

FORMAL MODELING AND DATA VALIDATION OF GENERAL RAILWAY INTERLOCKING SYSTEM

WANG KEMING^{1,2}, WANG ZHENG¹ & ZHANG CHUANDONG³

¹Department of Railway Information Engineering, Southwest Jiaotong University, China

²National-Local Joint Engineering Laboratory of System Credibility Automatic Verification,
Southwest Jiaotong University, China

³Beijing HollySys Co. Ltd., China

ABSTRACT

Railway interlocking system is a typical safety-critical system, design defects of the system will pose the great risks on the safety and affect the operation efficiency of the railway station. Formal method is an important approach to verify the design requirement and to get the reliable logic for coding. By analysing the requirement of railway interlocking system, the properties of specification and the events of system's function were obtained, and then a multilayer formal model using the Event-B language and refinement strategy was established. The safety attributes of the system were verified and the formal model was refined based the theorem proving. Taking a real railway station as example, the contradictions of the axioms and the deadlock of the model were checked, as well as the correctness of the interlocking data was validated. Finally, the correctness of the model function was tested by simulation. We developed a formal prototype model for the general interlocking system and proposed an approach of data validation for the real station with the interlocking table.

Keywords: railway signalling system, station interlocking, formal modelling, data validation, theorem proving, model checking, simulation.

1 INTRODUCTION

Interlocking is important technology to guarantee efficiency and safety of the railway station's operation. Interlocking system is a typical safety critical system, any potential design flaws of system are likely to lead to huge risks for the station operation. Due to the increasing demand of system design and capacity of stations, it is a great challenge to the development of interlocking system. The statistical data show that the most risks of system are produced in the requirements analysis stage [1], as a result, it is very important to check design flaw to ensure the reliability and completeness of the system in the early design stage.

Formal method is an effective way to enhance the reliability of system, which can greatly reduce the testing errors, significantly improve project quality and development efficiency [2]. This method has been used in the development of the interlocking system.

As the interlocking system is a typical data-driven system, validating the correctness and completeness of data is critical to the safe operation of the system. The research on interlocking data validation is still in the process of exploration, and it is necessary to study more efficient methods of general interlock data validation.

In this paper, we established a formal model with general function of interlocking system, proposed an approach for interlocking data validation based the general model. In the follow sections, the attributions of system and events were obtained by analyzing the design specifications of the interlocking system. The properties and flows of events were described by Event-B language on Rodin platform, UML diagrams were used to assist in establishing the model. Through proving the proof obligations of each layer, and verifying the properties of the system, the defects in the specification and the process of analysis were observed, and then the formal model was improved. After successful building the initial model, the subsequent layer models were established by refinement policy. Taking a real station yard as



example, the correctness of axioms was proved, the deadlock and the invariant violation were checked, as well as the interlocking data were validated. Finally, the response of the model in the environment simulated by the real case was tested, which further confirmed the correctness of the general model and the interlocking data.

2 ANALYSIS OF PROPERTY AND EVENT FLOW

The verification of the interlocking system needs to confirm whether the system can work properly under the interlocking management specification. Fig. 1 describes the technical route in this paper, it showing that: in order to express the specification requirements in the model for verification, firstly we must analysis deeply the interlocking specification, so that obtain the requirements of the system on environmental properties, functional properties, and safety properties, and extract the logic flows of the functional events, which will build a foundation for modeling and verification.

2.1 Property analysis

In this paper, the property of requirements in the interlocking system has environment properties, function properties and safety properties. The property of environment (ENV) is used to definition the objects of modeling in the system, including the system object and its subclasses. ENV also contains specific description for the typical feature of the object. Based on the environment objects, the description of the whole and part of the system can be established, and a variety of scenarios involved in system operation can be built.

There are 3 kinds of environmental objects in the railway station system: signal, track section and switch. Based on the interlocking management specification, taking the track section as an example, the environment properties of track section we extracted are as follows:

ENV_TS1: There are 3 kinds of track sections: section with a switch, section without a switch.

ENV_TS2: The section with a switch contains at least one switch.

