Formalizing Railway Signaling System ERTMS/ETCS Using UML/Event-B

Abderrahim Ait Wakrime[1(✉)], Rahma Ben Ayed[1], Simon Collart-Dutilleul[1,2], Yves Ledru[1,3], and Akram Idani[1,3]

[1] Institut de Recherche Technologique Railenium, 59300 Famars, France
{abderrahim.ait-wakrime,rahma.ben-ayed}@railenium.eu
[2] IFSTTAR-Lille, 20 Rue Elisée Reclus,
BP 70317, 59666 Villeneuve d'Ascq Cedex, France
simon.collart-dutilleul@ifsttar.fr
[3] Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
{yves.ledru,akram.idani}@imag.fr

Abstract. Critical systems like railway signaling systems need to guarantee important properties such as safety. Formal methods have achieved considerable success in designing critical systems with verified desirable properties. In this paper, we propose a formal model of ERTMS/ETCS (European Rail Traffic Management System/European Train Control System) which is an innovative railway signaling system. This work focuses on Hybrid ERTMS/ETCS Level 3 which is currently under design, by studying and modeling the functionalities and relations of its different sub-systems. The proposed model is based on model transformation from UML (Unified Modeling Language) class diagrams to the Event-B formal language. UML is used as the primary modeling notation to describe the structure and the main characteristics of the studied system. The generated Event-B model is enriched by the formalization of safety properties. We verify and validate the correctness of the proposed formalization using the ProB model-checker and animator.

Keywords: Formal methods · Verification · Event-B · UML
Model checking · Railway Signaling System · ERTMS/ETCS

1 Introduction

In order to harmonize the variety of railway signaling in Europe, European countries launched a major industrial project, ERTMS/ETCS, to streamline international rail traffic. This is achieved by improving border crossings of the signaling systems of each country and by eliminating the need for locomotive changes due to poor interoperability at border points between two countries. ERTMS/ETCS provides benefits regarding lower investment costs and improved safety.

Three levels of ERTMS/ETCS are defined in [1], which differ in the operated equipment and the operation mode. The first two levels are already operational.

© Springer Nature Switzerland AG 2018
E. H. Abdelwahed et al. (Eds.): MEDI 2018, LNCS 11163, pp. 321–330, 2018.
https://doi.org/10.1007/978-3-030-00856-7_21

Level 3 is in design and experimentation phases. Among the main objectives of installing this level is the reduction of operating costs. The ERTMS/ETCS Level 3 implementation requires a prior study which analyses requirements in order to satisfy railway signaling system needs. In this paper, we are interested in Hybrid ERTMS/ETCS Level 3 specified in [2]. We focus on the management of train movements by fixed virtual blocks.

The goal of the research is the formalization of Hybrid ERTMS/ETCS Level 3 in the Event-B formal language. The latter is an evolution of the (classical) B method [3]. Historically, the B method was chosen to develop and validate the automatic train control system in the scope of Meteor project [4]. It has been also used for the development of several safety critical railway systems [5].

In this paper, the starting point of our approach is the modeling of Hybrid ERTMS/ETCS Level 3 system as a UML class diagram. Thereafter, the B4MSecure tool [6] is used to transform the class diagram into B formal specification. The generated model is then adapted to Event-B and enriched with safety properties. Finally, the Event-B model is verified and validated, using model checking and animation. In brief, we adopt the following two phases: (i) we first propose a formal model based on Event-B whose data structures are produced from a UML class diagram, (ii) then we validate the correctness of this formal model using a model-checking and animation technique starting from a given initial state.

This paper is organized as follows. In Sect. 2, some principles of ERTMS/ETCS signaling system and UML modeling of Hybrid ERTMS/ETCS Level 3 are presented. In Sect. 3, we detail our proposed Event-B formalization and our verification process of Hybrid ERTMS/ETCS Level 3. Finally, in Sect. 4, we conclude and provide insights for future work.

