A Safety Analysis Method for Model Checking Based on Multiple Faults Injection

Xi Wang¹,*, Hui Li¹,a and Chengtian OuYang¹,b

¹School of Information Engineering, Jiangxi University of Science and Technology, Jiangxi, China

*Corresponding author e-mail: wang_xi_happy@163.com

¹297132949@qq.com, boyct@163.com

Abstract. To ensure the full safety of safety-critical system, a safety analysis method for model checking based on multiple faults injection is presented in the paper. In the proposed method, the single and multiple faults can be injected into the formal model of safety-critical system, and an extended formal model with various faults can be acquired. The safety of the extended formal model can be verified, the results of safety analysis can be obtained, thus a minimal cut set of fault tree can be acquired, which violates the safety requirements of the system, and formal safety requirements of the system can be represented by computational tree logic. The method proposed in this paper solves the problem of multi-fault injection and the generation of formal safety requirements in the process of safety analysis, it makes safety analysis for system more comprehensive, so the quality of system’s safety analysis is further improved.

1. Introduction

Formal methods are strongly recommended for the development of safety systems with safety integrity level 4 in EN standard [1]. Formal Method is used to describe and verify the system by mathematical and logical methods in principle. It is an important way to ensure the correctness of system’s requirements specification and design. Formal methods are mainly divided into formal verification and formal description. Model checking is an important method for formal verification [2].

Model checking technology is adopted in literature [2-7] to analyze and verify whether system’s functional requirements and design meet the safety requirements. Due to the rigorous formal verification technology and the ability to automatically analyze and verify the safety of the system, the
efficiency of safety verification has been improved and the quality of safety analysis has been improved. So model checking can make up for the shortcomings of traditional safety analysis methods such as FTA, FMEA and HAZOP. The safety of the system is analyze by combining model checking with single fault injection in literature [3, 5-6], which improve the quality of system’s safety analysis.

Based on the researches mentioned above, whether the functional requirements and the scenarios of the system meet the safety requirements is further studied in the case of multi-fault combination, and a safety analysis method based on multi-fault model checking is proposed in the paper. In the proposed method, a safety analysis process of model checking based on multi-fault injection is presented firstly, then model checking algorithms based on multiple faults injection is proposed. On the basis of formal extended model of system with single or multiple faults is generated, the method for model checking is used to verify the safety of formal extended model of system, and the results of safety verification is obtained, the formal safety requirements specifications of the system are obtained accordingly. Finally, this paper takes the route establishment subsystem of railway station interlocking system as a case study, and the proposed method is applied to the safety analysis and verification of the system. The experimental results show that the proposed safety analysis method for model checking based on multiple faults injection is correct and feasible.

2. The safety analysis method for model checking based on multiple faults injection

2.1. The process of safety analysis for model checking based on multiple faults injection

In order to ensure the adequate safety of safety-critical computer systems, it is necessary to study whether the functional behavior meets the safety requirements in the normal cases, and whether the control logic of the software meets the specified safety requirements when the system is failure. In this section, a safety analysis method for model checking based on multi-fault injection is proposed, and the process of safety analysis is shown in Figure 1 accordingly.

According to Figure 1, the safety analysis method for model checking based on multi-fault injection includes the following steps. (1) The requirement scenario of the system is analyzed and modeled with finite state model (FSM for short). (2) A preliminary functional hazard analysis of system’s requirement scenario is done, and the basic safety requirements and the fault behavior patterns are acquired. (3) The fault defined by user and the acquired fault behavior patterns mentioned above are added into a fault pattern library. (4) The method for model checking is used to verify whether FSM meets the basic safety requirements. If it does not, the requirement scenario is modified and refined according to the counterexamples generated from the verification results, and the above process is repeated. (5) If FSM meets the basic safety requirements, single fault and multiple faults selected from the fault mode library are injected into FSM to form an extended finite state model of the system (EFSM for short). The algorithm of fault injection in this step is more complex and should be represented in detail in Section 2.2. (6) Verifying whether EFSM meets the basic safety requirements, and the results of safety analysis are obtained. (7) The formal safety requirements of the system can be obtained.