ENV_TS3: More than one switch can exist in the section with a switch.

ENV_TS4: The section without a switch has no switch.

ENV_TS5: The section without a switch contains general sub-sections without a switch and the leading/towing lines.

Function Properties (FUN) are used to describe the basic functions of the system objects and the execution and switching conditions of different functions. Safety properties (SAF) are used to describe the conditions or the standards of safety during operation. Essentially the safety properties of the system are the safety-critical functional requirements for the system object.

Take the signal showing and signal holding as an example to describe the establishment of functional properties and safety properties.

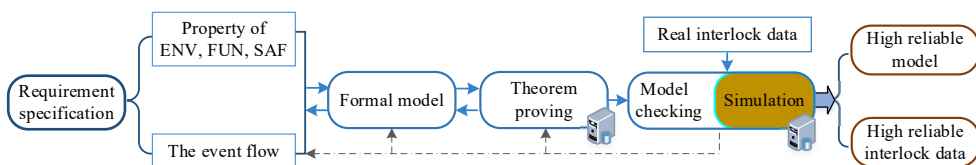


Figure 1: The technical route.

FUN_SIG1: The signal can normally display the right colour when the route is locked.

FUN_SIG2: The interlocking computer should keep logically judging the conditions for lighting during the signal-on and signal-hold period.

SAF_SIG1: The signal is allowed to be open only when the filament of the signal is in good condition, all the track sections are free, all switches are locked in the specified location, and all hostile signals are not open.

SAF_SIG2: During signal holding period, if the section is faulty occupied, the signal should be off immediately, but do not unlock the route.

SAF_SIG3: During signal holding period, if the switch fails to indicate, the signal machine should be off immediately, but don't unlock the route.

SAF_SIG4: During signal holding period, if the signal machine is failure, the signal should be red; if the filament of red light is also broken, the signal should not light for any other colours, and should be in the off state.

SAF_SIG5: When interlocking device has a systematic failure or any other fault is detected, the permissible signal should be closed in time, and the signal should be red. After the signal is closed for any reasons, the signal shall not light if without handling the fault.

2.2 The flow of event

The interlocking process includes route selection, route lock, signal showing, signal holding and route unlocking. The events can describe in detail all the functions of the system in the scenarios which is defined by the environment properties under the constraints of safety property. So, the construction process of the event is an organic combination of various properties. In the process, it is necessary to obtain the basic events of the various functions in the system and their execution conditions, decompose the complex functions to get the initialization states and the basic events as well as their execution sequences, and build the scenarios of functions and the safety constraints.

3 FORMAL MODELING BASED ON EVENT-B METHOD

Event-B is a modelling language for describing the discrete systems, which uses first order logic and set theory as a modelling notation [3]. An Event-B model consists of two parts: *Context file* and *Machine file*. *Context file* defines the static properties of the system. *Context file* consists of *Carrier Set* (define the abstract sets of the actual objects that make up the system), *Constant* (define the fixed objects in the system) and *Axiom* (General principles in systems or a priori general guidelines).

Machine file defines the dynamic properties of the system's operation, it consists of *Variable*, *Invariant* (the basic principles obey by the system's dynamic operation), *Variant* and *Event*. Events define the state transition.

Axioms, *invariants* and execution conditions of events can all be identified as theorem, we should prove they are true using previously defined axioms, invariants or proved theorems.

3.1 The initial model

The initial model is the starting point for modeling and verification of the system, which will affect the simplicity of the whole model structure and the convenience of the subsequent refinement. The core function of the interlocking system is to control the process of lock and release route, therefore, the initial model only contains locking and releasing of the route. The data type to be defined in the context file should include an abstract set of *ROUTE*; two



events should be included in the machine file: *lock_route* and *free_route*, and a variable *lockedRoute* need to define to represent the locked route. We use a plug-in iUML-B to establish the UML diagrams of event flow, then generate the context file and machine file of the models, so improve the automation of formal processes [4]. The UML diagram of the initial model is shown in Fig. 2.

