J. Mod. Transport. (2016) 24(4):277–283
DOI 10.1007/s40534-016-0119-1

www.springer.com/40534

Verifying the accuracy of interlocking tables for railway signalling systems using abstract state machines

Basri Tugcan Celebi[1] · **Ozgur Turay Kaymakci**[1]

Received: 31 May 2016 / Revised: 28 September 2016 / Accepted: 30 September 2016 / Published online: 27 October 2016

Abstract Railway transportation system is a critical sector where design methods and techniques are defined by international standards in order to reduce possible risks to an acceptable minimum level. CENELEC 50128 strongly recommends the utilization of finite state machines during system modelling stage and formal proof methods during the verification and testing stages of control algorithms. Due to the high importance of interlocking table at the design state of a signalization system, the modelling and verification of interlocking tables are examined in this work. For this purpose, abstract state machines are used as a modelling tool. The developed models have been performed in a generalized structure such that the model control can be done automatically for the interlocking systems. In this study, NuSMV is used at the verification state. Also, the consistency of the developed models has been supervised through fault injection. The developed models and software components are applied on a real railway station operated by Metro Istanbul Co.

Keywords Model checking · Abstract state machines · Interlocking

1 Introduction

The importance of railway transportation is increasing day by day as the world becomes more populated. European Rail Infrastructure Masterplan (ERIM) Final Report states that the traffic growth forecast for the period of 2006–2020 is 34 % for the passenger traffic and 57 % for the freight traffic. It is also reported that the tackling all infrastructure issues would require about 200 billion euros for the entire ERIM network and about 60 billion euros for the six ERTMS corridors [1]. Such an increasing level of traffic congestion and need for infrastructure reveal many new topics of research in this area.

The interlocking is a subsystem that avoids conflicting movements of vehicles by adjusting the routes. The safety critical software of the interlocking system is generally composed of two different parts: generic application and specific application. The generic application part is usually designed with the general structure. On the other hand, the specific application part is obtained based on the site topology using this developed general application components. Certification stage of a railway signalization project generally includes this specific application part because general application part has been already certified for one time only. At this point, interlocking tables are taken as a reference in order to obtain specific application from general one.

In general, an interlocking table is a list which includes all possible routes at the signalization system, and it defines the situations of the signalling components in order to open the routes safely. Sometimes the design of the interlocking table is a difficult process for complicated field topologies. A very slight failure means that a route will be opened in a different condition than it should be. This could lead to many possible different dangerous situations including derailment or accident. For all these reasons, the interlocking table have to be verified. CENELEC EN 50128 strongly recommends the use of formal methods for SIL4 safety level in Table A-4 such that model checking is listed among those formal methods.

✉ Ozgur Turay Kaymakci
kaymakci@yildiz.edu.tr

[1] Department of Control and Automation Engineering, Faculty of Electrical and Electronics Engineering, Yildiz Technical University, Esenler, Istanbul, Turkey

The advantage of model checking is to detect the failures during the design stage. This positive aspect makes it possible to identify and correct these failures which would be very hard or even impossible to rectify during the design stage. On the other hand, Simulation methods could also be considered as another option for the verification. However, some studies have revealed that there is a residue risk of not being able to detect all the probable failures through simulation [2].

Many studies can be found in the literature regarding the design of railway signalling and interlocking systems through the use of formal methods. Winter designed a model through CSP (Communicating Sequential Processes) formal modelling language and performed the model checking through FDR (Failures-Divergences Refinement) [3, 4]. Similarly, Simpson presented some models for British Railway Systems in accordance with GDL (Geographic Data Language) notation as CSP, and utilized FDR as the model checking software [5]. In a further study, Winter formed a model through ASM (Abstract State Machine) notation which is a similar language to Robinson and state-transition diagrams. She performed the model check through NuSMV [6]. It has been maintained that ASM notation has a more comprehensible expression than models performed through CSP regarding the system structure and performing state transitions with dynamic properties through the same notation [7, 8]. In another study, the model was formed through GCL (Guarded Command Language) notation and modelled through CSP for the correspondence of the processes [9]. CPN (Coloured Petrinet) was used in the modelling of the interlocking system formed in Anunchai State Railway of Thailand format, and Panthong station was used as the exemplary system [10]. CPN models are distinct from Petri Net models as the tokens have specific properties [11].