2.2. Model Checking Algorithm Based on Multiple Fault Injection

The main fault modes affecting system safety are single fault modes and multiple faults modes. In this part, the fault injection algorithm for these two types of fault modes is proposed, it is described as
follows. (1) The fault behavior pattern for each fault record is read from the fault mode library in turn, and the single fault injection and model extension are carried out to obtain the EFSM with single fault. (2) Verifying whether EFSM with single fault violates the safety requirements of the system. If it violates, the name of the component, the ID identifier of the fault and the result of the safety verification will be written to the result table of safety verification. Otherwise, the ID identifier of the fault will be stored in the multi-fault table for further multi-fault injection and verification. (3) Judging whether the safety verification for single fault injection has been finished, if not, redo steps (1) and (2). Otherwise, do the next step. (4) Selecting a fault record from multiple fault tables, combining it with single fault that does not violate the safety requirements in the fault mode library in turn, and adding the combination of multiple faults to FSM, then the EFSM with multiple faults is got. (5) Verifying whether the EFSM with multiple faults violates the safety requirements. If it violates, the name of components, the ID identifier of multiple faults and the results of safety verification are written into the results table. Otherwise, ID identifier of multiple faults is stored in the multi-fault table. (6) Judging whether the multi-fault injection has been finished, if not, redo step 4 and 5 to continue the safety verification with multi-fault injection. Otherwise, the multi-fault records in the result table are analyzed for obtaining the minimum cut set of multiple faults. (7) The results of safety verification in the table are output in turn.

Figure 1. The process of safety analysis

3. A Case Study
In railway station interlocking system [7], the system is mainly composed of the components for User, Interlocking Control Center (ILController for short), Switch, Section and Signal. Detailed description of requirements scenarios is represented in literature [7]. The corresponding state transition model of the components is described in Table 1, where Q0 is the initial state, Qi (i < N) is the state, the symbol "->" represents the transition, the word before the symbol "->" represents the action. After performing
combination operations in LTSA [8], the formal model of the system is obtained as shown in Figure 2. From Figure 2, we can see that there is no error mark “-1”, which indicates that there is no deadlock phenomenon of cyclic waiting caused by competing resources in the process of route establishment. All functions and behaviors of the system are usually achievable. Functional simulation in LTSA shows that the formal model of the system can meet the functional requirements.

Through the preliminary functional hazard analysis for the requirement scenario, the formal model for the basic safety requirement is shown in Figure 3, the acceptable set $F$ is as follows.

$$F=\{\{\text{routeSelect}, \text{switchInCorrect}, \text{routeSetUpOK}\}, \{\text{routeSelect}, \text{switchLockFailure}, \text{routeSetUpOK}\}, \{\text{routeSelect}, \text{conflictRouteExisting}, \text{routeSetUpOK}\}, \{\text{routeSelect}, \text{sectionOccupied}, \text{routeSetUpOK}\}, \{\text{routeSelect}, \text{conflictSignalClear}, \text{routeSetUpOK}\}\},$$

$F$ is a acceptable set consisting of multiple subsets of preventable safety conditions in the system, the condition for state transition “sectionOccupied” indicates the section for route building is occupied, “conflictSignalClear” indicates opening conflict signal light, and “switchLockFailure” indicates locking switch failure.