3.2 Refinement policy

In the process of building the model, a refinement policy is used to decompose the complex system functions. When refining initial model after it is proved, the new context files and machine files will inherit the proven theorems and invariant. The system functions will gradually add and improve in the process of refinement and verification. The policy of refinement ensures the correctness of lower level models, so can reduce the difficulty of system modelling and improves the flexibility of the modification process. The process of refinement is shown as Table 1. In the refinement processes, we can continue to generate Event-B language model by improving the UML diagram.

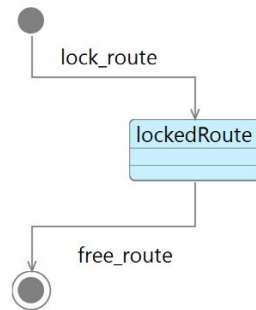


Figure 2: The UML diagram in the initial model.

Table 1: The refinement processes of the model.

Steps	Objects and Events
First refinement	Bring in the object as train, build the framework of lock route, normal unlock route after the train passed, cancel route, artificial delay free route and fault free route.
Second refinement	Bring in the concepts of route type, route direction, track section type, train type and hostile route, further improve the attributes of the route, track section and train.
Third refinement	Bring in the object as switch, synchronize the locking and release of the switch to the control flow of lock route and free the route.
Fourth refinement	Improve the related events of the switch, introduce the position information of the switch, add the events of switch turning and timing.
Fifth refinement	Bring in the objects as signal, improve the event of cancel route and artificial delay free route.
Sixth refinement	Add the concepts of route process (establish the various stages of the process) and add signal equipment failure and repair events, improve the function of fault release.

4 VERIFICATION OF THE SYSTEM PROPERTIES

4.1 Event-B description of system properties

In a context file, we define the properties of the *set* or the *constant* as *Axiom*. Taking the context file of first layer model as an example, the related properties are defined by Event-B as follows:

1. For each route, the track section in the *routeTrack* must also belong to the track section in the *routeTrackNum*. Its description by Event-B is:

$$\forall r.(r \in ROUTE \Rightarrow \text{dom}(\text{routeTrack} \triangleright \{r\}) \subseteq \text{dom}(\text{routeTrackNum}(r))) .$$
2. *routeTrack* is not equivalent to *routeTrackNum*. When the route is a shunting route, and the last section of the track belongs to the section without a switch or the receiving-departure track, it is no need to consider the section in the interlocking table, so *routeTrack* and *routeTrackNum* are same or only different from the last section of the track, they are subsets relation. So:

$$\forall r.(r \in ROUTE \Rightarrow \text{card}(\text{dom}(\text{routeTrack} \triangleright \{r\})) = \text{card}(\text{dom}(\text{routeTrackNum}(r))) \vee \text{card}(\text{routeTrack}) = \text{card}(\text{dom}(\text{routeTrackNum}(r)) - 1) .$$
3. For each route, the number of the track section in the *routeTrackNum* should be a continuous number from 1 to *routeTrackCount*. So:

$$\forall r.(r \in ROUTE \Rightarrow \text{ran}(\text{routeTrackNum}(r)) = 1..\text{routeTrackCount}(r)) .$$
4. The number of track sections in each route should be greater than or equal to 1. So:

$$\forall r.(r \in ROUTE \Rightarrow \text{card}(\text{dom}(\text{routeTrack} \triangleright \{r\}) \geq 1)) .$$

Functional properties have also been achieved in each machine file, *invariants* in machine file are used to describe the safety properties that the system must comply throughout its operation. To keeping the establishment of safety properties, it is necessary to ensure that the related invariants are true during the execution of first layer model, otherwise, the safety properties cannot be completely satisfied. Taking the machine file in first layer model as an example, the safety properties are described by Event-B as follows:

