

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/333225577>

B-Specification of Relay-Based Railway Interlocking Systems Based on the Propositional Logic of the System State Evolution

Chapter · January 2019

DOI: 10.1007/978-3-030-18744-6_16

CITATIONS

14

READS

176

4 authors, including:



Dalay Almeida

ClearSy System Engineering

10 PUBLICATIONS 23 CITATIONS

[SEE PROFILE](#)



David Déharbe

ClearSy System Engineering

114 PUBLICATIONS 940 CITATIONS

[SEE PROFILE](#)



Matthieu Perin

IRT Railenium, Lille, France

20 PUBLICATIONS 72 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



LCHIP (FUI21) [View project](#)



An Extension of a Tool for Formal Support to Component-based Development [View project](#)

B-specification of Relay-based Railway Interlocking Systems Based on the Propositional Logic of the System State Evolution*

Dalay Israel de Almeida Pereira¹[0000-0001-9698-5569],
David Deharbe²[0000-0001-7589-3323], Matthieu Perin³[0000-0002-9726-2458], and
Philippe Bon¹

¹ Univ. Lille Nord de France, IFSTTAR, COSYS/ESTAS
59650 Villeneuve dAscq, France

`dalay-israel.de-almeida-pereira@ifsttar.fr`

² ClearSy S.A., Aix-en-Provence, France

³ Institut de Recherche Technologique Railenium, F-59300 Famars

Abstract. In the railway signalling domain, a railway interlocking system (RIS) is responsible for controlling the movement of trains by allowing or denying their routing according to safety rules. Relay diagrams are a commonly used abstraction in order to model relay-based RIS, describing these systems by graph-like schemata that present the connections between electrical components. The verification of these diagrams regarding safety, however, is a challenging task, due to their complexity and the lack of tools for the automatic proof and animation. The analysis of relay diagrams by a specialist is the main method to verify the correctness and the safety of these systems. Nonetheless, human manual analysis is error prone. This paper presents an approach for formally specifying the behaviour of the systems described in relay diagrams in the B-method formal language. Considering that each relay has only two states, it is possible to describe the rules for the state evolution of a system by logical propositions. Furthermore, it is possible to use ProB in order to animate and model-check the specification.

Keywords: Railway Interlocking Systems · Relay Diagrams · B-method · Propositional Logic

1 Introduction

Railway Interlocking Systems (RIS) are built with the objective of controlling the movement of trains by allowing and denying their movements in specific tracks in order to avoid the occurrence of problems like collisions, for instance. The first built RIS was purely mechanical, than it evolved to use new technologies, becoming electrical mechanical systems, relay-based systems and, more recently, computer controlled systems [10]. As critical systems, these RIS must be specified

* Supported by the LCHIP (Low Cost High Integrity Platform) project.

and safety proved in order to guarantee the absence of critical errors, which, in this case, may lead to the loss of people lives.

Despite the existence of new computer technologies, many old companies still implement RIS as relay-based systems, which are electrical circuits containing relays. However, the safety proof of these systems is a challenging task, since they are modelled by electrical circuits drawings (relay diagrams) and the only way to verify them is by manually inspecting and drawing conclusions, which is error prone [11]. Consequently, the railway domain needs new methodologies for the specification and verification of railway interlocking systems.

In this context, formal methods arises as a useful tool for the specification, proof and analysis of RIS. Among many formal methodologies, the B-method has excelled in the railway field. The work presented in [6] compares the applicability of different formal methods to railway signalling and the B-method was shown to be one of the strongest approaches that may be used in the verification of such systems. Some of the reasons of this success are: the existence of rigorous mathematical foundations, the well-developed underlying methodology and the existence of reasonably advanced support tools, which allows the specification, refinement and implementation of B-machines with automatic code generation and performing verifications at each stage.

This paper presents a methodology for the specification of relay-based RIS behaviours in B-method based on the preconditions for the state evolution of the system electrical components. These conditions may be written in propositional logic based solely on the specific behaviour of each component. From the specification of these preconditions, it is possible to specify the complete behaviour of a RIS in B, which allows the proof of safety properties and the animation of the system by using the B-method supporting tools. Another contribution of this paper is to provide a non-exhaustive dictionary for the specification of state evolution preconditions for each type of electrical components described in relay diagrams, which is the basis of the specification of these systems in B.

There are many different formalisms and patterns that may be used in order to model relay diagrams. In this work, we focus on the models used by SNCF (the French National Railway Company) in order to implement their relay-based RIS. The choice of using B-method as a targeting language is explained by the success of the application of this language for the specification and proof of railway systems ([3], [12], [9]). Besides, B-method disposes of a complete set of supporting tools that allows its specification, verification, refinement and automatic code generation, which may be used in the future in order to transform these systems from relay-based to computer controlled RIS. Regarding components failures, they are not taken into account in our approach, since it requires a complete RIS failure model that is subject for a future work.

Many existing works have presented approaches for the formal specification and verification of RIS in order to verify safety properties ([17], [10], [8]). However, only some of them focused on the specification of relay-based RIS ([11], [4], [16], [7]), which is a technology that is still used by many railway companies. Despite the fact that these works are in the same field, the context is a

differentiating factor, since each company uses different notations, patterns and languages in order to model relay-based RIS.