### Table 1. Description of state transition model for components

| Component | State Transition Model |
|-----------|-----------------------|
| **User** | $Q_0$, $Q_0= (\text{routeSelect} \rightarrow Q_1), Q_1= (\text{routeSetUpOK} \rightarrow Q_0), Q_1= (\text{routeSetUpFailed} \rightarrow Q_2), Q_2= (\text{stop} \rightarrow Q_0).$ |
| **II.Controller** | $Q_0$, $Q_0= (\text{routeSelect} \rightarrow Q_1), Q_1= (\text{conflictRouteCheck} \rightarrow Q_2), Q_2= (\text{routeCheck} \rightarrow Q_3), Q_3= (\text{switchCheck} \rightarrow Q_4)$ |
| **Switch** | $Q_0$, $Q_0= (\text{switchCheck} \rightarrow Q_1), Q_1= (\text{switchInCorrect} \rightarrow Q_2), Q_2= (\text{switchCorrect} \rightarrow Q_4)$ |
| **Section** | $Q_0$, $Q_0= (\text{conflictRouteCheck} \rightarrow Q_1), Q_1= (\text{routeCheck} \rightarrow Q_4)$ |
| **Signal** | $Q_0$, $Q_0= (\text{signalClear} \rightarrow Q_1), Q_1= (\text{signalCleared} \rightarrow Q_0).$ |
Figure 3. The formal model for basic safety requirement of the system

Figure 3 shows that after the message for routeSelect is sent, the route can’t be established when conflict route is existed, switch is not in the specified location, locking switch is failure, section is occupied, conflict signal is open and so on. In the model checking tool for LTSA, the formal models shown in Figure 2 and Figure 3 are calculated by synchronous product operation, then an ETGBA model [9] is generated accordingly, which contains 119 state nodes and 210 transitions. The safety of ETGBA model is verified by using the previous research method [9] which is proposed by the author of this paper. The result for verification is "No, empty ETGBA", and the time required is 0.0056 second, and the required memory is 8872 KB. Because the result of safety verification is "No", the ETGBA is empty, which indicates that the control logic of the system meets the safety requirements under normal circumstances.

The fault modes selected in this paper is shown in Table 2. According to the safety analysis process proposed in this paper, if the formal model of the system is injected with the fault "ILController_F2", the corresponding formal models with the fault for Switch and ILController is shown in Figure 4 and Figure 5. The formal models of the component for User, Section, Signal, Figure 4, Figure 5 and Figure 3 are operated by combination in LTSA, then the new ETGBA model is generated which contains 214 state nodes and 427 transitions. The result of safety verification is "yes, ETGBA is not empty", and the path for corresponding counterexample is “routeSelect → sectionOccupied → noConflictRouteSetUp → routeCheck → routeEmpty → switchCheck → routeLock → switchLock → switchLockFail → signalClear → signalCleared → routeSetUpOK → routeSelect”, which violates the subsets for \{routeSelect, switchLockFailure, routeSetUpOK\} and \{routeSelect, sectionOccupied, routeSetUpOK\} in the set \(F\). Which indicates the route shouldn’t be built when locking switch is failure and track section is occupied, otherwise, the system is unsafe.

Figure 4. Extended formal model of switch

switchCorrect

Switch

switchCheck

switchInCorrect, switchLock

switchLocked

Figure 4. Extended formal model of switch
According to the process mentioned above, after the system’s formal model injected with failures in Table 2, the results of safety analysis are obtained as shown in Table 3. The results in Table 3 are consistent with the requirements of the relevant technical conditions stipulated in the railway industry standard [10]. Therefore, the results of the experiments show that the proposed safety analysis process and the corresponding algorithm are correct and feasible.

