A formal modeling method based on multiple composite scenarios analysis for railway station interlocking system

Xi Wang¹,a, Cheng-tian OuYang¹ and Pei-pei Li¹

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

Corresponding author: wang_xi_happy@163.com, oyct@163.com, 947135345@qq.com

Abstract. In order to ensure the safety of railway station interlocking system, the subsystem for route creating is used as a case study, and a formal modeling method based on multiple composite scenarios analysis is provided. In the proposed method, multiple UML sequence diagrams are adopted to specify requirements of the system, and consistent requirements specification are acquired by combining pre-condition and post-condition of object constraint language with domain knowledge to analyze conflicts in multiple UML sequence diagrams. Besides, a model conversion method for transforming behaviour sequences to finite state process model is proposed, despite these behaviour sequences are usually synchronous, asynchronous, concurrency and alternate, which interacted with each other in UML sequence diagram. Finally, an FSP model for system’s formal model is generated by combinatorial operation, it is conformed to system’s functional requirements. The correctness and feasibility of the proposed method have been confirmed by formal model generation for railway station interlocking system.

1. Introduction

System’s safety is particularly important in the field of safety critical computer systems[1]. In some safety standards such as EN50128[2] and EN50126[3], formal method is strongly recommended for software requirements specification and software design. Railway station interlocking system is a safety critical system, how to apply formal method and technology to this field has become a hot topic in academia and industry.

Functional behavior and safety protection for railway station interlocking system are usually verified and validated mainly by test[4-5], simulation[6-7] and formal verification[8-10]. These methods are beneficial to the safety analysis, verification and validation. However, testing and simulation can only prove that the system is wrong and can’t prove the system is not wrong. Because the soundness and completeness for testing and simulation are needed to depend on how to select test vectors, it is a very arduous task to select reasonably and fully test vector. Using formal methods to analyze and verify system’s safety is an important way for constructing reliable and safety software system[11-13], but the accuracy of formal modeling will directly affect results for safety analysis and verification. Safety analyst are usually required to master skilled formalization technique when they adopt formal modelling method.
To provide an effective way for formal modeling, the subsystem for route creating in railway station interlocking system is used as an example, and a method for formal modeling based on multi-scenario analysis is proposed. At first, this paper makes consistent analysis for multiple combined sequence diagrams which are described in unified modeling language (UML for short), then transforms messages in multiple combinatorial UML sequence diagram into finite state process (FSP for short) model, and finally obtains a formal model for the system by combinatorial operation. An effective way to solve the difficult problem for formal modeling is provided, it is beneficial to improve the design and development of safety critical software.

2. Scenario description for railway station interlocking system

In railway station interlocking system [13], from the operator on duty sends the message of preparation for route selection to the signal light of safety protection is opened, there are six stages which are included in the literature [13]. This paper use UML sequence diagrams to describe multiple composite scenarios of route creating as shown in Figure 1 and Figure 2, the operator for alt is indicated that behaviors divided by dotted dashed line in the box are optional, specific meaning of each message is described as follows.

Figure 1 Composite scenario 1 of UML sequence diagram
Operator | Interlocking Controller | Turnout | Track Section | Signal Light
--- | --- | --- | --- | ---
routePreSelect | turnoutCheck | turnoutCheck | turnoutCorrect | turnoutTransactionErro
routeCreateFailed | turnoutCorrect | turnoutTransact | turnoutCorrect | turnoutErro
routeCheck | turnoutCreateFailed | conflictRouteCheck | conflictRouteCreateFailed | conflictRouteExisting
routeCreatFailed | conflictRouteCreateFailed | conflictRouteLocked | conflictRouteLocked | routeLocked
routeCreatFailed | conflictRouteLocked | signalClear | signalClear | routeCreatFailed
routeCreatOK | signalCleared | routeCreatOK | routeCreatOK | routeCreatOK

Figure 2. Composite scenario 2 of UML sequence diagram

The message for routePreSelect indicates preparation for route selection, the message for turnoutCheck indicates checking turnout, the message for turnoutErro indicates turnover is not in correct position, the message for turnoutTransact indicates converting turnover, the message for turnoutCorrect indicates turnover is in correct position, the message for turnoutTransactionErro indicates that turnover is not converted to correct position, the message for routeCreateFailed indicates creating route is failed, the message for routeCheck indicates checking route, the message for routeEmpty indicates route is empty, the message for routeOccupied indicates route is occupied, the message for conflictRouteCheck indicates checking conflict route, the message for noConflictRouteCreat indicates conflict route is not built, the message for conflictRouteExisting indicates existing conflict route, the message for turnoutLocked indicates locking turnout, the message for turnoutLocked indicates turnover is locked, the message for routeLocked indicates locking route, the message for routeLocked indicates route is locked, the message for signalClear indicates opening signal light, the message for signalCleared indicates signal light is opened, the message for routeCreatOK indicates route is built successfully.

3. Method based on multi composite scenarios analysis

Method based on multi composite scenarios analysis is described as follows:

1. According to scenario description of system’s functional requirement, global variables that need to be defined in the process of system design are combined into a state vector, the form of the state vector is $v_i = \langle v_{i,1}, v_{i,2}, \ldots, v_{i,n} \rangle$, $v_{i,j}$ ($1 \leq j \leq n$) is a state variable.

2. Writing out pre/post-conditions of object constraint language (OCL pre/post-conditions for short) that must be meet by message in UML sequence diagram.