In [11] it is presented an approach for the formal specification and verification of relay-based RIS applied for the danish RIS specification. Unlike SNCF RIS models, the relay diagrams presented in [11] use different patterns and notations, besides the fact that it contains fewer different types of components, which makes these systems less complex. Consequently, this work allows the behavioural translation of a smaller set of electrical components in comparison to our work, since their context (danish systems) is different from ours context (SNCF systems).

Furthermore, although the proximity with our work, [11] focuses on the specification of the temporal logic of the system based on the process description allowed by LTL [2] and our work focuses on the specification of the stable states of the RIS in order to find possible unsafe states. A stable state is defined as a moment when the system "waits" for an interaction (input changes) in order to change its own state. Furthermore, B-method allows the specification of abstract machines that can be refined in order to generate code. Our work represents a first step towards a transformation from relay-based RIS towards computer-controlled RIS, which may be supported by the B-method refinement and automatic code generation processes.

The Section 2 of this paper presents some details about the modelling of RIS in relay diagrams, followed by the formal methodology of specification, B-method, in Section 3. Then, based on these specification languages, it is possible to discuss about the specification of RIS behaviour in B-method based on relay diagrams in Section 4. The last Section of this paper concludes our work and presents some perspectives.

2 Relay-based Modelling

Interlocking systems are the signalling functions controlling the trains movements in a particular location in order to meet safety requirements [17]. Many railways interlocking systems implemented by railway companies are relay-based systems, which is formed by electrical circuits containing relays. Responsible for the transmission, reception and use of information, a relay is an electromechanical switching element comprised by electromagnets (coils) and contacts [14]. In a relay-based RIS, an electrical circuit is composed by a source of energy, a command element (like contacts or levers, for instance) and a receiver (like relays or outputs, for instance), which are connected by conductive wires. A component or wire is electrified if it is connected to the positive and negative sources of energy poles.

2.1 Relay-based Railway Interlocking Systems Modelling

Relay-based RIS are usually modelled by relay diagrams, which are graph-like representations of how the electrical elements are connected by wires. This sec-

tion presents some details about the modelling of RIS in industry, more specifically, how SNCF uses relay diagrams in order to model railway interlocking systems. An example of a relay diagram is presented in the Figure 1 (detailed in Section 2.2). Some graphical notations that may be used in a relay diagram are presented in Table 1.

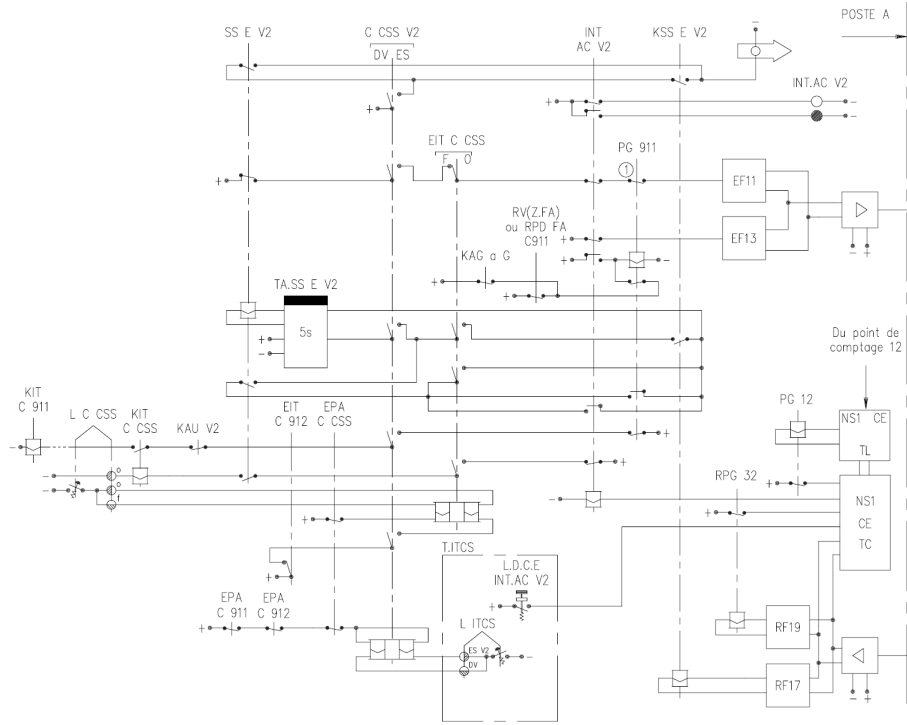
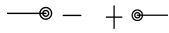
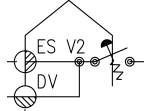
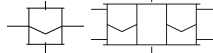

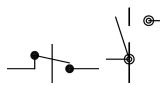


Fig. 1. Relay-based system model of the signalling control point A

Originally, a contact has two different states in a RIS: closed or opened. The former state allows the electrical current to flow from a wire to another. In the latter state, the contact is not connected to both wires, so it does not allow the electrical current to flow. A contact may be closed or opened by the influence of the gravity or by the electromagnetic influence of a relay. Furthermore, a contact is "normally closed" if it is necessary a magnetic influence to open it (the gravity maintains it closed otherwise). Following the same principle, a contact is "normally opened" when the gravity maintains it opened as a way that it is necessary a magnetic influence in order to close it.