2 UML Modeling of Hybrid ERTMS/ETCS Level 3

Figure 1 gives an overview of the basic ERTMS/ETCS components: trackside equipment, on-board sub-system and GSM-R (Global System for Mobile Communications-Railway). RBC (Radio Block Center) and Eurobalise belong to trackside equipment. RBC uses train reports and interlocking status to generate MA (Movement Authority), an authorization given to a train to move to a given point as a supervised movement. Eurobalise is a spot transmission device mainly for location referencing. As on-board sub-system, EVC (European Vital Computer) is connected with trackside equipment to ensure speed regulation of the train. GSM-R is a radio system used to provide communication, i.e. exchange information (voice and data), between trackside equipment and on-board subsystem. Three levels of ERTMS/ETCS are defined. The ERTMS/ETCS Level 3 is in design and experimentation phases. Among the main objectives of installing this level is the reduction of operating costs, because it reduces the need for trackside equipment. The ERTMS/ETCS Level 2 and ERTMS/ETCS Level 3 have similar ERTMS equipment and functionalities, but in the Level 3, shown in Fig. 1, the train detection and the train integrity check are performed by the

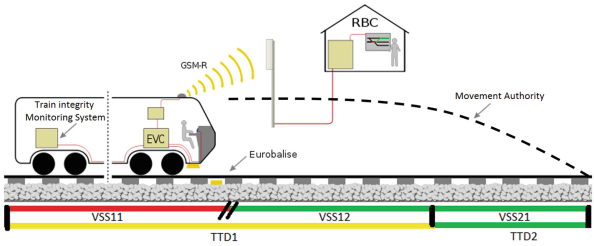

Fig. 1. Hybrid ERTMS/ETCS Level 3

on-board sub-system. The movement authority, in the Level 3, is used without visual signals or marker boards. It is sent by RBC to the train via GSM-R. This movement authority is based on train position and integrity reported by the train. In Fig. 1, the TTD (Trackside Train Detection) corresponds to conventional trackside train location equipment (e.g. track circuits and axle counters). This TTD can be divided into several VSSs (Virtual Sub-Sections). These subdivision principles represent the Hybrid ERTMS/ETCS Level 3, a variant of ERTMS/ETCS Level 3 system. Figure 2 represents an extract of class diagram that is a static view of our proposed Hybrid ERTMS/ETCS Level 3 model. This model provides an overview of the management of train movements depending on the occupancy state of the VSSs. This diagram shows the following system's classes: Train, MA, VSS and TTD.

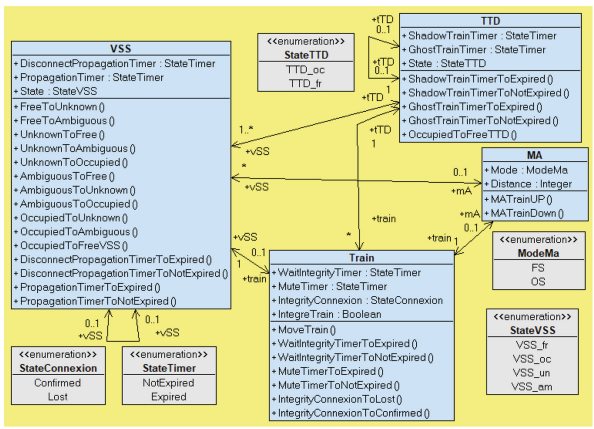

Fig. 2. Class diagram of a part of Hybrid ERTMS/ETCS Level 3.

VSSs are used by RBC and interlocking to ensure the safety of the system since it allows the spacing between trains going in the same direction on the same track. The presence of a train detected by a TTD in a given VSS, makes

the VSS state *Occupied*. If a TTD did not detect a train in a VSS, the VSS state becomes *Free*. In this paper, we do not deal with others VSS states like *Ambiguous* and *Unknown*. In the nominal situation, an MA is determined, by the RBC, according to the position of the other trains in front of it in terms of VSSs. In this paper, we deal with nominal situation when train movements are under FS (Full Supervision) operating mode. Under this mode, the real speed, maximal authorized speed, and optionally the target speed and the target distance are displayed by driver machine interface. In addition, the on-board system supervises train speed and movements.

