TRANSPORT PROBLEMS 2026 Volume 21 Issue 1
PROBLEMY TRANSPORTU DOI: 10.20858/tp.2026.21.1.16

Keywords: formal methods; railway signaling; relay circuits; TLA+

Michał GRZYBOWSKI[1]*, Jakub MŁYŃCZAK[2], Lucyna SOKOŁOWSKA[3], Michał MATOWICKI[4]

MODELING OF RELAY-BASED RAILWAY INTERLOCKING SYSTEMS WITH TLA+

Summary. This paper presents application of TLA+ formalism to describe and analyze relay-based railway interlocking system behavior. Many railway signaling systems constructed in the twentieth century were based on relay technology. Due to this and the longevity of relay-based interlocking systems, they are still widely used by railway infrastructure managers. Relay circuits in railway signaling systems are often used in safety-critical applications. Such applications require special considerations in terms of relay circuit design and application of appropriate relay types to guarantee proper system reaction for potential failures. Logic models can be applied to ensure safe relay circuit behavior by automatically verifying circuit operation. Relay circuit modeling has been widely studied in the literature with various tools. This paper considers the problem of modeling relay circuits in TLA+ temporal logic and the application of the model for safety analysis of the relay circuit. The TLA+ temporal logic allows us to describe the considered system by state variables and actions, which modify the system state. TLA+ formalism allows us to describe safety properties and liveness properties. This paper presents a model of a relay circuit in TLA+ logic. The presented model allows us to describe and automatically verify the safety properties of the relay circuit. The main contribution of the model is the considerations potential failures of the employed relays, which are characteristic of type N and type C relays. The model allows us to analyze circuit behavior under possible failure modes and, thus, can be used for automatic verification of the relay circuit behavior.

1. INTRODUCTION

This paper presents an application of TLA+ to describe and analyze relay circuits. Railway signaling systems were implemented using relay technology during the second half of twentieth century. Due to this and the general longevity of relay-based railway signaling systems, these systems are still widely used on various railway networks [15]. In countries such as Bulgaria, Croatia, and Romania, over 60% of all signaling systems are still relay based [15]. In Poland, type E relay interlocking is especially popular [13]. Due to their maturity and lower cost when compared to computer-based railway signaling systems, new relay-based interlockings are still being installed in some countries, including Poland. Although type E interlocking is a legacy design (the last major revision was released in 1989),

[1] Silesian University of Technology, Faculty of Transport and Aviation Engineering; Krasińskiego 8, 40-019 Katowice, Poland; e-mail: michal.grzybowski@polsl.pl; orcid.org/0000-0002-4841-147X
[2] Silesian University of Technology, Faculty of Transport and Aviation Engineering; Krasińskiego 8, 40-019 Katowice, Poland; e-mail: jakub.mlynczak@polsl.pl; orcid.org/0000-0003-2947-7980
[3] Railway Research Institute IK; Józefa Chłopickiego 50, 04-275 Warsaw, Poland; e-mail: lsokolowska@ikolej.pl; orcid.org/0000-0002-0699-4312
[4] Czech Technical University in Prague, Faculty of Transportation Sciences; Jugoslávských partyzánů 1580/3, 160 00 Prague 6 - Dejvice, Czech Republic; e-mail: michal.matowicki@cvut.cz; orcid.org/0000-0002-2630-1704
* Corresponding author. E-mail: michal.grzybowski@polsl.pl

it is still being improved. The last modifications were introduced in 2020. Construction of computer-based railway signaling systems also usually require designing new relay circuits for interfacing the newly constructed system with existing relay-based systems.

Relay circuits are often used in railway signaling systems in safety-critical applications. The relay circuits are responsible for controlling point machines and wayside signals. The incorrect operation of the devices may result in train derailment or collision. The potential consequences force a particular circuit construction and application of appropriate relay types to ensure that the system reacts properly to potential failures. To confirm safe operation of relay circuits, the design currently needs to analyze the effects of potential failures of all devices present in the circuits and to determine if potential failures do not lead to dangerous situations.

The paper discusses the problem of modeling a relay circuit in TLA+ temporal logic and the application of the model for a safety analysis of the relay circuit. TLA+ allows one to describe the modeled system by state variables and actions, which affect the values of state variables. TLA+ facilitates the description of safety properties and liveness properties.

A system of relay circuits is a set of electrical circuits built using relays. A relay is a device in which the closure/opening of relay contacts is controlled by the current flow through the relay coil. Due to the variety of applications, there are many types of relays, which differ in terms of control, contact configuration, and potential faults. This work considers typical relays employed in type E interlocking. These relays are mainly direct current relays and have two states – picked-up and dropped. The relay picks up when the current through the relay coil crosses a specific threshold. A picked-up relay closes its normally open (NO) contacts and opens its normally closed (NC) contacts. A dropped relay opens its NO contacts and closes its NC contacts.

Relays in railway signaling applications can be categorized into three groups (in accordance with standard [1]):

- Non-class – relays that are neither type C nor N – these relays should not be used for the implementation of safety-related functions.
- Type C – relays with forced contact guiding – these relays may exhibit failures such as relay dropped even though current flows through the relay coil (lack of relay activation) and relay picked-up though no current flows through the relay coil (lack of relay deactivation). The application of a type C relay requires its correct operation to be controlled by the relay circuit itself.
- Type N – relays with forced contact guiding protected from welding of NO contacts – these relays may exhibit failure of the relay dropped even though current flows through the relay coil (lack of relay activation). It is assumed that the risk of the relay not dropping after interruption of the current flow through the relay coil is negligible. The application of a type N relay does not require its correct operation to be controlled.

Relay circuits can be used to implement basic logic functions through appropriate circuit construction. A serial contact connection implements a logical conjunction, and a parallel connection implements a logical alternative. Because of this, relay technology allows one to implement any combinational logic function (a function whose outputs depend only on the inputs) and sequential logic function (a function whose outputs depend on inputs and the current function state).