There are two different types of relays: monostable and bistable. A monostable relay contains only one electromagnet that is responsible for attracting or repulsing one or more movable contacts. In this case, the contacts related to a

Table 1. Elements that may be used in a Relay-based diagram.

	Electrical sources of energy poles (negative and positive, respectively).
	Couple button-lever.
	Monostable and bistable relays, respectively.
	Blocks for timed activation and deactivation, respectively.
	A normally closed contact related to a monostable relay and a contact related to a bistable relay, respectively.

monostable relay are disposed horizontally in a way that the gravity may maintain these contacts closed or opened when the electromagnet is not electrified. The relay is the responsible for changing and maintaining the states of the contacts related to it. On the other hand, a bistable relay contains two coils, each one may pull the contacts to one side. In this case, the contacts are disposed vertically as a way that the coils may pull the contacts to the left or right side in order to change their states. However, if both coils lose energy, the last contact state is maintained by the gravity.

Blocks are not physical components in a railway system, but they represent an important part in relay diagram models, since they allow the modelling of timed relays. A block with a black thicker line on the top indicates that it delays the activation of a relay. On the other hand, a block with a thicker black line on the bottom is responsible for retarding the deactivation of a relay. Inside the white part of the block it is indicated the time spent on these delays, which is normally indicated in seconds.

There are many types of inputs that may be used in the relay diagrams modelling, which represent the interface between these diagrams and the environment. Some examples of inputs that allows the human intervention inside the system are buttons and levers. A button acts like a contact since it may connect two different wires. However, its states are controlled by the environment, which means that it is not magnetically connected to a relay. A lever is similar to a button, since it needs a physical force in order to change its states. However, a lever controls the flow of electrical current in more than one pair of wires. Furthermore, if a lever allows the current flow in one pair of wires and at the

same time it blocks the current in another pair, it will maintain these states alternated after the external intervention by changing all the states together.

An input may also be represented by a contact whose associated relay is not presented in the diagram. The behaviour of this abstracted relay is considered as an input since it controls the state of the contact. As an example, the detection of the position of a train may be modelled by abstracting the relay in the diagram, since the train (environment) is the responsible for the activation of the relay, which controls the state of the contacts represented inside the diagram.

The outputs of the RIS can be generally understood as a permission or denial for a train to enter in a determined track. These outputs must be verified in order to avoid giving permission to two different trains entering in the same track, for instance, which may cause a collision. An industrial example of a model that can be verified in order to avoid collisions is presented in the next subsection. This model is used as an example throughout this paper.

2.2 Industrial example

This subsection presents details about an example that is used in industry. The diagram presented in Figure 2 represents a track plan containing two tracks (one for each direction) and the space between the signalling control point A and C. In this example, we consider that a train that arrives in the point A may change to the track below because of problems on its own track. In this case, this train will start going on the "wrong way", which may cause a frontal collision with a train that may come from the point C. In order to avoid this type of collision, there must exist a signalisation that indicates if a train may enter or not in this portion of the tracks. In this example we focus on the signalisation existing in the point A.

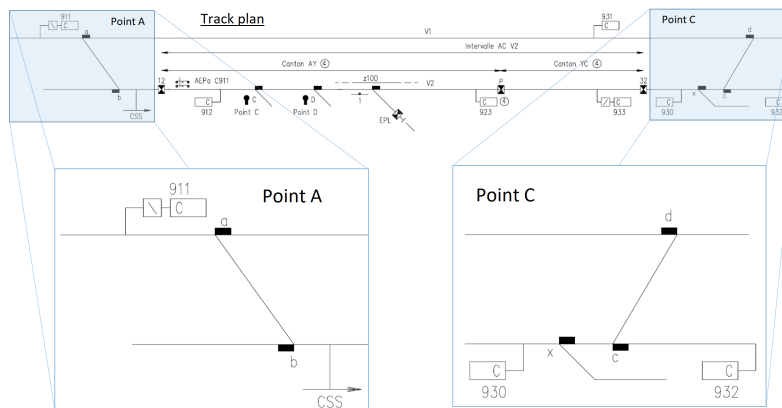


Fig. 2. Track plan from the signalling control point A to C

In order to control the signalisation in the signalling control point A, a relay-based RIS must be implemented according to the model presented in Figure 1. In this system, the pair button-lever *L ITCS* is responsible for indicating if the tracks are working normally (*DV* - "Double Voie") or if it is necessary to use only one track (*ES* - "En Service"), which is a situation that may cause a collision if not safety proved. After switching *L ITCS* to the *ES* state, it also changes the bistable relay *C CSS V2* to the *ES* state, which allows the train that arrives in the point C to enter in the dangerous zone (activation of the output *EF11*), since it refuses a train that arrives in point A to enter (*KIT C 911* deactivated). If *L ITCS* is set to *DV*, it means that the two tracks are working normally and the signal is never closed.