3 Formalizing Hybrid ERTMS/ETCS Level 3

3.1 Proposed Approach

Our approach follows two steps. First, we use the B4MSecure tool [6] to translate the class diagram of Hybrid ERTMS/ETCS Level 3 into B specifications. B4MSecure produces data structures and basic operations for model instantiation (object constructor/destructor, setters for attributes and associations). Some of these basic operations are not useful in our work. However, the data structure is relevant since it reproduces correctly the structure of the class diagram. Then, we adapt the generated B model to Event-B model manually by keeping the data structure of the model, suppressing B operations and introducing events. This is accomplished by introducing user defined events describing the behavioral semantics of Hybrid ERTMS/ETCS Level 3. We also introduce the formalization of safety properties as invariants in the generated model. Finally, we proceed with the verification process using the ProB model-checker and animator. The full details of the Event-B method are not given in this paper, references [3,7] can be useful. We give a short description of Event-B method for understanding the Event-B model of our case study below. Event-B is a state-based formal method for modeling and analyzing systems. A model uses two types of entities to describe a system: Machines and Contexts. A Machine represents the dynamic part of a model *i.e.* states and transitions. A Context contains the static part of the model *i.e.* static types (constants and sets). In our case, Hybrid ERTMS/ETCS Level 3 is modeled by a single Machine, that includes both static and dynamic parts. Generally, a model is defined by a name, sets, constants and their properties, variables and their invariants and events. An event takes the form: **evt** $\triangleq$ **any** x **where** G **then** Act **end**. Where x is the list of event parameters, G represents predicates which define the guard of the event and Act is an action that modifies some state variables. When the guard is satisfied, the event can be fired.

In summary, the main steps of our approach are depicted in Fig. 3: (1) Formalization of Hybrid ERTMS/ETCS Level 3, that is motivated by the management of VSSs. (2) Validation of obtained Event-B formalization using a model-checker and an animator.

3.2 Hybrid ERTMS/ETCS Level 3 Event-B Model

In the initial step, the B4MSecure tool is used to automatically generate the B data structure of the class diagram. Thereafter, this model is enriched by some safety properties as invariants and by the definition of events. The guard and action of each event must be specified in such a way that it establishes invariant preservation. To illustrate our approach, we generate the B Machine from the UML class diagram shown in Fig. 2. Listing 1 represents an extract of the generated machine called *ERTMSETCS3*.

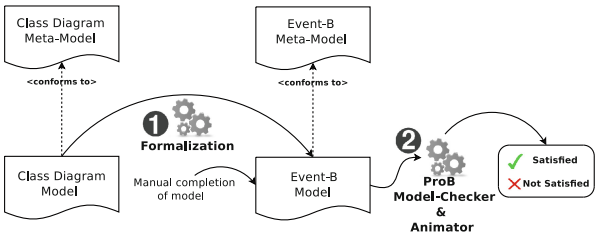

Fig. 3. Overview of the transformation approach.

In Listing 1, the *TRAIN*, *VSS_AS*, *TTD_AS*, *MA_AS* are specified as sets in the Event-B Machine corresponding respectively to *Train*, *VSS*, *TTD* and *MA* classes. In this paper, we introduced only the useful invariants for the presented events. The model is more complicated than that, because it covers other mechanisms like propagation timers, waiting timers, mute timers and state of integrity connection. These latter are used to constrain changes in the state of a VSS, for example going from *Ambiguous* to *Occupied* states [2]. This work does not show these mechanisms in the Event-B formalization. We focus only on the VSS state changes from *Free* to *Occupied* and from *Occupied* to *Free*.

Added Variables. We manually enrich the formalization of the generated Machine with some variables to expand the state space of the Machine with additional information. These variables are represented in the *abstract variables added manually* part of Listing 1.

Added Invariants. We define the typing invariant properties of each added variable to complete the model construction. Listing 1. includes the invariants generated by B4MSecure tool and the added invariants (see comments in the invariant clause). These added invariants ease the formalization of the safety properties and the specification of events.