Relay circuits in railway signaling systems are used for automation and interlocking. In station railway signaling systems, relay circuits implement point switching, route locking, and signaling. In line railway signaling systems, relay circuits implement traffic direction management and train separation by signaling.

2. LITERATURE REVIEW

Relay circuit behavior analysis is discussed in various contexts in the literature. In [3], Haxthausen proposes tools aiding the analysis of relay circuits. Haxthausen notes that the relay circuit analysis can be facilitated by both formal verification tools and simulators. The simulator described by Haxthausen allows one to enter a description of a relay circuit as input and to force input states and observe relay

circuit response as successive relay state combinations. In [4], Zhang describes a simulator of a station railway signaling system. The simulator described by Zhang permits an analysis of various traffic scenarios. The simulation logic is based on relay circuits of type 6502 interlocking. The main difference between this simulation and the method proposed in this work is the scope of analysis – the simulation allows one to verify circuit operation by checking its behavior in a sufficiently large number of scenarios. Model checking, as proposed by the authors, instead checks if all possible circuit operations satisfy a given property without requiring manual elaboration of all different scenarios.

In [5], Almeida considers relay circuits as a model for electronic systems. Almeida describes the formal model of relay circuit and its transformation into B method specification, which can be used to automatically generate computer program implementing functionality of the modeled relay circuit.

In [6], Cavada describes a model of relay circuit based on Switched Multi-Domain Kirchoff Networks formalism, which allows to model current flow in electrical circuits with relays. The method presented by Cavada allows to model circuits containing various types of elements and for automatic verification of model properties by using Satisfiability Modulo Theories solvers. Compared to Cavada's work, the current work presents conceptually simpler model, which does not require the analysis of current flow in various circuit fragments. In addition, the current work discusses the impact of relay type on circuit behavior, which is not discussed in the literature.

In [7,8], Haxthausen presents a method for translating relay circuits into RSL-SAL specifications and method of verifying model properties encoded as linear temporal logic (LTL) formulas. The method proposed by Haxthausen allows for the automatic generation of the model specification and for the verification of liveness and safety properties. Paper [7] presents the formal model of relay circuit, while paper [8] presents a tool for automatically translating relay circuits into corresponding models. The algorithms presented by Haxthausen are of limited use due to assumptions on the structure of analyzed circuits (for example, the algorithms do not handle the case of relay coil shorted by contacts of a different relay). In addition, the method described by Haxthausen does not consider the types of relays used in the circuits, which impact the circuit's behavior.

In [10], Amendola describes a method of modeling relay circuits in SMV language. The method proposed by Amendola allows for the automatic verification of properties encoded as LTL logic formulas with the nuXmv tool.

In [2], Salierno describes the application of TLA+ to model and verify computer-based interlocking described with state machines. Compared to that work, the current work considers the description of relay circuit behavior in greater detail, in particular, by considering potential relay failures.

Relay circuits were often a base for the development of computer-based railway signaling systems. One method for representing relay circuit functionality in computer programs is to use ladder languages, in which the program consists of variables corresponding to relays and logical expressions representing relay coil circuits. This analogy permits the application of ladder language program analysis methods for the analysis of relay circuits. In [11], Kanso describes a model of a ladder program in propositional calculus. Kanso also describes how to represent the program's initial state, single program execution (ladder scan), and how to analyze program properties encoded in propositional formulas.

For computer-based interlocking systems, alternative formal methods, such as theorem proving, may be used. An example is the B method, which was used in RATP projects [14]. The use of formal methods allows for the development of correct-by-design software. The method proposed in the current work differs from these methods in that it can be used with legacy, existing relay-based interlocking. Thus, it can be used not only in new developments but also during modifications of existing interlocking designs.

3. METHODS

3.1. TLA+ temporal logic

TLA+ [12] is a temporal logic. A temporal logic is a logic system that allows one to describe system behaviors. These types of logic extend basic predicate logic with additional operators for describing system state changes.

A classic predicate logic consists of a certain family of terms (expressions representing objects), atomic formulas created from the terms (such as $a = b$, $a \in X$), and complex formulas created from formulas connected with connectives $\vee$, $\wedge$, $\sim$ or quantified with $\forall$, $\exists$. Temporal logics introduce additional temporal operators describing system behavior in successive states. TLA+ operates on sequences of states of the considered system. Such a sequence is called a *behavior*. TLA+ introduces two operators for describing behaviors. Prefix operator $\Box$ signifies that the formula following it is satisfied in every state of the behavior. The prefix operator $\Diamond$ signifies that the formula following it is satisfied in some state of the behavior. TLA+ also introduces the notion of *action*, which is a formula containing variables of the next state, marked with an apostrophe ($'$). An example of action is the formula $x' = x + 1$, which means that variable x in the next state has a value that is one greater than the value in the current state. Actions together with temporal operators allow one to describe system behavior. For example, the formula $\Box(x' = x + 1)$ describes a system in which the variable x increases by one in each consecutive state. TLA+ also allows one to describe *stuttering* steps, in which the system state is preserved (such steps may model system state changes not captured by the specification). Usually, TLA+ specifications allow for stuttering steps, so that formalism has a simple notation for specifying them. If A is an action, then $[A]_x$ is defined as action $A \vee (x' = x)$. A typical TLA+ specification has the form of

$$Spec \triangleq Init \wedge \Box[\mathrm{Next}]_{\langle vars \rangle} \tag{1}$$

where:
$Init$ – initial state specification,
$Next$ – action specification,
$\langle vars \rangle$ – sequence of model variables.

TLA+ action specification may contain the formula UNCHANGED $\langle vars \rangle$, which is a shorthand for the following formula:

$$(v_1' = v_1) \wedge \cdots \wedge (v_n' = v_n) \tag{2}$$

where: $v_1, \ldots, v_n$ – variables in the $vars$ sequence ($\langle vars \rangle = \langle v_1, \ldots, v_n \rangle$).