If a train aim to enter in the dangerous zone from the point A, a permission may be given by changing the state of the lever *L C CSS* to *O*, which also changes the relay *EIT C CSS* to the *O* state. This action will deactivate the output *EF11*, which no longer gives permission to trains in the point C to enter in the dangerous zone. If the point C agrees with these changes, it may allow a train in the point A to enter by activating the relay *KSS E V2*. These sequence of actions may activate the relay *SS E V2* after a delay of five seconds, counted by the block *TA.SS E V2*.

After the activation of the relay *KSS E V2*, a permission is granted to a train in the point A to enter in the dangerous zone by the activation of the relay *KIT C 911*. This permission must be given only if *EF11* is deactivated and vice versa. This is a condition that must be guaranteed.

3 B-Method

According to [1], B is a method for specifying, describing and coding software systems. By making use of a strong mathematical background, it allows the specification and verification of systems in a formal manner in order to guarantee a high level of reliability. The first successful use of this method in an industrial case was the Meteor line 14 drive-less metro [3], in Paris, in which it was specified over than 110,000 lines of B-models, generating 86,000 lines of code [12]. No one has ever detected a bug in this system in the functional and integration validation, neither on the on-site test. Since then, other systems has been successfully specified and implemented using B-method, like, for instance, the COPPILOT [12] system. Besides, B has been also used for proving the correctness of other existing systems, like, for instance, SACEM [9].

The basic building block of a B-method specification is the abstract machine [15]. One system may be specified by one or several machines. The specification inside a machine is divided in many parts, each one under an appropriate heading (or clause) describing a different aspect of the specification. The first heading, *MACHINE*, starts the specification of an abstract machine, whose name must be described under this heading.

The local state of a machine is kept by the variables which are defined under the *VARIABLES* clause and whose details are defined under the *INVARIANT*

heading. These details comprise variables typing and properties that must be satisfied by the specification. The initial state of the machine must be described in the *INITIALISATION* clause. It is also possible to describe constant information, like, for instance, sets of constant information that can be used inside the machine, which are described under the *SETS* heading.

It is also possible to define operations for a machine inside the *OPERATIONS* clause. These operations may receive inputs, provide outputs and change the state of the machine by changing the values of the variables. In order to define an operation it is required to define the preconditions that must be met in order to execute this operation.

```

5- MACHINE
6   example
7
8- SETS ANSWERS = {yes, no}
9
10 VARIABLES answer
11 INVARIANT answer : ANSWERS
12 INITIALISATION answer :: ANSWERS
13- OPERATIONS
14
15   ans <-- set_answer(input) =
16   PRE input : ANSWERS
17   THEN answer := input || ans := input
18   END
19
20 END

```

Fig. 3. Example of a simple B-machine

A small example of a B-machine is depicted on Figure 3. This example presents how the clauses can be used in order to specify a machine that allows the storage of an information of the type defined by the set *ANSWERS*. The information that can be stored inside the variable *answer* are the elements *yes* or *no*. More details about the clauses and notations used in order to specify a B-machine can be found in [1].

4 B-specification of relay-diagrams behaviours

A relay-based RIS is composed of many components, each one with an independent specific behaviour. The specification of an entire RIS may be described as a combination of the behaviours of all the components that constitutes it. Furthermore, in order to activate a component, an electrical current is required (precondition) and the flux of electrical current may be controlled by other components with other preconditions. This type of reactive behaviour can be described inside a B-method abstract machine, which may be used for proof and animation purposes, as it is presented in this section.

4.1 Relay-diagram behavioural logic

The most important components of a relay-based RIS are the relays. This component is responsible for opening and closing the contacts, which controls the flux of electrical current inside the wires. Therefore, the activation and deactivation of relays commands the activation and deactivation of other components by controlling the flow of electrical current inside the wires. Before specifying any component behaviour, it is important to understand the precondition for a component to be activated.

Definition 1. (*Component Activation Precondition*) *An electrical component is activated if both of its wires are connected to a different pole (positive and negative) of energy sources as a way to allow the flow of electrical current inside the component. This means that all contacts, buttons and levers between the component and both poles of energy sources must be closed.*

It is clear that a component is activated if there is electrical current flowing inside it, however, the precondition for having electrical current is that each wire connected to a component must be connected to a different pole of sources of energy. A precondition for the component deactivation can also be defined.

Definition 2. (*Component Deactivation Precondition*) *An electrical component is deactivated if its wires are not connected to different poles (positive and negative) of energy sources as a way that there does not exist a flow of electrical current inside it. This means that at least one contact, button or lever between the component and one pole of energy source must be opened (considering that there is no other connection to the same type of pole).*

A monostable relay has two states: TRUE (activated) or FALSE (deactivated). The former represents the state where there is current passing through the coil and the latter represents the state where the coil is not electrified (according to Definition 1 and 2, respectively). The consequence of its activation or deactivation is the state evolution of the contacts related to this relay. The state of a monostable relay changes as soon as the precondition of this state is no longer met.

A bistable relay has also two states: right or left, representing the activation of the right and left coils, respectively. Generally, the two coils will not be activated at the same time, however, both coils may be deactivated. If the right coil activates, the relay assumes the "right" state and it changes the state of the contacts related to this relay. Then, this may change to "left" if, and only if, the right coil is deactivated and the left coil activates, which changes the state of the contacts as well. The main difference between a monostable and a bistable relay relies on the fact that the latter may maintain its last state even if the coils are no longer activated.