MACHINE $ERTMSETCS3$
SETS $VSS\_AS; TTD\_AS; MA\_AS; TRAIN; ModeMA = \{OS, FS\};$
$StateVSS = \{VSS\_fr, VSS\_oc, VSS\_un, VSS\_am\};$
$StateTTD = \{TTD\_fr, TTD\_oc\}$
ABSTRACT_VARIABLES
\\The generated abstract variables by B4MSecure tool
$VSS, TTD, MA, MA\_Mode, Train, VSS\_State, TTD\_State, Train\_MA,$
$MA\_VSS, TTD\_VSS, IntegerTrain, MA\_Distance$
VSS_next, TTD_next

```
\\The abstract variables added manually
  train_VSS_current, train_TTD_current, last_VSS_MA, VSS_with_TTD
INVARIANT
\\The generated invariants by B4MSecure tool
 VSS ⊆ VSS_AS  ∧  TTD ⊆ TTD_AS  ∧
 MA ⊆ MA_AS  ∧  MA_Mode ∈ MA → ModeMA  ∧
 Train ⊆ TRAIN  ∧
 VSS__State : VSS → StateVSS  ∧
 TTD__State : TTD → StateTTD ∧ MA_Distance : MA → Integer
 MA_VSS ∈ MA ↔ ℙ(VSS)  ∧   \\ Set of VSS of each MA
 TTD_VSS ∈ TTD ↔ ℙ(VSS)  ∧   \\ Set of VSS of each TTD
 Train_MA ∈ Train → MA  ∧   \\ MA of each train
 TTD_next ∈ TTD ⤖ TTD  ∧   \\ Next TTD of each TTD
 VSS_next ∈ VSS ⤖ VSS  ∧   \\ Next VSS of each VSS
 IntegerTrain ∈ Train → BOOL \\ Train integrity
\\The invariants added manually
 train_VSS_current ∈ Train → VSS  ∧   \\ Occupied VSS by train
 train_TTD_current ∈ Train → TTD  ∧   \\ Occupied TTD by train
 VSS_with_TTD ∈ VSS ⇸ TTD  ∧   \\ TTD of each VSS
 last_VSS_MA ∈ MA ⤖ VSS  ∧   \\ Last VSS of each MA
 ∀(vss, vssSet, tr, ma).(vss ∈ VSS ∧ vssSet ∈ ℙ(VSS) ∧ tr ∈ Train ∧ ma ∈ MA  ∧
     (tr ↦ vss ∈ train_VSS_current) ∧ (tr ↦ ma ∈ Train_MA)  ∧
         (ma ↦ vssSet ∈ MA_VSS) ⇒ ¬(vss : vssSet))
           \\ Current VSS does not belong to MA ...
 END
```

Listing 1. The description of SETS, ABSTRACT_VARIABLES and INVARIANT clauses.

Safety Invariants. In nominal situations, a movement authority of a train moving under FS mode is composed of free VSSs so that the train can run at the maximum authorized speed. However, in some exceptional situations, such as the coupling of two trains (e.g. in OS mode), there are degraded modes which allow the train to be moved to an occupied VSS. We specify the safety properties avoiding trains accidents under FS mode as invariants, in Listing 2. The first invariant allows to verify that, in FS mode, all VSSs affected to an MA are free. The second one checks that each VSS will never contain two trains, hence accidents can not happen. However, the last one ensures that two MA of two different trains do not share any VSS.

```
MACHINE ERTMSETCS3
INVARIANT
\\Invariant 1
 ∀ma.(ma ∈ MA ∧ MA_Mode(ma) = FS ⇒
      ∀vss.(vss ∈ VSS ∧ vss ∈ MA_VSS(ma) ⇒ VSS__State(vss) = VSS_fr))  ∧
\\Invariant 2
 ∀(t1, t2).(t1 ∈ Train ∧ t2 ∈ Train ∧ t1 ≠ t2  ∧
   MA_Mode(Train_MA(t1)) = FS ∧ MA_Mode(Train_MA(t2)) = FS ⇒
    train_VSS_current(t1) ≠ train_VSS_current(t2))  ∧
\\Invariant 3
 ∀(t1, t2).(t1 ∈ Train ∧ t2 ∈ Train ∧ t1 ≠ t2  ∧
   MA_Mode(Train_MA(t1)) = FS ∧ MA_Mode(Train_MA(t2)) = FS ⇒
    MA_VSS(Train_MA(t1)) ∩ MA_VSS(Train_MA(t2)) = ∅)
END
```

Listing 2. Safety properties invariants under FS mode.