Specification (1) describes a system whose initial state satisfies the formula $Init$ and in which each consecutive state either preserves the values of variables $\langle vars \rangle$ or satisfies the action $Next$. In practice, the $Init$ formula should in some way define the value of all model variables, and $Next$ is the alternative of all actions, which can be performed by the system ($Next \triangleq A_1 \vee \ldots \vee A_n$).

A set of related variables, constants, definitions, and formulas is called a *module* in TLA+ formalism. TLA+ provides tools for creating complex specifications by module extensions or by module instantiation. If module B extends module A, then it may use any variable, constant, or definition specified in module A. Module instantiation allows one to define instances of other modules. For example, if module A declares variables x, y, z and definition D expresses the relation between variables x, y, z, then module B, which extends module A, may directly use variables x, y, z and definition D, still expressing the relation between variables x, y, z. If module B defines instance IA of module A, then the instance declaration must specify the mapping of module A variables x, y, z to module B's variables ix, iy, iz, and module B may use the definition $IA!D$ to express the relation between variables ix, iy, iz.

Several tools have been created for working with TLA+ specifications. The tools include a syntax checker (SANY), a specification typesetting tool (TLATeX) and a tool for specification verification (TLC). TLC allows the user to check whether a given specification $Spec$ satisfies some property (temporal formula) $Prop$. Formally, it checks whether the formula $Spec \Rightarrow Prop$ is true. TLC may be executed in model checking or simulation mode. Two modes of operation stem from the TLC algorithms

and the complexity of temporal specification models. In model checking mode, TLC computes the entire state space and verifies whether each possible behavior satisfies the input formula. In simulation mode, TLC checks randomly selected behaviors of a given length. Simulation mode is useful in cases where the state space is infinite or too large to exhaustively check by TLC.

3.2. Relay circuit model

The model proposed by the authors considers three types of relays – type C, type N, and ideal relay. Type C relay can fail in a permanent pick-up state or a permanent drop state. Type N relay can fail only in a permanent drop state. The ideal relay cannot fail. A relay can be either activated (picked up) or deactivated (dropped). An activated relay closes NO contacts and opens NC contacts. Deactivated relay opens NO contacts and closes NC contacts. Formally, a relay is described in the module "relays" by a state variable $relay$ and constant $type$ denoting the relay type:

$$\text{VARIABLE } relay \tag{3}$$

$$\text{CONSTANT } type \tag{4}$$

Relay type can assume one of the values of $RelayTypes$ set:

$$RelayTypes \triangleq \{\text{"Ideal"}, \text{"N"}, \text{"C"}\} \tag{5}$$

A relay state is a record describing a relay position (field $relay$), state of NC contact (field nc), state of NO contact (field no), and fault (field $fault$):

$$State \triangleq [state: \{\text{"Active"}, \text{"Inactive"}\}, nc: \{TRUE, FALSE\},$$
$$no: \{TRUE, FALSE\},$$
$$fault: \{\text{"Stuck-active"} , \text{"Stuck-inactive"}, \text{"None"}\}] \tag{6}$$

Relay in the initial state can be activated or deactivated, which is expressed by the definitions $RelayActive$ and $RelayInactive$:

$$RelayActive \triangleq$$
$$relay = [state \mapsto \text{"Active"}, nc \mapsto \text{TRUE}, no \mapsto \text{FALSE},$$
$$fault \mapsto \text{"NONE"}] \tag{7}$$

$$RelayInactive \triangleq$$
$$relay = [state \mapsto \text{"Inactive"}, nc \mapsto \text{TRUE}, no \mapsto \text{FALSE},$$
$$fault \mapsto \text{"NONE"}] \tag{8}$$

A relay can perform one of four actions: activate, deactivate, become permanently activated, or become permanently deactivated. A relay can activate if it is deactivated and is not permanently deactivated. As a result of activation, it opens NC contacts and closes NO contacts. Formally, this is captured by the action $Activate$:

$$Activate \triangleq relay.state = \text{"Inactive"} \wedge relay.fault \neq \text{"Stuck-inactive"} \wedge$$
$$relay' = [relay \text{ EXCEPT } !.state = \text{"Active"}, !.nc = \text{FALSE}, !.no = \text{TRUE}] \tag{9}$$

The notation $[x \text{ EXCEPT } \ldots]$ in Formula (9) denotes a function (record), which assumes the same values as function (record) x with the exception of specified arguments (fields).

A relay can deactivate if it is activated and is not permanently activated. As a result of deactivation, it closes NC contacts and opens NO contacts. Formally, this is captured by th action $Deactivate$:

$$Deactivate \triangleq relay.state = \text{"Active"} \wedge relay.fault \neq \text{"Stuck-active"} \wedge$$
$$relay' = [relay \text{ EXCEPT } !.state = \text{"Inactive"}, !.nc = \text{TRUE},$$
$$!.no = \text{FALSE}] \tag{10}$$

The model considers two failures – permanent deactivation and permanent activation. A relay can permanently deactivate if it is not an ideal relay and is not failed. As a result of permanent deactivation, it closes NC contacts and opens NO contacts. Formally, this is captured by the action $FailStuckInactive$:

$$FailStuckInactive \triangleq relay.fault = \text{"None"} \wedge type \neq \text{"Ideal"} \wedge relay' = [relay \text{ EXCEPT } !.state = \text{"Inactive"}, !.nc = \text{TRUE}, !.no = \text{FALSE}, !.fault = \text{"Stuck-inactive"}] \quad (11)$$

A relay can permanently activate if it is neither an ideal nor a type N relay, is not failed, and is active (model assumes permanent activation because of contact welding). Formally this is captured by the action $FailStuckActive$:

$$FailStuckActive \triangleq relay.fault = \text{"None"} \wedge relay.state = \text{"Active"} \wedge type \neq \text{"Ideal"} \wedge type \neq \text{"N"} \wedge relay' = [relay \text{ EXCEPT } !.fault = \text{"Stuck-active"}] \quad (12)$$

The relay model can be used to describe the operation of a relay circuit. A relay circuit model lists all relays present in the circuit, the initial states of all relays, and the connections between them (relay coil circuits). Relays are represented in the model by state variables. A relay is introduced in the model by Declarations (13) and (14):

$$\text{VARIABLE } r \quad (13)$$

$$Relay\_r \triangleq \text{INSTANCE } relays \text{ WITH } relay \leftarrow r, type \leftarrow [\![\text{type}]\!] \quad (14)$$

where:
r – model variable representing the relay,
$[\![\text{type}]\!]$ – type of modeled relay (one of "C", "N", "Ideal").