Considering that the contacts states are directly defined by the relays related to them, the states of the contacts in the relay diagrams can be completely abstracted by means of the relays states. As an example, the precondition for

the relay *KIT C CSS* to be activated in the relay diagram depicted on Figure 1 is that *SS E V2* must be activated (in order to close the normally opened contact), *EIT C CSS* must be set to the right (which closes the bistable contact) and *INT AC V2* must be activated (in order to close the normally opened contact).

Furthermore, there are two special cases concerning the logic of relays: timed activated/deactivated and self-alimented relays. In the former case, the Definitions 1 and 2 changes in the presence of a block. A relay connected to a block that retards the activation will be activated if the block is activated (according to Definition 1) after the time represented by the block. In this case the relay deactivation occurs right after the deactivation of the block (not timed). Contrarily, a relay connected to a block that retards the deactivation activates right after the activation of the block, however, this relay deactivates only after the time defined inside the block when the block is deactivated. The relay *SS E V2* is an example of timed-activation relay.

A relay can also be self-alimented when it controls a contact that may activate it. In a case that the activation of a relay closes a contact that also alimets it with energy (like the relay *PG 911*, for instance), this contact may never activate the relay by itself, so it is not considered as a precondition for the relay activation. Furthermore, this contact does not open unless the relay is deactivated, so it is also not considered in the precondition for the deactivation of the relay either. However this contact has a high importance in order to maintain the activated state of the relay after its activation.

Similar to contacts, buttons and levers are responsible for the activation or deactivation of other components, since they control the electrical current flow inside the wires. However, the states of buttons and levers cannot be abstracted, since they are controlled directly by the environment. So, these components may be treated as inputs and the information inserted in the system by these inputs are important for the definition of the system state.

Furthermore, the outputs are part of the general state of the diagram, representing an important part on the safety analysis, since they represent the response given to the environment calculated based on the inputs. In a relay-diagram, an output may be connected to only one energy source pole (like *EF11*, for instance), or to two different poles (like the *INT.AC V2* lights, for instance) in order to be activated. Besides, it may be depicted as another component, like relays (*KIT C 911*, for instance). One verification that is possible to make in our running example in order to analyse its safety is that it there must never exist a state where the component *KIT C 911* (permission for the train in the point A to enter in the dangerous zone) and the component *EF11* (permission for the train in the point C to enter in the dangerous zone) are activated at the same time, since it may cause a collision.

4.2 Relay-based Logic Specification in B

Based on the relay diagrams and on the logic for the state evolution of the electrical components, it is possible to specify the behaviour of the RIS in B-method. In order to specify these systems, it is necessary to define what is inside

each header of the B specification. After the *MACHINE* header, which contains the machine, one must define sets containing the special states related to levers or bistable relays under the *SETS* clause. Regarding the example used throughout this paper, It is necessary to define the special states *POS_O* and *POS_F* for the components *EIT C CSS* and *L C CSS* as well as the states *POS_DV* and *POS_ES* for the component *C CSS V2*. When related to bistable relays, these states represent the left (*DV, F*) or right (*ES, O*) state of these components. These states represent the positions that these components may assume. Hence, our running example can be initially specified as shown in Figure 4.

<pre> MACHINE itcs SETS O_OU_F = {POS_O, POS_F}; DV_OU_ES = {POS_ES, POS_DV} </pre>	<pre> VARIABLES KIT_C_CSS, SS_E_V2, TA_SS_E_V2, EIT_C_CSS, C_CSS_V2, PG_911, EF11, KIT_C_911 </pre>
---	---

Fig. 4. B-method MACHINE, SET and VARIABLES clauses

In a B-method specification the variables (listed inside the *VARIABLES* clause) define the state of a machine. In the RIS context, this state is defined by the state of each component. Hence, in our methodology, the variables must represent components. However, in order to simplify the specification and decrease a significantly number of variables and, by consequence, the number of possible states, inputs and contacts are not treated as variables. As presented before, the state of the contact is directly linked to the state of the relay in a way that contact states can be easily abstracted. Furthermore, since inputs are responsible for directly or indirectly changing the states of all other components, we chose to specify them as the inputs for the operation responsible for the state evolution. In other words, these components affect the system, but their states are not maintained by the system, since they may be changed at any time by the environment. This option does not affect negatively the safety verification and neither the animation of the specification. In fact, by avoiding the specification of inputs as variables we decrease the number of specified states, which allows a faster and lighter model-checker verification.

A special case regarding variables is the specification of blocks. The state of this component is maintained by the system because they are nor directly activated by the environment. However, the environment has an effect over the blocks, since the time has an important part on its behaviour. In this work, time is considered as an environmental factor. So, this type of component must not only be specified as a variable, but it must also be specified as an input of the state evolution. This input represents the passing of time. In our running example, the *VARIABLES* clause is specified as presented in Figure 4.

Inside the *INVARIANT* clause, one must define the type of the variables. In this case, regarding RIS components, the types represent the possible states that the components may assume. As presented before, monostable relays may assume the states *TRUE* or *FALSE*, in other words, they have the Boolean type. Since the outputs may also be activated or deactivated, they also must be Boolean. However, bistable relays may have special types defined inside the diagrams in order to indicate the left and the right states. In our running example, for instance, we have the types *O_OU_F* and *DV_OU_ES* for the relays *EIT C CSS* and *C CSS V2*, respectively.