Events Specification. In this stage, we specify events in order to model the system behavior. In Listing 3, the event *MoveTrain* allows to move the train from a VSS to another one. However, it does not release the VSS just left by

the train. This release is insured by the *OccupiedToFreeVSS* event in Listing 4. Here, we do not take into account the intermediate state in which the train is located on two VSSs. The train moves to the next VSS if it is free and if it belongs to the allocated MA of this train. We note that trains circulate in the same direction. Once the train moves from the current VSS to the next one, the next VSS becomes occupied. The MA is updated by removing the new current VSS. In fact, the current VSS does not belong to the MA (the last invariant in Listing 1). If the train leaves a TTD to go to the next one, the state of this new current TTD changes to occupied. To update MA in parallel with movement of train, we define *MATrainUP* event which is not described here for lack of space. It allows to extend the MA of a train with a free VSS. This event updates an authority for a train to proceed up to an end point of a VSS where it has to stop. It exhibits the safe path of a train in terms of free VSSs.

```
MACHINE ERTMSETCS3
EVENTS
        MoveTrain = ANY train, vss_crnt, vss_nxt, ma
WHERE   train ∈ Train ∧ vss_crnt ∈ VSS ∧ vss_nxt ∈ VSS ∧ ma ∈ MA ∧
        vss_crnt ↦ vss_nxt ∈ VSS_next ∧
        train_VSS_current(train) = vss_crnt ∧
        train ↦ ma ∈ Train_MA ∧ vss_nxt ∈ MA_VSS(ma) ∧
        VSS_State(vss_nxt) = VSS_fr
THEN    train_VSS_current := train_VSS_current ⩤ {train ↦ vss_nxt} ||
        train_TTD_current := train_TTD_current ⩤
                    {train ↦ VSS_with_TTD(vss_nxt)} ||
        TTD_State(VSS_with_TTD(vss_nxt)) := TTD_oc ||
        MA_VSS := MA_VSS ⩤ {ma ↦ MA_VSS(ma)
           − {VSS_next(train_VSS_current(train))}} ||
        VSS_State(vss_nxt) := VSS_oc
   END; ...
END
```

Listing 3. *Moving train* event.

The event *OccupiedToFreeVSS*, as shown in Listing 4, represents the transition from occupied state to free state of a VSS. When a train with checked integrity (specified by *IntegerTrain* function whose state changes in a specific event) has reported to have left the VSS, the VSS that the train leaves will become free (using a total function *VSS_State(vss_crnt) := VSS_fr*). It is the same case for each TTD: when all its VSSs are free, the TTD becomes free.

```
MACHINE ERTMSETCS3
EVENTS
        OccupiedToFreeVSS = ANY vss_crnt, vss_nxt, ttd_crnt, ttd_nxt, train
WHERE   ttd_crnt ∈ TTD ∧ ttd_nxt ∈ TTD ∧
        vss_crnt ∈ VSS ∧ vss_nxt ∈ VSS ∧
        train ∈ Train ∧
        VSS_State(vss_crnt) = VSS_oc ∧
        vss_crnt ↦ vss_nxt ∈ VSS_next ∧
        ttd_crnt ↦ ttd_nxt ∈ TTD_next ∧
        IntegerTrain(train) = TRUE ∧
        train_VSS_current(train) = vss_nxt ∧
        vss_crnt ∉ ran(train_VSS_current)
THEN    VSS_State(vss_crnt) := VSS_fr
   END; ...
END
```

Listing 4. *OccupiedToFreeVSS* event.

3.3 Verification and Validation

We aim to validate Event-B models and to verify that the invariants (typing and safety properties invariants) are preserved by all events. These models have finite state spaces, *i.e.* the objects set remains constant, because, the events do not add Train, TTD, VSS or MA instances. Figure 4 represents our motivating example derived from [2]. It is a running of two single trains (train1 and train2) with integrity confirmed by external device. This example includes three TTDs: TTD10, TTD20 and TTD30. Each TTD contains a set of VSSs like: TTD10 is composed of VSS11 and VSS12, TTD20 consists of VSS21, VSS22 and VSS23, TTD30 is composed of VSS31, VSS32 and VSS33. We give values to the constants, carrier sets and the variables of the model through INITIALISATION clause. This clause specifies the Hybrid ERTMS/ETCS Level 3 model elements values in the initial state. This initialisation and events must preserve typing and safety invariants showing the absence of rear-end accidents. In this model, the face to face accidents are not treated since we do not take into consideration trains moving in opposite directions. We use the ProB model-checker [8]. In this work, theorem proving is not used, because, it requires more training and effort than model checking. On the other side, since the system has a finite state space, the model checking is sufficient to check safety properties for a given initial state. Step 1 of Fig. 4. represents an initial state of our case study.