Definition (14) introduces a new instance of module "relays" that describes the behavior of a relay represented by the r variable.

The initial state of the circuit is specified by specifying the initial state of each relay:

$$Init \triangleq Relay\_r_1!RelayInactive \wedge \cdots \wedge Relay\_r_n!RelayActive \quad (15)$$

where:
n – number of relays in the model,
$r_1, \ldots, r_n$ – model variables representing relays.

The most important part of the model is the representation of the relay circuit coils of the considered relay circuit. Consider a coil circuit of a relay r and assume that it consists of NC and NO contacts of relays $r_1, \ldots, r_n$ and that the coil circuit is a series-parallel connection of considered contacts (i.e. the circuit was constructed from series or parallel connections of contacts and series-parallel contact connections). For such circuit:

1. The NC contact of r_i relay corresponds to the expression $r_i.nc$,
2. The NO contact of r_i relay corresponds to the expression $r_i.no$,
3. The connection in a series of contacts Nx_i and Nx_j of relays r_i and r_j corresponds to the formula $r_i.nx_i \wedge r_j.nx_j$,
4. The parallel connection of contacts Nx_i and Nx_j of relays r_i and r_j corresponds to the formula $r_i.nx_i \vee r_j.nx_j$,
5. If formula F_U corresponds to circuit U and formula F_V corresponds to circuit V, then formula $F_U \wedge F_V$ corresponds to the connection of circuits U and V in series.
6. If formula F_U corresponds to circuit U and formula F_V corresponds to circuit V, then formula $F_U \vee F_V$ corresponds to a parallel connection of circuits U and V.

Intuitively, the model describes the possibility that current flows in the coil circuit of the considered relay. A current can flow through a connection of circuits in series if it can flow through both connected circuits. Formally, the formula describing the possibility of current flow through the connection of circuits in series is a conjunction of formulas describing the possibility of current flow through each

circuit. A current can flow through a parallel circuit connection if it can flow through at least one of the connected circuits. Formally, the formula describing the possibility of current flow through a parallel connection of circuits is a disjunction of formulas describing the possibility of current flow through each circuit. The assumption of the shape of the relay coil circuit is not a limitation. Any circuit consisting of relay contacts is equivalent to a series-parallel circuit, as demonstrated by Shannon [2].

In the model, for each relay *r*, a definition $Coil\_r$ is introduced according to the above rules. A relay can change its state according to the state of the coil circuit, or it can fail in the model. The model assumes that no more than one relay changes state in each consecutive state. A relay activates when its coil circuit closes and deactivates when its coil circuit opens. A change of state of relay r is described by action $Transition\_r$:

$$Transition\_r \triangleq ((Coil\_r \wedge Relay\_r!\,Activate) \vee (\sim Coil\_r \wedge Relay\_r!\,Deactivate)) \wedge \text{UNCHANGED } vars\_without\_r \tag{16}$$

where:

$vars\_without\_r$ – sequence of all model variables with the r variable.

The UNCHANGED $vars\_without\_r$ part represents the assumption of a single relay change per step – if the action $Transition\_r$ is executed, then no model variable, with the exception of *r*, should change its value. The change of state of any relay according to its coil circuit state is described by the action $Transition$:

$$Transition \triangleq Transition\_r_1 \vee \ldots \vee Transition\_r_n \tag{17}$$

where:

$r_1, \ldots, r_n$ – all relays for which the coil circuit model is specified.

The model author always has to specify the model scope (i.e. relays), which shall be accurately reflected in the model and relays (or other signals), which are to be treated as input signals and can change arbitrarily. The model assumes that input signals also change one at a time. A change of input signals is represented by the formula $InputChange$:

$$\begin{aligned} InputChange \triangleq\ & (Relay\_r_1!\,Activate \vee Relay\_r_1!\,Deactivate) \\ & \wedge \text{UNCHANGED } vars\_without\_r_1 \vee \\ & \vee \ldots \vee \\ & (Relay\_r_n!\,Activate \vee Relay\_r_n!\,Deactivate) \\ & \wedge \text{UNCHANGED } vars\_without\_r_n \end{aligned} \tag{18}$$

where:

$r_1, \ldots, r_n$ – all relays, for which the coil circuit model is not specified.

Relays for which coil circuits are represented in the model can fail. Failure does not need to be considered for other relay since the model does not impose any restriction on their behavior. Relay failures are represented by the formula $Fail$:

$$\begin{aligned} Fail \triangleq\ & (Relay\_r_1!\,Fail \wedge \text{UNCHANGED } vars\_without\_r_1) \vee \\ & \vee \ldots \vee \\ & (Relay\_r_n!\,Fail \wedge \text{UNCHANGED } vars\_without\_r_n) \end{aligned} \tag{19}$$

where:

$r_1, \ldots, r_n$ – all relays for which the coil circuit model is specified.

The definitions introduced so far are sufficient for model construction. The action $Next$ of the next step is defined as:

$$Next \triangleq InputChange \vee Transition \vee Fail \tag{20}$$

Circuit specification is defined as:

$$Spec \triangleq Init \vee \Box[\text{Next}]_{\langle r_1, \ldots, r_n \rangle} \tag{21}$$

where:

$r_1, \ldots, r_n$ – model variables representing relays.