Moreover, inside the *INVARIANT* clause, it is also possible to define conditions that must be respected at any possible state of the machine. In the case of RIS specification, one may describe safety properties in order to guarantee that the system will never reach a dangerous state. In our running example, for instance, a safety property that must be always met is that the components *KIT C 911* and *EF11* must never be activated at the same time in order to avoid collision. The specification of this property and the complete *INVARIANT* clause of our example are depicted in Figure 5.

INVARIANT	INITIALISATION
KIT_C_CSS : BOOL &	KIT_C_CSS := FALSE
SS_E_V2 : BOOL &	SS_E_V2 := FALSE
TA_SS_E_V2 : BOOL &	EIT_C_CSS := POS_F
EIT_C_CSS : O_OU_F &	C_CSS_V2 := POS_DV
C_CSS_V2 : DV_OU_ES &	PG_911 := TRUE
PG_911 : BOOL &	TA_SS_E_V2 := FALSE
EF11 : BOOL &	EF11 := FALSE
KIT_C_911 : BOOL &	KIT_C_911 := FALSE
not(KIT_C_911 = TRUE &	
EF11 = TRUE)	

Fig. 5. B-Machine INVARIANT and INITIALISATION clauses

The initial state of the system is defined by the relay diagram drawing, since it shows the initial position of the levers, bistable relays, and if the monostable relays are connected to the energy or not. In our running example, the *INITIALISATION* clause is defined as presented in Figure 5.

The state evolution of a railway interlocking system sheet may be specified inside a B-method operation. The use of a unique operation allows us to reach all the stable states only by changing the inputs given by the environment. Since the inputs are the responsible for triggering the state evolution, they are the inputs of the operation. Inside the precondition clause of the operation, all the inputs must be typed. The operation must also be able to change the state of all variables considering that the state of one variable may affect the final state of another. This type of behaviour can be specified in B by the following notation:

```
<<variables>>:(<<variables typing>> & <<variables information>>)
```

By using this expression, it is possible to change the value of a set of variables (<<variables>>) by informing their types (<<variables typing>>) and the conditions they must meet after the execution of this expression (<<variables information>>). Inside the conditions, it is possible to define the values of the variables according to the inputs and other variables states (activation and deactivation preconditions). As an example, the state evolution of the variable *KIT_C_CSS* that must be described inside the <<variables information>> part of the notation is the activation condition presented in Figure 6.

```
KIT_C_CSS = bool(SS_E_V2 = TRUE & EIT_C_CSS = POS_0 &
INT_AC_V2 = TRUE & L_C_CSS = POS_0)
```

Fig. 6. Example of a state evolution

So, in order to activate the component *KIT_C_CSS*, *SS_E_V2* and *INT_AC_V2* must be electrified at the same time that *EIT_C_CSS* and *L_C_CSS* are in the *O* position (*POS_O*). Furthermore, it is possible to specify the same type of expression for each variable. The complete operation defined for our running example can be defined as presented in Figure 7, where all variables are represented in red and all the inputs are represented in green for sake of clarification.

In some cases, in order to define the state evolution of a relay, it is necessary to consider its previous state (before the execution of the operation), which may be specified by using the notation *\$0* after the name of the variable [5]. This rule applies for the specification of bistable, self-alimented or timed relay behaviours. In case of bistable relays (as for the relays *EIT_C_CSS* and *C_CSS_V2*, for instance), one must consider that the previous state must be maintained if there is no electricity inside the coils, since gravity maintains the contacts closed.

Regarding self-alimented relays, the component *PG_911* is one example of it. The contact related to this relay may never be responsible for changing the relay state. However, although this contact does not directly interfere in the relay activation and deactivation, it "blocks" other contacts that could be related to the relay activation. For instance, although the contacts of the relays *KAG_a_G* and *RPD_FA_C_911* may deactivate the relay *PG_911*, they are not able to activate it, since the *PG_911* contact is not able to activate the relay.