1. Any two conflicting routes cannot be established at the same time.

$$\forall r1, r2.(r1 \in \text{lockedRoute} \wedge r2 \in \text{lockedRoute} \wedge r1 \neq r2 \Rightarrow r1 \mapsto r2 \notin \text{conflictRoute}) .$$
conflictRoute is the set of conflicting routes.
2. A track section can be only locked by one route.
$$\forall r1, r2.((r1 \in \text{lockedRoute} \wedge r2 \in \text{lockedRoute} \wedge r1 \neq r2) \Rightarrow \text{dom}(\text{lockedTrack} \triangleright \{r1\}) \cap \text{dom}(\text{lockedTrack} \triangleright \{r2\}) = \emptyset) .$$
3. The section of the route established should have been locked.

$$\forall r.(r \in \text{lockedRoute} \Rightarrow \text{routeTrack} \triangleright \{r\} \subseteq \text{lockedTrack}) .$$
4. The section of the route established is not occupied by a train.

$$\forall r.(r \in \text{lockedRoute} \Rightarrow \text{ran}(\text{train_Pos}) \cap \text{dom}(\text{routeTrack} \triangleright \{r\}) = \emptyset) .$$
5. The section that has been occupied by the train cannot be used to build another route.

$$\forall r.(\text{ran}(\text{train_Pos}) \cap \text{dom}(\text{routeTrack} \triangleright \{r\}) \neq \emptyset \Rightarrow r \notin \text{lockedRoute}) .$$




4.2 The proof obligation

Due to the huge leap from requirement to formal model, it is difficult to avoid missing some key logical information when describing the system, also maybe there are defects in related standard. These defects will be found when the generated proof obligations are proving. The interactive process of proving proof obligation is very important for the developers to deeply understand the standard, so improve system analysis for properties and formalizing model.

After improvement of analysis of system properties, modification of the model includes adding of event condition (*guard*), adding *axioms* or *invariants* [5], which will improve the constraints of properties in the model. An example as adding the axiom to revise the model is follow:

When proving the proof obligation of *rotate_switch_complete_2/inv1/INV* in the fourth layer model, after the predicate formula $routeSwitchPos(r \mapsto sw) = dw \vee routeSwitchPos(r \mapsto sw) = fw$ added (*RouteSwitchPos* is the correct position that the switch should be locked when the route is building, *dw* is normal position, *fw* is reverse position), the proof obligation generates the proving target which this predicate formula can be obtained by other existing predicate calculus, but the proof of the target cannot be proving. After analysising the reason is:

When defining *routeSwitchPos* in Context4, its data type defined as $(ROUTE \leftrightarrow SWITCH) \mapsto SWITCH_POS$, *SWITCH* is the set of the switch, the state of the switch $SWITCH_POS = \{dw, fw, null\}$, *null* stands for the switch is in four open states. The value of *routeSwitchPos* ($r \mapsto sw$) may be *null*, so the above target cannot be proved. Add an axiom $ran(routeSwitchPos) \subseteq \{dw, fw\}$, $ran(r_otherSwitchPos) \subseteq \{dw, fw\}$ in Context4, (*r_otherSwitchPos* is used to record the state of the other end of a double moving switch associated with the *routeSwitchPos*) and then return to the proof interface again, choose the above axiom as the assumption of this proof obligation, once again do proving and this target has been proving, that is  *rotate_switch_complete_2/inv1/INV*.

The model generated 710 proof obligations, some proof obligations were proving with manual interactive adjustment and model modification. The statistics are shown in Fig. 3.

Element Name	Total	Auto	Manual	Reviewed	Undischarged
Interlock_virtual5	710	199	511	0	0
C0	1	1	0	0	0
C1	6	5	1	0	0
C2	4	2	2	0	0
C3	2	2	0	0	0
C4	6	6	0	0	0
C5	5	3	2	0	0
C6	0	0	0	0	0
M0	0	0	0	0	0
M1	80	47	33	0	0
M2	59	16	43	0	0
M3	46	13	33	0	0
M4	118	32	86	0	0
M5	146	18	128	0	0
M6	237	54	183	0	0

Figure 3: The statistics of proof obligation in the refinement processes.