3.3. Model application

The model presented in Section 3.2 can be used for relay circuit behavior analysis. The model allows the user to describe circuit behaviors, which are expressible as TLA+ temporal formulas. The most important properties, which can be formalized with TLA+, are safety properties (i.e., properties that can be described by specifying allowed states of the model. If P is a state predicate (Boolean-valued expression consisting of constants and variables without apostrophe character), then $\Box P$ is a temporal logic, which expresses that every state of the circuit preserves the property described by P. Properties of this type can be automatically analyzed with TLC.

Formulas of type $\Box P$ can be too strict for describing a relay circuit model. Relays in the circuit operate asynchronously – each relay responds independently to the current flow through its coil. This behavior is reflected in Formula (17) – all relays are considered equal and the $Transition$ action is satisfied by any $Transition\_r_1$, …, $Transition\_r_n$ action. Therefore, a simple predicate P expressing the relations between a relay's states do not necessarily capture the designer intent, as it does not consider the intermediate state, during which a relay has not yet responded to the change in its coil circuit state.

When analyzing a relay circuit, a designer is mostly interested in settled states (i.e., states in which the relay state is consistent with the state of its coil circuit. The proposed model and TLA+ allow for a simple description of settled states by modifying predicate P. If A is an action, then the formula ENABLED A is a state predicate, which is satisfied if the current state satisfies the conditions of action A (if A can be executed in the current state). It leads to a simple description of the settled state – the state is settled if no $Transition\_r_1$, …, $Transition\_r_n$ action can be executed. Formally, this is described by the definition $Settled$:

$$Settled \triangleq \sim \text{ENABLED } Transition \tag{22}$$

Now, the requirement that every settled state satisfies predicate P can be written as a temporal formula $\Box(Settled \Rightarrow P)$.

4. RESULTS AND DISCUSSION

4.1. Case study

The authors modeled station block circuits with type E relay interlocking to verify method correctness. The authors selected the following circuits for the case study: consent request circuits (Figs. 1 and 2), consent grant circuits ([13] diagram EI-C2a), consent return circuits ([13] diagram EI-C2a), and consent reception circuits ([13] diagram EI-C2b). The station block circuits are used to perform the interlocking between two interlocking areas located in one station. In other words, these circuits ensure that the dispatcher of interlocking area "WA" can issue traffic commands only if the system of interlocking area "WB" is in an appropriate state. The authors selected these circuits because they are non-trivial and relatively independent from other circuits of type E interlocking. The figures present circuits in their original form, as specified in [13]. The circuits in the figure are designed for operation over four station tracks. For the analysis, the authors modeled circuits for operation over one track.

The authors used TLC for the analysis. During the research, the authors considered three main variants of the model: model of only consent request circuits (Figs. 1 and 2), model of consent request circuits, consent grant circuits, consent return circuits and consent reception circuits (Figs. 1, 2, 3, 4, 5) and a model of consent request circuits, consent grant circuits, consent return circuits, and consent reception circuits extended with one route in each interlocking area dependent on the station block conditions designed according to type E interlocking album [13].

4.2. Model analysis with TLC

The authors constructed a series of models of circuits presented in Section 4.1. The models differed in terms of complexity:

1. Consent request circuits model
2. Consent request, consent grant, consent reception and consent return circuits model
3. Consent request, consent grant, consent reception, and consent return circuits model with the assumption of lack of possible faults for a subset of relays
4. Consent grant, consent reception, and consent request circuits model
5. Consent request, consent grant, consent reception, consent return circuits, and signaling relay and locking relay circuits for one route in each interlocking area model

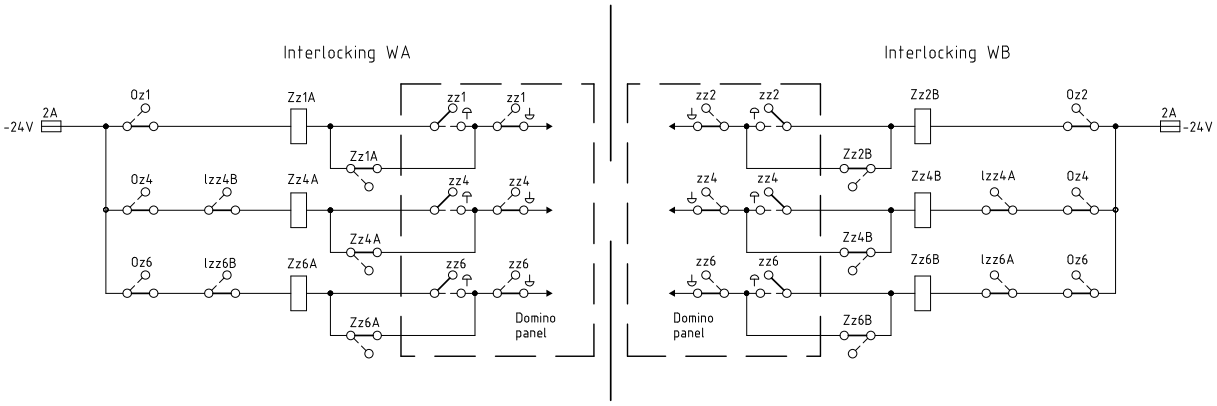

Fig. 1. Consent request circuits of station block in type E interlocking (source: [13] diagram EI-C1)