In the last special case, timed relays, as the relay *SS_E_V2*, for instance, one must consider the state of the blocks that they are related to. In order to activate or deactivate timed relays, the input related to the block time must be considered. This means that these relays can only be activated or deactivated if the time has passed (input set to *TRUE*). In our operation responsible for the state evolution, the input *TA_SS_E_V2_echue* represents the passing of time related to the block *SS_E_V2*. In this case, this input is only considered for the activation of the relay.

```

mise_a_jour_poste_A(L_C_CSS, INT_AC_V2, EPA_C_CSS, EIT_C_912, KAG_a_G, RPD_FA_C_911, L_ITCS, KAU_V2,
KSS_E_V2, EPA_C_911, TA_SS_E_V2_echue) =
PRE
L_C_CSS : O_OU_F & INT_AC_V2 : BOOL & EPA_C_CSS : BOOL & EIT_C_912 : BOOL & KAG_a_G : BOOL &
RPD_FA_C_911 : BOOL & L_ITCS : DV_OU_ES & KAU_V2 : BOOL & KSS_E_V2 : BOOL & EPA_C_911 : BOOL &
TA_SS_E_V2_echue : BOOL
THEN
KIT_C_CSS, PLUS_KIT_C_911, EIT_C_CSS, PG_911, C_CSS_V2, EF11, TA_SS_E_V2, SS_E_V2:(

KIT_C_CSS : BOOL & PLUS_KIT_C_911 : BOOL & EIT_C_CSS : O_OU_F & PG_911 : BOOL &
C_CSS_V2 : DV_OU_ES & EF11 : BOOL & SS_E_V2 : BOOL & TA_SS_E_V2 : BOOL &

KIT_C_CSS = bool(SS_E_V2 = TRUE & EIT_C_CSS = POS_O & INT_AC_V2 = TRUE & L_C_CSS = POS_O) &

PLUS_KIT_C_911 = bool(KIT_C_CSS = TRUE & KAU_V2 = TRUE & C_CSS_V2 = POS_ES & PG_911 = TRUE) &

(EIT_C_CSS$0 = POS_O =>
EIT_C_CSS = { TRUE |-> POS_F, FALSE |-> POS_O } (bool(L_C_CSS = POS_F & EPA_C_CSS = TRUE))) &
(EIT_C_CSS$0 = POS_F =>
EIT_C_CSS = { TRUE |-> POS_O, FALSE |-> POS_F } (bool(L_C_CSS = POS_O & C_CSS_V2 = POS_ES &
EIT_C_912 = FALSE))) &

(PG_911$0 = FALSE => PG_911 = bool(INT_AC_V2 = FALSE)) &
(PG_911$0 = TRUE => PG_911 = bool(RPD_FA_C_911 = TRUE or KAG_a_G = TRUE or INT_AC_V2 = FALSE)) &

(C_CSS_V2$0 = POS_ES =>
C_CSS_V2 = { TRUE |-> POS_DV, FALSE |-> POS_ES } (bool(L_ITCS = POS_DV & EPA_C_CSS = TRUE &
EPA_C_911 = TRUE))) &
(C_CSS_V2$0 = POS_DV =>
C_CSS_V2 = { TRUE |-> POS_ES, FALSE |-> POS_DV } (bool(L_ITCS = POS_ES & EPA_C_CSS = TRUE &
EPA_C_911 = TRUE))) &

EF11 = bool(SS_E_V2 = FALSE & C_CSS_V2 = POS_ES & EIT_C_CSS = POS_F & INT_AC_V2 = TRUE &
PG_911 = TRUE) &

(SS_E_V2$0 = TRUE =>
SS_E_V2 = bool(
(C_CSS_V2 = POS_ES & (EIT_C_CSS = POS_O or PG_911 = FALSE or INT_AC_V2 = FALSE)) or
(C_CSS_V2 = POS_ES & EIT_C_CSS = POS_O & KSS_E_V2 = TRUE))
) &
(SS_E_V2$0 = FALSE =>
SS_E_V2 = bool(TA_SS_E_V2$0 = TRUE & TA_SS_E_V2_echue = TRUE & (C_CSS_V2 = POS_ES &
EIT_C_CSS = POS_O & KSS_E_V2 = TRUE))) &

TA_SS_E_V2 = bool(SS_E_V2$0 = FALSE & (C_CSS_V2 = POS_ES & EIT_C_CSS = POS_O &
KSS_E_V2 = TRUE) & TA_SS_E_V2_echue = FALSE)
)
END

```

Fig. 7. State evolution of the signalling control point A specified in B-method

4.3 Animation and Verification

Many tools have been developed in order to support the B-method. One example of these tools is the ProB [13], which allows not only the animation of the

machines but also their specification and model-checking. In this work, ProB may be useful in order to animate the specification as a way to analyse the system behaviour. During the animation, the tool allows operations to be called and it always verifies if the machine state is valid according to the invariant.

Regarding the machine verification, ProB contains a model-checker that allows the verification of each possible state of the machine in order to find the existence of a state that does not meet the invariant. If an invalid state is found, the tool presents it as a counter example.

In order to analyse the machine representing our running example, it is possible to animate and verify it. The animation provides an overview of the execution of the system when implemented, and, in this case, the animation of the specification has shown to be accurate with the reality. Furthermore, the verification of the system by the model-checker guaranteed that, in a case where all components are working normally, two trains must not have the permission to go in opposite ways in the same track at the same time, meaning that the invariant `not(KIT_C_911 = TRUE & EF11 = TRUE)` is false in every possible machine state. The model-checking process took 3031 milliseconds, verifying the 36,865 possible transitions between the 18 existing states in order to analyse if any transition may lead to an inconsistent state. The verification was made by a 64bits Intel(R) Core(TM) i7-7600U 2.80GHz CPU with 16Gb RAM and running the Windows 10 operating system in its professional version.

However, although this strategy of specification is able to guarantee the absence of states that may lead to a collision, it is important to admit that the verification is not enough in order to guarantee the complete absence of collisions in the real field. The execution of the system in reality contains many other variables related to the context that are not specified in this work. These variables are related, for example, to the position of the trains in the tracks, the decisions made by the driver or even the well functioning of each component. This type of contextual information may be considered in the specification in order to prove the safety of the system. The specification and use of context variables in the B-method relay-based RIS specification are in our near future agenda.