5 INTERLOCKING DATA VALIDATION

5.1 Axiom verification and data validation

The above kinds of attributes of proof obligation have been passed to ensure that the accessibility and safety of the functions of interlocking system. But to ensure the security and reliability of the system, we also need to: (1) Confirm the correctness of axioms. In the above proof of obligation, it assumed that all axioms have been correctly defined and their attributes are true. All the axioms in the model are formally built by the user after analyzing the requirements, as the potential errors of description will cause contradictions among the axioms; (2) Validate the data security of the interlocking in the real railway station. Interlocking data import errors in the developing process. The data related to the correct realization of computer interlocking function, therefore, the validation of data security is the key of the development and application of computer interlocking.

In order to eliminate the errors in the establishment of axioms and the mistakes in the compilation of interlocking data as much as possible, we design the following technical solution for data validation, as shown in Fig. 4. The approach is to create a new model that only has the context files, and all context files are instantiated from the former model's context files: Using the real railway station data to instantiate the constants and setting the axioms of attributes as the theorem. All theorems will automatically generate the corresponding proof obligation, the purpose is to prove that all the theorems can be derived from the previously defined axioms and theorems, in order to ensure the correctness of the axioms defined by each property.

The example used in this paper shown in Fig. 5 is a typical three-tracks railway station, its interlocking table (including 12 operations of receiving routes and departure routes) is as shown in Table 2. ProB is used to verify the model, ProB is an animator, constraint solver and model checker for the B-Method [6].

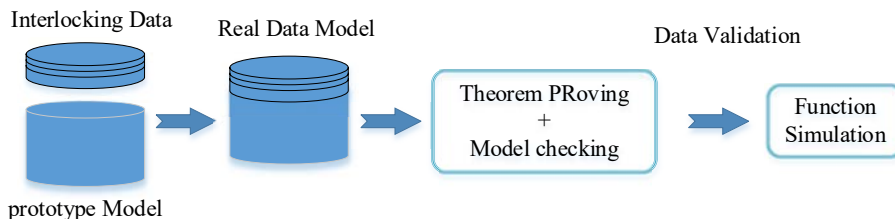


Figure 4: The technical solution for data validation.

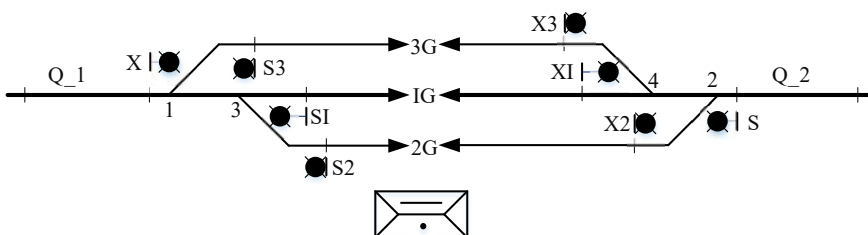


Figure 5: An example layout of a railway station.

Table 2: The interlocking table of the example station.

Route Number	From	To	Signal		Lock points	Tracks	Route locked
			Name	Aspect			
1	X	1G	X	Y	1, 3	1/3DG, 1G	7
2	X	2G	X	Y	1, (3)	1/3DG, 2G	8
3	X	3G	X	Y	(1), 3	1/3DG, 3G	9
4	S	1G	S	Y	2, 4	2/4DG, 1G	10
5	S	2G	S	Y	(2), 4	2/4DG, 2G	11
6	S	3G	S	Y	2, (4)	2/4DG, 3G	12
7	1G	X	S ₁	G	1, 3	1/3DG, Q ₁	1
8	2G	X	S ₂	G	1, (3)	1/3DG, Q ₁	2
9	3G	X	S ₃	G	(1), 3	1/3DG, Q ₁	3
10	1G	S	X ₁	G	2, 4	2/4DG, Q ₂	4
11	2G	S	X ₂	G	(2), 4	2/4DG, Q ₂	5
12	3G	S	X ₃	G	2, (4)	2/4DG, Q ₂	6

Y: Yellow. G: Green. (3): 3# Switch in Reverse. 1/3DG: 1G to 3G Section.