Distinct from the other studies carried out, a general modelling structure concerning the general properties of station topology and interlocking table has been formed in this study. A generalized software base design has also been performed to be able to automatically check this model through NuSMV software. A general structure of the interlocking system has been analysed, and a general ASM model structure has been obtained. At the same time, conversion systematic has been revealed so that the ASM model of the interlocking table can be used with NuSMV software. In this way, it has become possible to establish an infrastructure that can enable the testing of interlocking table automatically.

The study consists of four sections. Section 2 defines the basic definitions of model checking, and Sect. 3 mentions the model checking of the interlocking table on the basis of railway station operated by Metro Istanbul Co. Finally, the paper is concluded in Sect. 4.

2 Model checking modelling

Recently, model checking was put forth as a result of studies carried out by Edmund M. Clarke and Allen Emerson as well as through J.P Quielle and J.Sifakis independently [12, 13]. As a formula, with M a Kripke structure and f a temporal logic, model checking is defined as the study of all s states that are $s := f$ in M structure [14].

It is possible to express the behaviour of a system with a modelling technique such as petri nets, state-transition diagrams or abstract state machines (ASM). As complex topologies are not applicable in practice due to the problem of state explosion, modelling the system behaviour as unsophisticated as possible is very important [15]. Different solutions for the state explosion problem have been offered in literature [16, 17]. On the other hand, forming the models in a simplest way as possible is always a big advantage in industrial applications. Hence, the possible solutions can be achieved easily and system requirements can be identified accurately.

States are the set of values which are the result of functions. Functions are split into distinct groups based on their interrelations and assessments. "Static function" has an assessment is for static functions, "dynamic function" has an assessment that could vary based on the model and "external function" has an assessment that may change as a result of external factors independent from the model.

Transitions define how dynamic functions will vary while going from one state to another. Such operations can be conditioned through transition rules in ASM. In addition, transitions are realized in a synchronized way concurrently based on the transition rules.

Updating Rule Enables the $(f, (v_1,\ldots,v_n))$ state to be updated according to the v value.
Conditioning Rule If *"cond"* condition, defined as logic, is true, R_t is applied, if it is not true, then R_e is valid

If *cond* **then** R_t **else** R_e **endif**

Forall Rule If the logic *"cond"* condition is true for any $v \in D$, R is verified for v values

Forall v **in** D **with** *cond* **do** R

After forming the model of the system, the requirements to be ensured and the propositions to be used for supervising the possible system failures are represented through temporal logic formulae [18].

Computational tree logic (CTL) is a form of temporal logic to express temporal logic formulae which can be checked by model checking software. CTL formula involves temporal expressions, with φ and Ψ being atomic propositions, $X\ \varphi$ (next φ), $\varphi\ U\ \Psi$ (φ atomic proposition is valid until Ψ), $G\ \varphi$ (φ is always globally valid) and $F\ \varphi$ (φ

is valid in the future). Boolean logic operators like ¬, ∧, ∨, which means, respectively, "not", "and", "or", "if", "if and only if" are included. In addition to these, path quantifying operators are also defined, which are "A" (for all probable paths) and "E" (for some probable paths). For further details regarding CTL formulization, it is recommended that the reader goes through referred studies [19].

3 Model checking of an urban railway interlocking system

Interlocking system is the core subsystem of railway signalling systems such that it defines the exact conditions of the field equipment in order to manage the vehicle movements safely. For detailed information about railway signalization systems refer to [20]. In this work, the considered system is an urban interlocking system that is operated under double-track operation such that each track is operated exclusively in one direction. In this context, the safety rules that cover the predetermined conditions of the blocks, points and signals within the system are examined. The examined set of conditions are directly linked with the elements contained by the desired route, system topology and states of the routes previously opened. It is obvious that interlocking system structures may vary depending on countries, operating companies, topology and different purposes of use but the basic safety rules remain the same everywhere. This study concentrates on these basic safety rules such that it aims to introduce a new model checking strategy by introducing a core model using the most existing elements in any urban interlocking system.

Three distinct elements have been taken into consideration in the modelling phase. These elements are topology, interlocking table and train motion. While the topology model includes the states and positions of the blocks, points and signals, the interlocking table model comprises the basic requirements for opening all the possible routes. On the other hand, the train motion model represents the vehicle movement according to the signalization system.