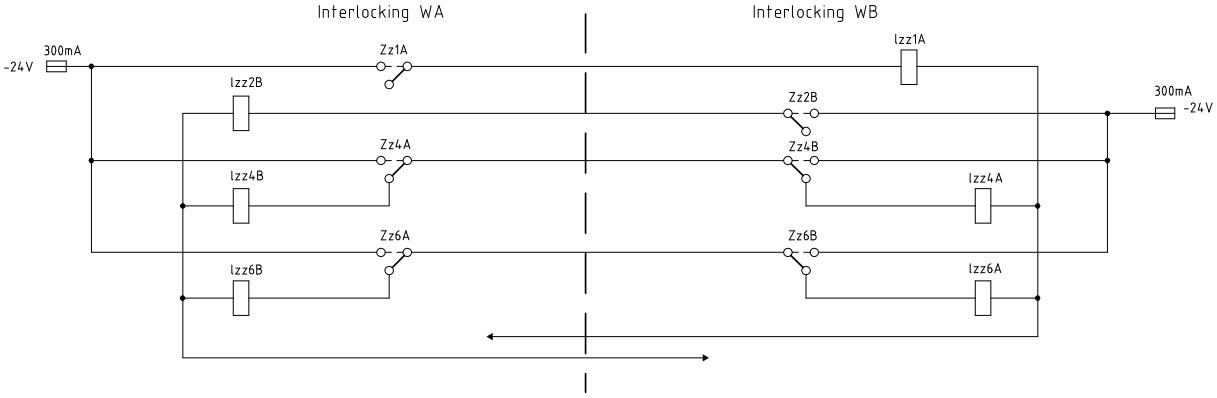

Fig. 2. Line consent request circuits of station block in type E interlocking (source: [13] diagram EI-C1)

Table 1 summarizes the model's complexity. Two-state buttons were modeled as an ideal relay, and three-state buttons were modeled as a pair of ideal relays. The column 'Number of modeled coils' specified the number of relays for which the coils were modeled. Other relays are treated as input signals.

Safety properties can be analyzed by formulating the required properties as a temporal formula of the form $\Box P$ and verifying if the model satisfies it. The consent request circuits model was analysed to determine if it satisfies the formula $\Box(Settled \Rightarrow \sim(lzza.no \wedge lzzb.no))$, which formalizes the requirement that relays lzzA and lzzB cannot be active at the same time (it should be possible for both dispatchers to simultaneously request consent from each other). Theoretically, this property is satisfied by the design of lzzA and lzzB relay coils' circuits, but it can be violated due to a fault because non-class relays are used (in the model, class C relays were used instead of non-class relays to simplify the model). The possibility of entering a state violating this property was automatically detected by TLC (Fig. 3). This example confirms that TLC is fit for the analysis of models proposed in this paper.

Table 1

Characteristics of models constructed during research

Model	Ideal relays	Type C relays	Type N relays	Modeled coils
1	4	4	0	4
2	10	18	0	16
3	6	14	0	12
4	24	4	0	16
5	12	22	2	22

The proposed model allows for the automatic analysis of circuits' properties. During the research, the authors considered three applications of the model:

1. Analysis of safety properties
2. Analysis of the conditions to enter selected states
3. Analysis of circuit sensitivity to types of relays used in the circuit

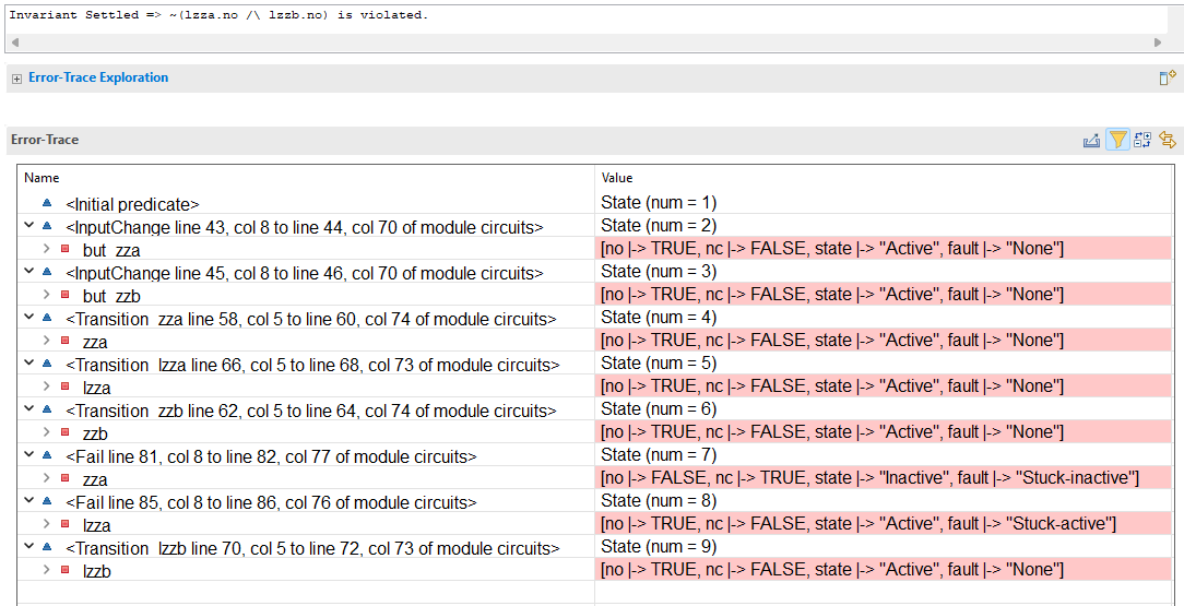

Fig. 3. Circuit model behavior leading to a violation of the considered temporal formula (source: original work)

TLC presents a sequence of states causing the formula to be not satisfied if such behavior exists. This functionality allows TLC to be used for the analysis of circuit dynamics. If the user is interested in finding out how the circuit can enter a certain state, then the user can describe the state as formula P and use TLC to verify if the model satisfies the temporal formula $\Box(\sim P)$. TLC performs breadth-first search. Thus, it will present a sequence of states leading to a state satisfying formula P. This procedure employs the *trap property* of model checkers, as described by [9].

Model construction and the explicit specification of relay types allow the user to verify the impact of relay types on the circuit behavior. The user may select a set of considered safety properties and observe analysis results for models differing only by the types of relays present in the analysis to find out how they impact the satisfaction of considered properties.