In the process of proving, the proof obligation generated by Theorem $routeTrackNum \in ROUTE \rightarrow (TRACK \rightsquigarrow \mathbb{Z})$ in C1 file has not been proved, where $ROUTE = 1 \dots 12$, $TRACK = \{1G, 1IG, 1IIG, DG_1_3, DG_2_4\}$, $routeTrackNum$ has been assigned as the connection number of each track section of route. The analysis shows that the result caused by the too large state space, so we design a way to decompose the state space:

1. Define 4 axioms as $ROUTE1 = \{1,2,3\}$, ..., $ROUTE4 = \{10,11,12\}$ to replaced $ROUTE = 1 \dots 12$, and define an axiom $ROUTE = ROUTE1 \cup ROUTE2 \cup ROUTE3 \cup ROUTE4$;
2. Assign $routeTrackNum1, \dots, routeTrackNum4$ as the track sections in $ROUTE1, \dots, ROUTE4$ corresponding to instead of the real data of $routeTrackNum$, and define the axioms: $routeTrackNum = routeTrackNum1 \cup routeTrackNum2 \cup routeTrackNum3 \cup routeTrackNum4$;
3. Define 4 new theorems: $routeTrackNum1 \in ROUTE1 \rightarrow (TRACK \rightsquigarrow \mathbb{Z}); \dots; routeTrackNum4 \in ROUTE4 \rightarrow (TRACK \rightsquigarrow \mathbb{Z})$, while adding a theorem to check the independence of the domains of $routeTrackNum1, \dots, routeTrackNum4$;
4. Put the original theorem after the above axioms and 5 theorems.

That proving the original theorem is replaced by proving 5 smaller-scale theorems. As shown in Fig. 6, all the proof obligation in six theorems have been approved. The above

Element Name	Total	Auto	Manual	Reviewed	Undischarged
interlock_virtual5_2	93	33	59	0	1
C0	7	3	4	0	0
C1	22	6	15	0	1
C1_routeTrackNum	6	3	3	0	0
C2	15	2	13	0	0
C3	9	3	6	0	0
C4	20	16	4	0	0
C5	14	0	14	0	0
C6	0	0	0	0	0

Figure 6: The results of axiomatic contradiction checking

modification is listed in the file C1_routeTrackNum which is only used for the demonstration of the above process, in the actual process the original C1 file is needed to revise.

Fig. 6 shows that all the theorems in the model can be verified by the real interlocking data, it shows that the real data can satisfy all the theorems in Model 2, it means that the actual station data can satisfy all the axioms in the former model, which proves the correctness of axiomatic definition of the Model 1.

In Model 2, it has been proved that the interlock data of the example railway station is a solution under all axioms of the prototyping model. Create a new model 3, which context files are the context files in model 2, and its machine files are the machine file in the prototyping model. In order to ensure that the system will not be locked in one state, we need to check whether there is deadlock in the model, so as to ensure that the running process of the model can be smooth executed until the end of any scenario. We do the deadlock checking and the correctness checking of model invariants in same time, the checking result is shown in Fig. 7.

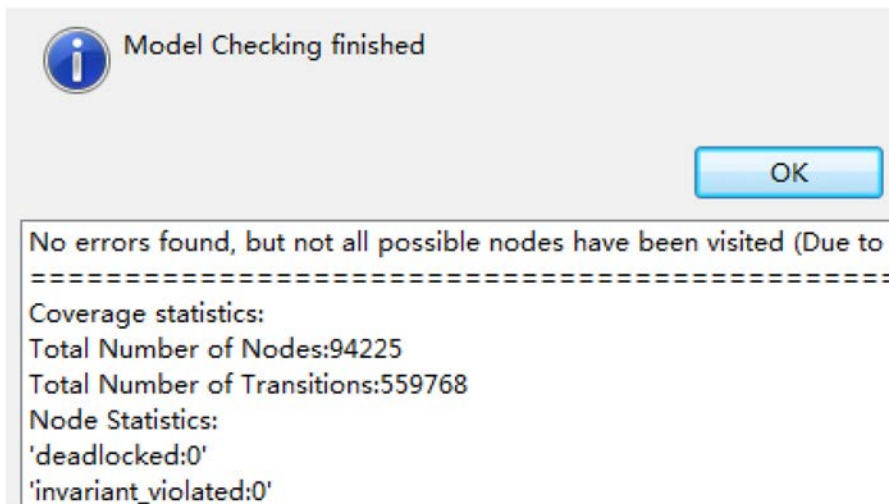


Figure 7: The model checking result.