A railway station, operated by Metro Istanbul Co., is taken as a reference for this study. The layout of the station is given Fig. 1. The station topology includes blocks, points and antennae. Route inquiries are transmitted to the interlocking system through antennae by the conductors. Signals can be categorized into three, which are straight line, vertical line and side line. A straight line indicates that the train goes along the route straightly, while a vertical line shows that train transitions are not allowed. Finally, a side line informs that the route is formed in such a way that the train diverges over the point. The positions of the points vary depending on the route. The track circuits are positioned following the signals in order to get the information if a vehicle occupies the corresponding block or not. Here expressing the interlocking table perfectly is of vital importance.

Here the modelling is initiated by taking the static structure of the system as a reference. This structure is designed as static functions in ASM model in such a way that a subordinate space for all kind of elements is identified. It is also important to note that the interlocking table is identified when all the elements of the system are totally functional. Here the mechanical or electrical failure cases are ignored while forming the system model.

Track: {Track*x*, Track1, Track11, Track2, Track21}

- TrackName: {TC*x*, TC1, TC11, TC2, TC21}
- Point:{Point*x*, Point1, Point 11, Point2, Point 21}
- PointName:{P*x*, P1, P11, P2, P21}
- PointPosition:{Normal, Inverse}

The accepted case is that the positions of these points are concurrently diverging and concurrently normal.

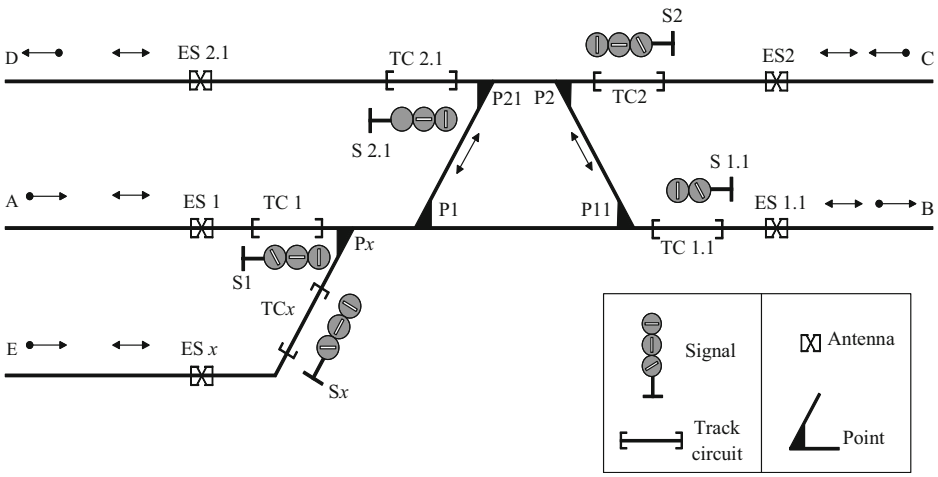

Fig. 1 Layout of the railway station operated by Metro Istanbul Co.

Naming for the blocks is incorporated into the model as a static function, while occupation is handled as a dynamic function. As the states of the signals vary depending on the opened routes and the occupation of the rail blocks, they are depicted as static functions.

The starting positions of the trains have been set randomly and route requirements have also been chosen randomly, ensuring the scanning of all possible states. The names and directions of the trains are defined as static functions, while the position of the train is identified as a dynamic function. External functions have been used for nondeterministic behaviours mentioned.

static function TrackID: Track → TrackName
dynamic function Occ: Track → Boolean
dynamic function SigColour: Signal → SignalColour
external function TrainRequestID: Train → AntennaName
external function Train_InitSituation: Train → AntennaName

The base of the model is formed upon the completion of the functions. The structure that ensures the model to evolve (transitions between the states) can be made possible through transition rules as mentioned earlier. Identification of the transition rules has ensured signalling, checking the points and modelling the train motion, depending on the conditions of the interlocking system. One part of the interlocking table designed for railway station is provided in Table 1.

Route column includes the route number, and it also refers to starting track circuit and the ending track circuit of that specific route. Blocks indicate which blocks are included in that route. Point and signal columns define the proper situation of the points and the signals for that specific route. As it is obviously seen in the "signals" column, these conditions are not deterministic. Furthermore, the counter routes column is to point out the routes which of them should be locked, while the route which is taken into account is assigned. Signals are redefined as R (red), G (green) and Y (yellow) which replaces vertical, horizontal and side lines, respectively, to make the table more understandable.