5 Conclusion

This paper presented a strategy for the formal specification of relay-based railway interlocking systems in B-method based on the behavioural logic of the relay diagram electrical components. Since the complete behaviour of a system is comprised by the behaviour of each of its components, it is possible to specify the complete RIS system by using the specification of its components behaviours. Furthermore, as a reactive system based on the activation and deactivation of the components (boolean states), it is possible to specify the conditions for each component to be activated and the effect of their activation by using propositional logic, which is supported by B-method. Moreover, by using B-method, it is possible to animate and prove safety properties that should be enforced by the relay diagrams by the use of the tools that supports B-method.

By using the strategy presented in this paper, an example of relay diagram used in industry was specified in B and a safety property about this diagram was proved. The Prob model-checker was used in order to verify the B-specification and, as specified in the INVARIANT of the B-machine, the tool was able to prove that two trains may not have the permission to enter in the same track in opposite directions at the same time. Although this property is important in order to avoid frontal collisions, it is not enough, since there are many contextual variables that must be considered in order to prove the safety of the system.

In our upcoming agenda, we aim to be able to specify contextual variables related to the environment of the RIS system. These variables may specify information that must be considered in order to prove the safety of the system, like the position of the train or the possibility of the driver to make unsafe decisions. Besides, we aim to specify relay-based RIS based on the possibility of the components to failure. By analysing this specification, we intend to demonstrate the impact of these failures on the safety of the system and provide methodologies in order to avoid dangerous states.

Furthermore, B-method disposes of methods for system refinement and implementation, which may be explored in order to implement RIS as computer-controlled systems in the future. This paper presents a first step towards the possibility of evolving relay-based RIS into computer-controlled RIS by specifying the logic of relay diagrams in B-method based on propositional logic. This specification presents how the inputs affects the system in order to produce outputs, which may be refined in order to generate code that can be executed. Once compiled, the implementation of this system may be used inside small computers which are able to receive electrical inputs, process them based on the implemented system and emit outputs also in the form of electrical signals that manages the movement of the trains in the tracks.

References

1. Abrial, J.R.: The B-book: assigning programs to meanings. Cambridge University Press, New York and NY and USA (1996)
2. Beer, H.: The LTL Checker Plugins: A Reference Manual. Eindhoven University of Technology, Eindhoven (2004)
3. Behm, P., Benoit, P., Faivre, A., Meynadier, J.M.: Meteor: A successful application of b in a large project. In: International Symposium on Formal Methods. pp. 369–387. Springer (1999)
4. Cavada, R., Cimatti, A., Sessa, M.: Analysis of relay interlocking systems via smt-based model checking of switched multi-domain kirchhoff networks. In: The eighteenth in a series of conferences on the theory and applications of formal methods in hardware and system verification (FMCAD 2018). vol. 18, pp. 179–187. IEEE (2018)
5. ClearSy: B Language Reference Manual, version 1.8.5
6. Fantechi, A., Fokink, W., Morzenti, A.: B-specification of relay-based railway interlocking systems based on the propositional logic of the system state evolution. Formal Methods for Industrial Critical Systems: A Survey of Applications pp. 61–84 (2013)

7. Ghosh, S., Das, A., Basak, N., Dasgupta, P., Katiyar, A.: Formal methods for validation and test point prioritization in railway signaling logic. *IEEE Transactions on Intelligent Transportation Systems* **18**(3), 678–689 (2017)
8. Gjaldbæk, T., Haxthausen, A.E.: Modelling and verification of interlocking systems for railway lines. *IFAC Proceedings Volumes* **36**(14), 233–238 (2003)
9. Guiho, G., Hennebert, C.: Sacem software validation. In: *Software Engineering, 1990. Proceedings., 12th International Conference on.* pp. 186–191. IEEE (1990)
10. Hansen, K.M.: Formalising railway interlocking systems. In: *Nordic Seminar on Dependable Computing Systems.* pp. 83–94. Citeseer (1998)
11. Haxthausen, A.E., Le Bliguet, M., Kjær, A.A.: Modelling and verification of relay interlocking systems. In: *Monterey Workshop.* pp. 141–153. Springer (2008)
12. Lecomte, T., Servat, T., Pouzancre, G., et al.: Formal methods in safety-critical railway systems. In: *10th Brazilian Symposium on Formal Methods.* pp. 29–31 (2007)
13. Leuschel, M., Butler, M.: Prob: A model checker for b. In: *International Symposium of Formal Methods Europe.* pp. 855–874. Springer (2003)
14. Rétiveau, R.: *La signalisation ferroviaire.* Presse de l'école nationale des Ponts et Chaussées (1987)
15. Schneider, S.: *The B-method: An introduction.* Palgrave (2001)
16. Sun, P., Collart-Dutilleul, S., Bon, P.: A model pattern of railway interlocking system by petri nets. In: *Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2015 International Conference on.* pp. 442–449. IEEE (2015)
17. Winter, K.: Model checking railway interlocking systems. In: *Australian Computer Science Communications.* vol. 24, pp. 303–310. Australian Computer Society, Inc. (2002)