5.2 Simulation experiment

Based on model 3, the execution scenarios of the real railway station is built using the plug-in tool BMotionWeb [7], so we can simulate the response of the model under different variables and parameters to test the data of the interlock table. The state of signal can obtain by the variable *color_colour*, the state of switch and the operating status of the train in the railway station can be controlled by the variable *switch_Pos* and *train_Pos*.

According to the above interlocking table, we can build routes and unlock routes individually in the visual simulation environment, the function of the system can be normally realized. In Fig. 8, seventh route in the interlocking table is the normal building for a departure route from track I to track X, the departure signal SI is shown as a green, and the switch of light colour is consistent with the interlocking table.

The above process of theorem proving and model checking not only check the correctness of the model, but also verify the correctness of the interlocking data, through the way of simulation testing, the correctness of the model function and the reliability of the process of the interlocking table making, as well as the credibility of the validation result are further guaranteed.

6 RELATED WORK

The formal method is already applied in the research of interlocking system for many years. For example, Literature [8] proposed the MPN (Mobile Petri Net) approach based on the Mobile intelligent agent and Petri Net, which was used to model the interlocking system and verify it by reachability graph. Literature [9] established the time automata model of interlocking system using UPPAAL software and verified the safety related properties. In the literature [10] the model of interlocking system which accommodated the sequence release was built, using a combination of SMT (Satisfiability Modulo theory) based Bounded Model Check and inductive reasoning, the safety properties were verify.

The above researches use the model checking approach. Theorem proving has achieved great development in recent years, which makes up for the defects of the model checking method can only deal with finite state space. the Event-B method with the support platform Rodin combines the two methods, which has been used in the research of interlocking system [11]. The literature [12] discussed how to effectively establish the formalized model of railway control products based on an example of interlocking system.

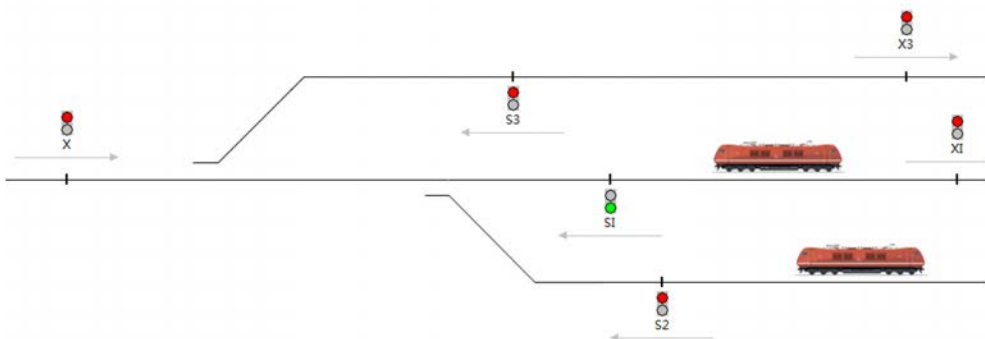


Figure 8: An example of the simulation testing.

Now validating the industrial Data becomes a great challenge which has attracted the researchers' attention. In the literature [13], considering the static and dynamic relationship of interlocking data, a method to transform interlocking data in engineering technical documents into electronic data was proposed. The literature [14], described the successful application of ProB for data validation in Communication-based Train Control (CBTC). We present and evaluate an approach of formal modelling and data validation for railway interlocking system based on the successful research experiences from [12], [14].

7 CONCLUSION AND FUTURE WORK

In this paper, after extracting the environmental, functional and safety properties, as well as each event flow, we built a general interlocking system function model by multistep refinement using the Event-B language.