Routes and points are not locked at the initial stage. All the signals are defined as vertical line (red), not allowing any passage. Route requirements and starting states are identified randomly in the model.

The system elements are assigned a value based on the interlocking table. For instance, if a train requires the route R1 and if the requirements for the opening of the route have been fulfilled, (which means the rail blocks are available and no counter route is locked), then that particular route is locked. After having the route locked, points are arranged and locked. Afterwards, the signal state changes according to the interlocking table. Also, the motion of the train is determined based on the signal and point states and the moving direction.

```
if TrainFrontSituationID(Train1) = TC21
 then if TrainDirection(Train1) = Right
   then if (SigColour(Signal21) = Green or
SigColour(Signal21) = Yellow)
     then if (PointPosition(P2) = Normal and
PointPosition(P21) = Normal
       then TrainFrontSituationID(Train1) := TC2
```

One of the primary aims of this study is to establish relations between ASM and NuSMV notations. Static and dynamic functions defined in ASM model can be expressed through variables defined, respectively, as FROZENVAR and VAR in NuSMV. Definition sets and value sets where functions are defined can be identified as the subobjects of the parent objects which are named as MODULE in NuSMV. Parent objects serve as structures through which several elements can be defined. Here transition rules are depicted as "next" and initial states are stated as "init". The relation between ASM and NuSMV is given over point model in Table 2.

Requirements include propositions that are necessary for the system to provide. In order to define the requirements, CTL notation has been preferred in this study. The CTL formula given below has been used to check the collisions of the trains.

SPEC NAME Collision: = AG!(Train1. SituationID = Train2. SituationID | Train1. NextSituationID = Train2. SituationID |Train1. SituationID = Train2. NextSituationID);

Also, fault injection technique has been utilized, and a failure has been intentionally incorporated into the system in order to test the accuracy of the introduced model. For this purpose, the state without "Route1" from the counter routes has been injected as a failure in the interlocking table for Route9. A counter example for this situation is presented below. The ASM and NuSMV code are presented below.

Specification AG! ((Train1. SituationID = Train2. SituationID | Train1. NextSituationID = Train2. SituationID) | Train1.SituationID = Train2.NextSituationID) is false.

The corresponding NuSMV execution sequence is as follows.

→State: 1.1←
Track*x*.TrackID = TC*x*
Track1.TrackID = TC1
Track11.TrackID = TC11
Track2.TrackID = TC2
Track21.TrackID = TC21
Antennax.TrackID = ES*x*
Antenna1.TrackID = ES1

Table 1 Designed interlocking table

Route	Blocks	Points	Signals	Counter routes
R1: TC*x* → TC11	TC*x*, TC11	P*x*: Inverse	S*x*: Y	All/{R1, R6, R10}
		P1: Normal	S1: R	
		P11: Normal	S21: {G,R}	
			S2: {G,R}	
R2: TC*x* → TC2	TC*x*, TC2	P*x*: Inverse	S*x*: Y	All
		P1: Inverse	S1: R	
		P2: Normal	S21: R	
		P21: Inverse		
R3: TC1 → TC11	TC1, TC11	P*x*: Normal	S*x*: R	All/{R3, R6, R10}
		P1: Normal	S1: G	
		P11: Normal	S21: {G,R}	
R4: TC1 → TC2	TC1, TC2	P*x*: Normal	S*x*: R	All
		P1: Inverse	S1: Y	
		P2: Normal	S21: R	
		P21: Inverse		
R5: TC21 → TC11	TC21, TC11	P11: Inverse	S21: Y	All
		P2: Inverse	S2: R	
		P21: Normal		
R6: TC21 → TC2	TC21, TC2	P21: Normal	S21: G	All/{R6,R1,R3}
		P2: Normal		
R7: TC11 → TC21	TC11, TC21	P21: Normal	S11: Y	All
		P2: Inverse	S2: R	
		PI1: Inverse		
R8: TC2 → TC*x*	TC2, TC*x*	P*x*: Inverse	S2: Y	All
		P1: Inverse	S21: R	
		P2: Normal	S1: R	
		P21: Inverse	S11: R	
R9: TC2 → TC1	TC2, TC1	P*x*: Normal	S2: Y	All/{R1}
		P1: Inverse	S21: R	
		P2: Normal	S1: R	
		P21: Inverse	S*x*: R	
R10: TC2 → TC21	TC2, TC21	P21: Normal	S2: G	All/{R10, R1, R3}
		P2: Normal		
R11: TC11 → TC*x*	TC11, TC*x*	P*x*: Inverse	S11: G	All/{R11, R6, R10}
		P1: Normal		
		P11: Normal		