4.3. Analysis time

The authors measured the TLC analysis time to determine the impact of the model's complexity on the analysis time. Measures were performed for analyzing the temporal formula not satisfied by the

model – formula $\Box(Settled \Rightarrow \sim(lzza.no \wedge lzzb.no)$ for the first model and formula $\Box(Settled \Rightarrow \sim(oza.no \wedge ozb.no)$ for the rest of the models. The formulas used for analysis also confirm that the model correctly considers potential relay faults. Given the circuits in Fig. 2 and the consent reception circuits ([13] diagram EI-C2b), it can be concluded that the formulas used for the analysis are not satisfied only due to the possibility of relay failures (permanent activation of lzzA or lzzB relay in case of the first formula and permanent activation of OzA or OzB relay in case of the second formula). The authors also measured the time necessary to check the entire state space of the model. This was done by asking TLC to verify a formula satisfied by the model – formula $\Box(lzza.no \vee \sim lzza.no)$ for models 1, 2, and 3 and formula $\Box(oza.no \vee \sim oza.no)$ for model 4.

Analysis was conducted on a computer equipped with an AMD Ryzen 5 8600G CPU and 32 GB of RAM. TLC was configured to use ten threads, a working area in RAM of 14 340 MB, and 200 GB of storage space. Tables 2 and 3 present the analysis time. The tables describe each model's analysis run time, the number of states generated by TLC during analysis, the number of unique states checked during the analysis, and the diameter of the analyzed state space. Diameter is the maximum of the shortest paths leading from the initial state to each state checked during the analysis. In the model, diameter can be considered the maximum number of relay state changes and relay faults in behaviors considered by TLC.

Table 2

Summary of the analysis of properties not satisfied by the model

Model	Analysis run time [s]	Number of states generated	Number of unique states checked	State space diameter
1	1	13 985	2400	10
2	1408	2 067 175 680	228 520 484	12
3	58	101 299 296	11 254 041	12
4	15	14 853 547	1 760 563	13
5	(abnormally terminated after three hours)	(14 260 197 104)	(1 628 396 603)	(12)

Table 3

Summary of exhaustive state space analysis

Model	Analysis run time [s]	Number of states generated	Number of unique states checked	State space diameter
1	1	32 257	4 096	19
2	(abnormally terminated after 5 hours)	(24 184 995 910)	(2 157 026 990)	(15)
3	48 139	62 360 911 873	3 221 225 472	45
4	21 690	32 812 040 193	1 610 612 736	49

The analysis of unsatisfied properties for model 5 was abnormally terminated due to exhaustion of the storage space (200 GB of disk space was allocated for the analysis). Exhaustive state space analysis for model 2 was abnormally terminated for the same reason. Exhaustive state space analysis for model 5 was not attempted due to the earlier abnormal termination of the analysis of not satisfied property for this model. The authors attempted the verification of not satisfied property for model 5 with TLC in simulation mode. The authors terminated the analysis in simulation mode after 66 hours and 179 617 707 449 generated states. TLC did not manage to find a state falsifying the input formula during this time.

The measurements indicate that finding a counterexample requires less time and resources than exhaustive state space analysis. In case of simple models, the time is similar, but for complex models,

the time required to find a counterexample may be one thousandth of the time needed for exhaustive state space analysis.

The measurements also indicate that both analyses' run time and state space size depend on the number of relays considered in the model. A similar dependency exists for the number of relays of each type. Models 2 and 4 contain the same number of relays and an identical description of relay coils' circuits. The types of relays used impact the time needed to find a counter-example – the time was reduced from 1408 seconds to 15 seconds. A similar conclusion can be drawn from the data obtained for models 3 and 4. Model 3 contains eight fewer relays than model 4, but the time needed to find the counter-example for model 3 was four times greater than the time needed to find the counter-example for model 4. The time needed to exhaustively check the state space for model 3 was more than two times greater than the time needed to exhaustively check the state space for model 4. This suggests that the deciding factor in the analysis run time is the number of type C or type N relays in the model.

4.4. Discussion

The experiment conducted in this research confirms the model's construction correctness, as the results obtained from TLC matched our expectations. The analysis time and results obtained from TLC indicate that the method can be used in practice for verification of circuits containing at most around 18 type C or type N relays and 10 input signals or relays for which faults are not considered.

The analysis times for models 2, 3, and 4 indicate a dependency between analysis run time and the number of relays in the model, especially type C relays, which have the greatest number of states.

The case study confirms the method's viability for the analysis of relay railway signaling systems used by PKP PLK or systems of similar complexity. The analysis run time indicates that the method is sufficient for the analysis of selected circuits (e.g., standard circuits from the albums of relay diagrams), but it is not applicable to the analysis of models of the entire interlocking.

The main disadvantage of the proposed method is the low complexity of the models, which can be analyzed in practice. Further research should investigate whether problem encoding can be modified to allow more complex circuits to be analyzed.

Another point for further research is the verification of interlockings built mainly with type C relays. Safety properties, as formulated in this work, can be successfully verified for circuits built with type N relays. The usual practice for designing circuits with type C relays is to introduce the checking of correct relay operation in the relay circuits. Authors have not considered how to encode this property (checking and detection of type C relay fault by the relay circuits) as TLA+ formulas and consider it an open problem.

The last point for further research is the incorporation of the proposed method into the actual design process. The development of railway signaling systems in Europe should follow the RAMS framework set by EN 50126-1 standard. Additional consideration should be given to placing the proposed method in the RAMS framework. Tools used for designing signaling systems are subject to the EN 50716 standard. In the authors' opinion, tools for working with TLA+ specifications, such as TLC, should be considered as T2 tools according to EN 50716, but the tools should be subject to proper validation before being used in actual projects.