Table 2. Corresponding fault modes for fault injection

| Name               | Fault ID    | Representation for fault                                                                 | Behavior mode for fault                                                                 |
|--------------------|-------------|----------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------|
| ILController       | ILController_F1 | Signal light is open when locking switch is failure.                                    | ILController_F1=(switchLock->switchLockFailure->signalClear->signalCleared->routeSetUpOK). |
| ILController       | ILController_F2 | Signal light is open when the switch is not positioned as required, but in reverse position. | ILController_F2 = (switchTransact-> switchInCorrect -> signalClear->signalCleared->routeSetUpOK). |
| ILController       | ILController_F3 | Signal light is open when the switch is not positioned as required, but in the state of four openings. | ILController_F3 = (switchTransact -> switchInCorrect -> signalClear->signalCleared->routeSetUpOK). |
| ILController       | ILController_F4 | Signal light is open when the switch is not in reverse position as required, but in positioning. | ILController_F4=(switchTransact-> switchInCorrect-> signalClear->signalCleared->routeSetUpOK). |
| ILController       | ILController_F5 | Signal light is open when the switch is not in reverse position as required, but in the state of four openings. | ILController_F5= (switchTransact ->switchInCorrect -> signalClear->signalCleared->routeSetUpOK). |
| ILController       | ILController_F6 | Signal light is open when conflict route is existed.                                    | ILController_F6=(conflictRouteExisting->signalClear->signalCleared->routeSetUpOK). |
| ILController       | ILController_F7 | Signal light is open when the section is occupied.                                      | ILController_F7=(sectionOccupied->signalClear->signalCleared->routeSetUpOK). |
| ILController       | ILController_F8 | Signal light is open the conflict signal is open.                                       | ILController_F8=(conflictSignalClear-> signalCleared->routeSetUpOK). |
4. The formal safety requirement specification of the system
According to the results of safety analysis shown in Table 3 and the minimum cut set theory of fault tree, the formal safety requirement "SRS_Interlock" described by CTL is as follows:

$$SRS_{\text{Interlock}} = \neg\text{EF ILController}_F1 \land \neg\text{EF ILController}_F2 \land \neg\text{EF ILController}_F3 \land$$
$$\neg\text{EF ILController}_F4 \land \neg\text{EF ILController}_F5 \land \neg\text{EF ILController}_F6 \land$$
$$\neg\text{EF ILController}_F7 \land \neg\text{EF ILController}_F8$$

5. Conclusion
In the paper, a method for model checking based on multi-fault injection is proposed, which not only considers whether system's functional behavior meets the safety requirements, but also whether the system is safety under single fault and combination of multiple faults conditions. The process of safety analysis is represented in detail. From safety analysis for route establishment subsystem in railway station interlocking system, it can be seen that the analysis results in this paper are consistent with the requirements of the relevant technical conditions stipulated in the current railway industry standard [10], which shows that the method proposed in this paper is correct and feasible. Finally, based on the results of safety analysis, the formal safety requirements of the system expressed by CTL are obtained. Compared with the existing safety analysis methods, the proposed safety analysis process and corresponding algorithms can not only verify the impact of hardware failure on safety, but also verify the safety of control logic of system software under normal circumstances. So the proposed method can improve the quality and efficiency of system’s safety analysis and verification, it provides some technical support for safety assurance and safety assessment for safety-critical systems.
References

[1] CENELEC, EN50128, Railway Applications- Software for Railway Control and Protection Systems, CENELEC, 2001.

[2] M. Meng, Z. Xu, W. Xi, et al. Model checking-based safety verification for railway signal safety protocol-I, International Journal of Computer Applications in Technology. 46 (2013) 3 195-202.

[3] L. Rogovchenko-Buffoni, A. Tundis, M. Z. Hossain, et al. An integrated toolchain for model based functional safety analysis, Journal of Computational Science. 5 (2014) 3 408-414.

[4] S. Gulan, J. Harnisch, S. Johr, et al. Model-Based Analysis for Safety Critical Software, International Conference on Computer Safety, Reliability, and Safety, 2014, pp. 111-120.

[5] S. Hordvik, K. Øseth, J. O. Blech, et al. A Methodology for Model-based Development and Safety Analysis of Transport Systems, International Conference on Evaluation of Novel Software Approaches To Software Engineering, 2016, pp. 91-101.

[6] O. El Ariss, D. Xu, W. E. Wong, Integrating Safety Analysis With Functional Modeling, IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 41 (2011) 4 610-624.

[7] Zhixi Zhao, Computer Interlocking System Technology, China railway publishing house, 1999.

[8] Magee J, Kramer J, Associated Concurrency: State Models and Java Programs, Wiley, 1999.

[9] Wang xi, Xu zhong-wei, Heuristic on-the-fly Model Checking Algorithm for Extended TGBA, Chinese Journal of Computer Science, 37 (2014) 12 2519-2529.

[10] Ministry of Railways of the People's Republic of China, Technical conditions for TB3027--2002 computer interlocking, China railway publishing house, 2002.