing the RBC Handover's transaction is the ability to use, without losing the logic of the process, the resulting B machine in formally proving safety properties of a bigger train operations model. Besides, such an bridge between the two formalisms could, in the case of ERTMS equipment development, help the implementation of safe by design sotware solutions.

Table 1 Incoming events of Handing over RBC

Event	Description
HOV condition detected	Handover condition detected
RRI request necessary	Handing over RBC detects that route related information is required from the accepting RBC
RRI received	NRBC message "Route Related Information" received
Condition "Border passed by safe front end" detected	Position report received and condition "Border passed by maximum safe front end" detected
TOR received	NRBC message "Taking Over Responsibility" received
Condition "Border passed by safe rear end" detected	Position report received and condition "Border passed by minimum safe rear end" detected
Cancellation condition detected	Condition for cancellation of the RBC/RBC handover transaction is detected in the handing over RBC
ACK received	NRBC message "Acknowledgement" has been received
Cancellation received	NRBC message "Cancellation" received
Life Sign received	NRBC message "Life Sign" received
Request for RRI Confirmation received	NRBC message "Request for RRI Confirmation" received
RRI Confirmation condition detected	RRI Confirmation condition detected

7 Conclusion and perspectives

Under the project PERFECT (Performing Enhanced Rail Formal Engineering Constraints Traceability), consistency between ERTMS specifications and national operating railway rules is to be checked and validated using formal techniques. In this context, researchers and experts model the railway conduct and rules through different formalisms, such as High-Level Petri Nets and UML. One of the biggest challenges of the project is to introduce validation methodologies based on formal techniques and convert all the obtained models to a common powerful formal tool able to link those models and keep all their initial information.

In this respect, this work makes a significant step toward the main goal of the project. It brings the base framework, of Colored Petri Nets conversion into B abstract machines, where behavioral equivalence of the two formalisms was demonstrated. An illustrative case study was conducted to show the application process of such a validation methodology in the railway context using the tools of the B method.

Therefore, this contribution has several perspectives. One of them is the establishment of an elaborate formal description of the transformation base on the rules presented here and the development of a software platform for its automation. Indeed, an automatic conversion tool is very important to facilitate the generation of B abstract machines because it takes a very long time to establish them manually, especially for large models. On the other hand, this work leads to question on how it is possible to link different B machines obtained from different input models, and those established by means of diverse formalisms, such as Petri Nets and UML in the case of the PERFECT project, with the purpose of checking the consistency between ERTMS and national railway specifications using the B tools [3, 4, 23]. This will help the project to achieve its main goal and propose an interesting background for railway safety scientific researches. A further purpose of this work is to develop a complete automatic railway safety rules validation tool.

References

1. Ghazel M (2014) Formalizing a subset of ERTMS/ETCS specifications for verification purposes. Transportation research-Part C: Emerging technologies 42: 60-75. DOI:10.1016/j.trc.2014.02.002, ELSEVIER, http://www.sciencedirect.com/science/journal/0968090X, 0968-090X
2. Jabri S, El-Koursi EM, Bourdeauhuy TH, Lemaire E (2010) European railway traffic management system validation using UML/Petri nets modelling strategy. Eur Transp Res Rev (ETRR), ECTRI 2:113-128. DOI: 10.1007/s12544-010-0030-5, Springer, http://www.springer.com/engineering/mechanical+eng/journal/12544
3. Boudi Z, El-Koursi EM, Collart-Dutilleul S, Khaddour M (2014) High level Petri net modeling for railway safety critical scenarios. In: Proc. of 10th FORMS/FORMAT symposium on formal methods. Braunchweig, pp 65-75
4. Ben Ayed R, Collart-Dutilleul S, Bon P, Idani A, Ledru Y (2014) B formal validation of ERTMS/ETCS railway operating rules. In: Ait Ameur Y, Schewe K-D (eds) ABZ 2014, LNCS, vol 8477. Springer, Heidelberg, pp 124–129
5. Sun P, Collart-dutilleul S, Bon P (2014) Formal modeling methodology of French railway interlocking system via HCPN. 14th International conference on Railway Engineering Design and Optimization, Rome, Italy
6. Antoni M (2009) Formal validation method for computerized railway interlocking systems. Computers Industrial Engineering. CIE 2009. International Conference on, pp 1532–1541
7. Murata T (1989) Petri nets: properties, analysis and applications, an invited survery paper. Proc IEEE 77(4):541–580
8. Genrich HJ (1991) Predicate/Transition Nets. K. Jensen and G. Rozenberg (Eds.): High-level Petri Nets. Theory and Application. Springer-Verlag, pp 3-43
9. Reisig W (1991) Petri nets and algebraic specifications. Theor Comput Sci 80(1):1–34
10. Jensen K (1992) Coloured Petri Nets – Basic Concepts, Analysis Methods and Practical Use-Vol. 1. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, pp 1-X, 1–236
11. Jensen K, Kristensen LM, Wells L (2007) Coloured Petri nets and CPN tools for Modelling and validation of concurrent systems. Int J Softw Tools Technol Trans (STTT) 9(3–4):213–254
12. Ratzer AV, Wells L, Lassen HM, Laursen M, Qvortrup JF, Stissing MS, Westergaard M, Christensen S, Jensen K (2003) CPN Tools for Editing, Simulating, and Analysing Coloured Petri Nets. Proc. of 24th International Conference on Applications and Theory of Petri Nets (Petri Nets 2003). Lecture notes in computer Science, vol 2679. Springer-Verlag Berlin, pp 450–462
13. Lalouette J, Brinzei N, Malasse O, Caron R, Scherb F, Aubry J-F (2010) Modeling and performance assessment of a railway signaling system integrating ETCS and BAL using colored Petri nets. Version 1, Sixth International French Conference of Automatic, CIFA 2010, Nancy, France
14. Abrial J-R (1996) The B-book: assigning programs to meanings. Cambridge University Press
15. Stuckenschmidt H (2002) Ontology-Based Information Sharing in Weakly Structured Environments. PhD thesis, University of Vrije
16. Redjimi M, Boukelkoul S (2013) Algorithmic tools for the transformation of Petri nets to DEVS. Informatica 37:411–418
17. Aubrecht P, Zakova M, Kouba Z (2005) Ontology Transformation Using Generalized Formalism. Znalosti, roèníku conference, pp. 154-161, V©B-TUO
18. Combemale B, Crégut X, Garoche PL, Thirioux X (2009) Essay on semantic definitions in MDE - an instrumented approach for model verification. JSW, vol 4(9):943–958
19. Istoan P. Methodology for the derivation of product behavior in a Software Product Line. PhD thesis, University of Rennes 1, University of Luxembourg, February 2013
20. Bon P, Collart-Dutilleul S (2013) From a solution model to a B model for verification of safety properties. J UCS 19(1):2–24
21. Korečko S, Sobota B (2014) Petri nets to B-language transformation in software development. Acta Plotechnica Hungarica 11(6):2014
22. UNISIG. Subset-039: FIS for the RBC/RBC Handover. April 2009
23. Bon P, Collart-Dutilleul S, Sun P. Study of implementation of ERTMS with respect to French national rules using a B centred methodology, International Conference on Industrial Engineering and Systems Management IESM'2013 October 28–October 30 Rabat – Morocco