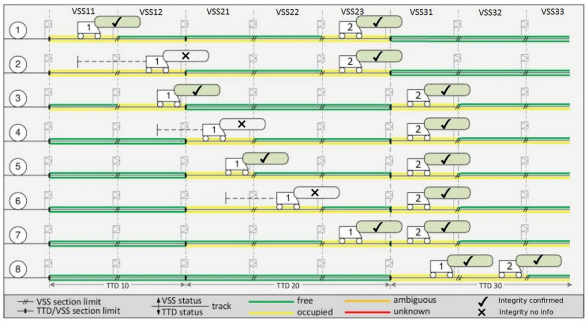

Fig. 4. An example of running of two single trains with integrity confirmed.

Verification Using Model Checking. Model checking is an automatic formal verification technique that is widely applied for the verification of a desired behavioral property of a given system [9]. It allows to verify that a system satisfies a given property using efficient algorithms. These algorithms are based on exhaustive enumeration (explicit or implicit) of all the reachable states from initial state. All experiments were conducted on a 64-bit PC, Ubuntu 16.04 operating system, an Intel Core i5, 2.3 GHz Processor with 4 cores and 8 GB RAM. Using the ProB model-checker and based on mixed breadth and depth search strategy, we have explored all states: 100% of checked states with 1347 distinct

states and 3431 transitions during 2566 ms. No invariant violation was found, and all the operations were covered. This verification ensures that invariants are preserved by each event. Otherwise, a counter-example would have been generated.

Validation by Animation. ProB can be used as an animator. It allows automatic animations of Event-B model and to play several scenarios. Indeed, ProB animator displays the values of each variable, the enabled events and the history of chosen events. We have successfully applied the animation of ProB on the operational scenario of Fig. 4. ProB checks our model step by step from initial step to the final one and it shows the behaviour of model in clear terms.

4 Conclusion and Future Works

In this paper, we have presented a formalization and verification of Hybrid ERTMS/ETCS Level 3, using the Event-B modeling language. This approach is based on model transformation of a UML class diagram to Event-B. To do so, we used the B4MSecure tool to transform the structure of the class diagram to Event-B. Then, we enriched the generated model with the specifications of Hybrid ERTMS/ETCS Level 3 and with safety properties to prevent train collisions. We used the ProB model-checker to verify invariant preservation for a given initial state. In addition, we used ProB animator to validate that it supports the simulation of operational scenarios.

In our future work, we are interested in enriching our model by adding agents (*e.g.* train driver, traffic agent) to the model of Hybrid ERTMS/ETCS Level 3. This model will complement the current model, presented in this paper, to express which human and software agents have permission to access the objects and operations of the current model. This adopts a separation of concerns approach similar to the one used in information systems.

Acknowledgments. This work is funded by the NExTRegio project of IRT Railenium. The authors would like to thank SNCF Réseau for its support.

References

1. Schön, W., Larraufie, G., Moëns, G., Poré, J.: Railway Signalling and Automation. Work in Three Volumes. La Vie du Rail, Paris (2013)
2. European Economic Interest Group: Hybrid ERTMS/ETCS Level 3: Principles, Brussels, Belgium, July 2017
3. Abrial, J.R., Abrial, J.R.: The B-book: Assigning Programs to Meanings. Cambridge University Press, New York (2005)
4. Behm, P., Benoit, P., Faivre, A., Meynadier, J.-M.: Météor: a successful application of B in a large project. In: Wing, J.M., Woodcock, J., Davies, J. (eds.) FM 1999. LNCS, vol. 1708, pp. 369–387. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48119-2_22
5. Lecomte, T., Servat, T., Pouzancre, G., et al.: Formal methods in safety-critical railway systems. In: 10th Brasilian Symposium on Formal Methods, pp. 29–31 (2007)

6. Idani, A., Ledru, Y.: B for modeling secure information systems. In: Butler, M., Conchon, S., Zaïdi, F. (eds.) ICFEM 2015. LNCS, vol. 9407, pp. 312–318. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25423-4_20
7. Abrial, J.R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press, New York (2010)
8. Leuschel, M., Butler, M.: ProB: an automated analysis toolset for the B method. Int. J. Softw. Tools Technol. Transf. **10**(2), 185–203 (2008)
9. Clarke, E.M., Grumberg, O., Peled, D.: Model Checking. MIT press, Cambridge (1999)