Antenna11.TrackID = ES11
→State: 1.2←
Pointx.Position = Inverse
Pointx.Locked = TRUE
Point1.Locked = TRUE
Point21.Locked = TRUE
Point2.Locked = TRUE
Point11.Locked = TRUE
Route1.Locked = TRUE
Route1.Free = FALSE
→State: 1.3←
Point1.Position = Inverse
Point21.Position = Inverse
Signal*x*.Colour = Yellow
Train1.NextSituationID = TC*x*
Route9.Locked = TRUE
Route9.Free = FALSE
Route10.Free = FALSE
→State: 1.4←
Trackx.Occ = TRUE
Antennax.Occ = FALSE
Signal2.Colour = Yellow
Train1.SituationID = TC*x*
Train1.NextSituationID = TC2

Table 2 ASM and NUSMV relation over a point model

ASM	NuSMV
Point:{Pointx, Point1, Point11, Point2, Point21} *-PointName:{Px, P1,P11, P2, P21}* *-PointPosition:{Normal, Inverse}* *static function PointID : Point --> PointName* *PointID(Pointx):=Px* *if cond then upd*	***MODULE** Point* ***FROZENVAR** PointID: {Px,P1,P11,P2,P21};* ***VAR** Position : {Normal,Inverse};* ***VAR** Locked: boolean;* ***MODULE** main* ***VAR** Pointx: Point* ***ASSIGN*** ***init**(Pointx.PointID) := Px* ***next**(Data) := **case** cond **:** upd* *TRUE **:** Data* ***esac***

Train2.NextSituationID = TC2
→State: 1.5←
Track*x*.Occ = FALSE
Track2.Occ = TRUE
Antenna2.Occ = FALSE
Signal*x*.Colour = Red
Train1.SituationID = TC2
Train2.SituationID = TC2

Collision of trains simply means that two trains are on the same rail block at the same time. As can be seen from the above ASM or NuSMV code, although Route1 and Route9 are counter routes, and not defined in the interlocking system, it has been able to lock Route9, while Route1 is already locked. This error has resulted with the case that trains can be on the same rail block in State 1.5.

Derailment is another condition that should be checked. If a route is opened for one of the trains but point positions are not correctly arranged, then a possible derailment can happen. CTL formularization identified for the detection of such cases is as follows:

SPEC NAME Derailment1: = AG! (Train1.SituationID = TC1 & Train1.Direction = Right & Point*x*. Position = Inverse);

SPEC NAME Derailment2: = AG! (Train1. SituationID = TC*x* & Train1. Direction = Right & Point*x*. Position = Normal);

The requirement Derailment2 states that when the train is on TC21 rail block with the direction toward right, point P*x* has been set normal as a failure or not is checked out. Probable failures have been incorporated into the system through fault injection technique. It has also been checked if the model, developed structurally, is able to detect the failures or not. Thus, system reliability has been boosted. The specification and its counter example in NuSMV structure are provided below.

Specification AG! ((Train1.SituationID = TC*x* & Train1.Direction = Right) & Point*x*. Position = Normal) is false
→State: 1.1←
Track*x*.TrackID = TC*x*
Track1.TrackID = TC1
Track11.TrackID = TC11
Track2.TrackID = TC2
Track21.TrackID = TC21
Antennax.TrackID = ES*x*
Antenna1.TrackID = ES1
→State: 1.2←
Pointx.Locked = TRUE
Point1.Locked = TRUE
Point21.Locked = TRUE
Point2.Locked = TRUE
Point11.Locked = TRUE
Route1.Locked = TRUE
Route1.Free = FALSE
Route2.Free = FALSE
Route8.Free = FALSE
Route9.Free = FALSE
InitTrack_1 = ES21
Request_1 = ES2
Request_2 = ES21
→State: 1.3←
Point*x*.Position = Inverse
Signal*x*.Colour = Yellow
Train1.NextSituationID = TC*x*
→State: 1.4←
Track*x*.Occ = TRUE
Antenna*x*.Occ = FALSE
Point*x*.Position = Normal
Train1.SituationID = TC*x*
Train1.NextSituationID = TC11