5. CONCLUSIONS

Railway signaling systems are implemented with different technologies. However, due to the widespread adoption of relay-based railway signaling systems, even computer-based railway signaling systems need to use relay technology. Relay circuits often implement safety-critical functions, meaning their verification and validation require additional effort. The standard means of circuit verification (FMEA) is manual and labor-intensive. This paper investigates the use of formal methods to establish relay circuit correctness.

The paper presents a method of constructing a model of a relay circuit in TLA+. TLA+ is a tool for specifying system behavior in temporal logic. TLA+ is usually applied for modeling software systems.

The authors show that TLA+ may be successfully applied to model the behavior of relay circuits. The authors describe a model of a single relay and present how the model can be used to compose a model of a set of interdependent relay circuits. Although the authors created the model manually, the model is derived from the considered relay circuits and, in principle, could be created mechanically based on the description of the considered relay circuits.

The novelty of the model proposed by the authors stems from the circuit characteristics, which are captured by the model. The model allows for an automated analysis of reachable model states (states of modeled relays), both in fault-free operation and in the face of potential faults. The authors present how relay types (type N and type C) can be considered during model construction to allow discovering system states, which occur as a result of relay faults.

The authors investigated the usage of TLA+ tools to analyze relay circuit models described in the paper. TLA+ tools (TLA+ toolbox and TLC) allow for the automatic verification of the syntactic correctness of the model specification and for automatic model checking. Model checking, as implemented by TLC, allows for the verification of circuit safety properties. The safety properties describe states that should never be entered by the model.

The authors verified the ability of TLC to check the proposed model on a subset of type E interlocking circuits related to station block (inter-interlocking blocking). The experiments confirmed that TLC correctly determines whether the relay circuit model preserves the analyzed properties. The experiments also showed that the model is amenable to model checking with TLC on small models, which naturally arise as elements of albums of typical relay circuit designs. The model is not amenable to model checking of entire relay-based railway signaling systems due to the large number of states of the model.

The experiments conducted by the authors confirm that the method may be used for analyzing railway signaling relay circuits. The method may be used for checking typical (template) relay circuit schemes, but it does not scale to the analysis of relay circuits for the entire interlocking. The experiments also validate the correctness of the {results obtained with the TLC.

References

1. *IEC 62912:2015. Railway applications - Direct current signalling monostable relays of type N and type C*. Geneva: International Electrotechnical Commision. 29 p.
2. Salierno, G. & Morvillo, S. & Leonardi, L. & Cabri, G. Specification and verification of railway safety-critical systems using TLA+: A Case Study. In: *2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE*). Bayonne. 2020. P. 207-212. DOI: 10.1109/WETICE49692.2020.00048.
3. Haxthausen, A.E. Towards a framework for modelling and verification of relay interlocking systems. In: *Foundations of Computer Software*. Redmond. 2010. P. 176-192. DOI: 10.1007/978-3-642-21292-5_10.
4. Zhang, L. Railway signal interlocking logic simulation system. In: *ICCCV '19: Proceedings of the 2nd International Conference on Control and Computer Vision*. Jeju. 2019. P. 3-7. DOI: 10.1145/3341016.3341017.
5. Almeida, D. *Analysis and formal specification of relay-based railway interlocking system*. PhD thesis. Champs-sur-Marne: Université Gustave Eiffel. 2020. 168 p.
6. Cavada, R. & Cimatti, A. & Mover, S. et al. Analysis of relay interlocking systems via SMT-based model checking of switched multi-domain Kirchhoff networks. In: *Proceedings of the 18th Conference on Formal Methods in Computer-Aided Design (FMCAD 2018)*. Austin. 2018. P. 179-187. DOI: 10.23919/FMCAD.2018.8603007.
7. Haxthausen, A.E. & Le Bliguet, M. & Kjær A.A. Modelling and verification of relay interlocking systems. In: *Foundations of Computer Software: Future Trends and Techniques for Development*. Springer. 2008. P. 141-153. DOI: 10.1007/978-3-642-12566-9_8.

8. Haxthausen, A.E. & Kjær, A.A. & Le Bliguet, M. Formal development of a tool for automated modelling and verification of relay interlocking systems. In: *FM 2011: Formal Methods*. Limerick. 2011. P. 118-132. DOI: 10.1007/978-3-642-21437-0_11.
9. Gargantini, A. & Heitmeyer, C. Using model checking to generate tests from requirements specifications. *SIGSOFT Softw. Eng*. Notes. 1999. Vol. 24. No. 6. P. 146-162. DOI: 10.1145/318774.318939.
10. Amendola, A. & Becchi, A. & Cavada, R. et al. NORMA: a tool for the analysis of Relay-based Railway Interlocking Systems. In: *Tools and Algorithms for the Construction and Analysis of Systems, 28th International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Munich.* 2022. P. 125-142. DOI: 10.1007/978-3-030-99524-9_7.
11. Kanso, K. & Moller, F. & Setzer, A. Automated verification of signalling principles in railway interlocking systems. *Electronic Notes in Theoretical Computer Science*. 2009. Vol. 250. P. 19-31. DOI: 10.1016/j.entcs.2009.08.015.
12. Lamport, L. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Pearson Education Inc. 2002. 364 p.
13. Szeniawski, Z. & Makała, J. *Album schematów przekaźnikowych urządzeń zabezpieczenia ruchu kolejowego typu E*. Centralne Biuro Projektowo-Badawcze Budownictwa Kolejowego. 1989. [In Polish: *Album of relay diagrams of E type railway signalling system*. Central Railway Design and Research Office.]
14. Bonvoisin, B. 25 years of formal methods at RATP. In: *International Railway Safety Council (IRSC)*. Paris. 2017.
15. Bădău, F. Railway interlockings – a review of the current state of railway safety technology in Europe. *Promet – Traffic & Transportation*. 2022. Vol. 34. No. 3. P. 443-454. DOI: 10.7307/ptt.v34i3.3992.

Received 25.05.2024; accepted in revised form 04.03.2026