The formalized model was improved by verifying the system attributes of each layer model. Axioms were proved based on a real station interlocking data, and through the checking of deadlock and the correctness of the invariants, the correctness of the model is further confirmed. We proposed a method to verify the interlocking data based on the universal formalized interlocking model. No additional program is required, and the validation of interlocking data can be effectively carried out by injecting interlock data using the validated model. Based on complementary application of the model checking and theorem proving, the response of the model in the real environment can be observed by simulation technology, which further ensured the completeness and security of the model function, as well as confirmed the validity of the interlocking data.

This approach can help developers to reduce the potential design flaws of the system design requirements in the early development stage, and to identify and correct defects in interlocking data. Validated model can serve as a foundation for coding work, by using code generation technology we can not only improve the degree of automation and correct coding phase, but also reduce the cost of testing stage, which improves the overall safety and reliability of the system.

In the process of formal modelling and validation, knowledge is required on logic, set theory, and strong ability of system analysis and modelling. Possible future research includes: (1) consistency maintenance from requirement text to UML model and Event-B model; and (2) improvement the automation degree for the interlocking data import and validation process, etc.

ACKNOWLEDGEMENT

We would like to thank the National Natural Science Foundation of China (No. 71502146, 61673320), Fundamental Research Funds for the Central Universities (No. 2682017ZT12).

REFERENCES

- [1] Feiler, P. et al., Four pillars for improving the quality of safety-critical software-reliant systems, Carnegie-Mellon University, Pittsburgh Pa, Software Engineering Institute, 2013.
- [2] Larsen, P.G. et al., Formal methods: practice and experience. *Acm Computing Surveys*, **41**(4), (19), pp. 1–36, 2009.
- [3] Abrial, J.R., *Modeling in Event-B: System and Software Engineering*, Cambridge University Press, 2010.
- [4] Snook, C. & Butler, M., UML-B: Formal modelling and design aided by UML. *ACM Transactions on Software Engineering & Methodology*, **15**(1), pp. 92–122, 2006.



- [5] Hoang, T.S., *How to Interpret Failed Proofs in Event-B*, ETH Zurich, Switzerland, 2010.
- [6] Leuschel, M. & Butler, M., ProB: an automated analysis toolset for the B method. *International Journal on Software Tools for Technology Transfer*, **10**(2), pp. 185–203, 2008.
- [7] Ladenberger, L. & Leuschel, M., BMotionWeb: A tool for rapid creation of formal prototypes. *International Conference on Software Engineering and Formal Methods*, Springer, Cham, pp. 403–417, 2016.
- [8] Khan, S.A. et al., Extending petri net to reduce control strategies of railway interlocking system. *Applied Mathematical Modelling*, **38**(2), pp. 413–424, 2014.
- [9] Khan, U. et al., On the real time modeling of interlocking system of passenger lines of Rawalpindi Cantt Train Station. *Complex Adaptive Systems Modeling*, **4**(1), p. 17, 2016.
- [10] Vu, L.H., Haxthausen, A.E. & Peleska, J., Formal modelling and verification of interlocking systems featuring sequential release. *Science of Computer Programming*, **133**, pp. 91–115, 2017.
- [11] Abrial, J.R. et al., Rodin: an open toolset for modelling and reasoning in Event-B. *International Journal on Software Tools for Technology Transfer*, **12**(6), pp. 447–466, 2010.
- [12] Butler, M. et al., Formal modelling techniques for efficient development of railway control products. *International Conference on Reliability, Safety and Security of Railway Systems*, Springer, Cham, pp. 71–86, 2017.
- [13] Bosschaart, M. et al., Efficient formalization of railway interlocking data in RailML. *Information Systems*, **49**, pp. 126–141, 2015.
- [14] Falampin, J., Le-Dang, H., Leuschel, M., Mokrani, M. & Plagge, D., Improving railway data validation with ProB. *In Industrial Deployment of System Engineering Methods*, Springer: Berlin, Heidelberg. pp. 27–43, 2010.