As a result of counter example analysis, it can be seen that when the train is in TC*x* position, point P*x* comes to the normal position. This situation would lead to derailing of the train. In this situation, the train arrives TCc but could not proceed to the next state P*x* because of the position misalignment of point P*x*. This case clearly shows that the problem lies with Route1. Here the consistency of the generated model is inspected, and the adequacy of it is shown with a counter example and fault injection method. It is seen that the developed models can be easily used in verification of interlocking tables.

4 Conclusion

This study reveals the modelling of interlocking system and the physical structures of the urban railway system. It also presents how the safety requirements can be stated in temporal logic notation. The modelling has been performed through ASM notation, and CTL temporal logic expressions are used to express the system requirements. Unlike

other researches, random failures have been incorporated into the model through fault injection rationale, and it has been expected that model detecting element is to identify the relevant failures. Here the introduced models have been carried out for a railway station operated by Metro Istanbul Co. The interlocking table has been created for the corresponding station, and the accuracy of it has been checked it out by the introduced models via NuSMV.

Acknowledgment Funding was provided by Yildiz Teknik Üniversitesi (TR).

References

1. International Union of Railways (2007) European rail infrastructure masterplan final report
2. Verma S, Lee P, Harris I (2006) Error detection using model checking vs. simulation. In: Proceedings of IEEE international high-level design validation and test workshop. Napa Valley, USA. doi: 10.1109/HLDVT.2006.319964
3. Hoare C (1985) Communicating sequential processes. Prentice-Hall, London
4. Winter K (2002) Model checking railway interlocking systems. In: Proceedings of Australian computer science conference (ACSC 2002). doi: 10.1145/563857.563836
5. Simpson A (1998) Model checking for interlocking safety. In: Proceeding of the second FMERail seminar
6. Winter K, Neil J (2003) Modeling large interlocking systems and model checking small ones. In: Proceedings of 26th Australasian computer science conference (ACSC 2003), vol 16, pp 309–316
7. Gurevich Y (1995) Evolving Algebras: 1993 Lipari Guide. Specification and validation methods. Oxford Universtiy Press, Oxford
8. Börger E, Stark R (2003) Abstract State machines: a method for high-level system design and analysis. Springer, New York
9. Cimatti A, Giunchiglia F, Mongardi G, Romano D, Torielli F, Traverso P (1998) Formal verification of a railway interlocking system using model checking. Form Asp Comput. doi:10.1007/s001650050022
10. Anunchai S (2009) Verification of railway interlocking tables using coloured Petri Nets. In: Proceedings of the 10th workshop and tutorial on practical use of coloured Petri Nets and the cpn tools
11. Jensen K (1987) Coloured Petri Nets. Petri Nets: central models and their properties. Lecture notes in computer science, vol. 254, pp 248–299
12. Clarke E, Emerson E (2005) Design and synthesis of synchronization skeletons using branching time temporal logic. Logic of Programs, vol 131, pp 52–71
13. Quielle J, Sifakis J (2005) Specification and verification of concurrent systems in CESAR. Lecture notes in computer science, vol 137, pp 337–351
14. Clarke E (2008). Birth of model checking. 25 years of model checking, vol 5000, pp 1–26
15. Lichtenstein O, Pnueli A (1985) Checking that finite state concurrent programs satisfy their linear specification. In: Proceedings of the 12th ACM SIGACT-SIGPLAN symposium on principles of programming languages. doi: 10.1145/318593.318622
16. McMilllan K (1992) The SMV system, symbolic model checking—an approach. Technical report, CMU-CS-92-131. Carnegie Mellon University
17. Tian C, Duan Z, Zhang N (2012) An efficient approach for abstraction-refinement in model checking. Theor Comput Sci. doi:10.1016/j.tcs.2011.12.014
18. Emerson E (1995) Temporal and modal logic. Handbook of theoretical computer science. Elsevier, pp 885–1072
19. Baier C, Katoen J (2008) Principles of model checking. MIT Press, Cambridge
20. Pachl J (2009) Railway operation and control, 2nd edn. VTD Rail Publishing