3. $v_0$ means initial state vector, $pre_v_i$ means the pre-state vector value generated by the OCL pre-condition of the message $m_i$, $post_v_i$ means the post-state vector value generated by the OCL post-condition of the message $m_i$.

4. The value of pre/post-state vector for all messages in the UML sequence diagram is generated by the rules: Rule one: if $v_{0,j}$ is undefined, then $v_{0,j}$ is null; Rule two: if the value of $pre_v_{i,j}$ is $x$, then $pre_v_{i,j} = x$; Rule three: If the value of $post_v_{i,j}$ is $y$, then $post_v_{i,j} = y$; Rule four: If $pre_v_{i,j}$ is undefined, then $post_v_{i,j} = post_v_{i,j}$; Rule five: If $post_v_{i,j}$ is undefined, then $post_v_{i,j} = pre_v_{i,j}$. 

3
(5) Contradictions between messages are detected according to two cases: one is for a UML sequence diagram, if \( m_{i-1} \) and \( m_i \) are two sequential messages, and \( post_{vi-1} \neq pre_{vi} \), then there are contradictions between \( m_{i-1} \) and \( m_i \). The other case is for both sequence diagram UML\( i \) and UML\( j \), it is assumed that UML\( i\cdot m_i \) is a message in UML\( i \), UML\( j\cdot m_j \) is a message in UML\( j \), UML\( i\cdot m_i \) and UML\( j\cdot m_j \) are the same message, if the value of state vector for UML\( i \cdot m_i \) and UML\( j \cdot m_j \) are not equal, then there are contradictions between UML\( i \) and UML\( j \).

(6) Analyzing contradictions based on domain knowledge, if scenario description of system’s functional requirement is incorrect, then the scenario description is modified, otherwise, the contradictory value of OCL pre/post-state vector is modified.

(7) After the contradictions are eliminated in UML sequence diagram, a consistent UML sequence diagram are acquired.

(8) For UML\( i \) and UML\( k \), if the message \( m_i \in \text{UML}_i \), \( m_k \in \text{UML}_k \), \( m_i \neq m_k \) and the value of state vector for \( m_i \) and \( m_k \) are equal, then \( m_i \) and \( m_k \) are concurrent or optional behavior. Combining the same or similar behavior in the UML sequence diagram to generate a new UML sequence diagram named UML\_new. If concurrent or optional behavior is encountered when merging, then adding to the corresponding interactive box in UML\_new.

If \( m_i \in \text{UML}\_\text{new} \), \( m_j \in \text{UML}\_\text{new} \), and the value of post-state vector for \( m_i \) is equal to the value of pre-state vector for \( m_j \), then \( m_i \) and \( m_j \) are sequential behavior, otherwise, the concurrent behavior and the optional behavior in UML\_new are identified based on the domain knowledge.

According the method mentioned above, the state vector for route creating subsystem is the form: \(< \text{StateTurnoutCorrect}, \text{StateRouteOccupied}, \text{StateConflictRouteExisting}, \text{StateRouteLocked}, \text{StateTurnoutLocked}, \text{StateSignalCleared}, \text{StateRouteCreate} > \), whose variables are Boolean. The boolean variable for StateTurnout-Correct indicates whether turnout is in the correct location, the variable for StateRouteOccupied indicates whether the route is occupied, the variable for StateConflictRoute-Existing indicates whether there is a conflict route, the variable for StateTurnoutLocked indicates whether the route is locked, the variable for StateSignalCleared indicates whether signal light is opened, the variable for StateRouteCreate indicates whether the route is set up.

By analyzing OCL pre/post-conditions of messages in Figure 1 and Figure 2, Generated contradictory pre/post-state vector values of messages in Figure 1 and Figure 2 are shown in table 1 and table 2.

In table 1 and table 2, there are contradictions as "Conflict2", "Conflict3", "Conflict4" and "Conflict5", "Conflict1" is a contradiction between table 1 and table 2. "Conflict-a1" is in table 1, "Conflict-b1" and "Conflict-b2" are in table 2. Analysing these contradictions, it can be found that turnoutCheck, routeCheck and conflictRouteCheck are concurrent messages, which can appear alternately on an execution path, turnoutCorrect and turnoutError are optional messages, which appear after turnoutCheck is sent out, routeEmpty and routeOccupied are optional messages, which appear after routeCheck is sent out, noConflictRouteCreate and conflictRouteExisting are optional messages that appear after conflictRouteCheck is sent out, the time sequence of the rest of the messages is unchanged. Then scenario description of system’s functional requirement is modified to get the new UML sequence diagram as shown in Figure 3, The operator for par represents concurrency, the operator for alt represents alternative, dotted line in the box is a separate line.
Table 1 Pre/post-state vector values of messages in Figure 1

| message          | pre-state vector value | post-state vector value |
|------------------|------------------------|-------------------------|
| turnoutCorrect   | <null,null,null,null,f,f,f> | <t,null,null,null,f,f,f> |
| routeCheck       | <null,null,null,null,f,f,f> | <null,null,null,null,f,f,f> |
| routeOcupied     | <null,null,null,null,f,f,f> | <null,t,null,null,f,f,f> |
| routeCreatFailed | <null,t,null,null,f,f,f> | <null,t,null,null,f,f,f> |
| conflictRouteCheck| <null,null,null,null,f,f,f> | <null,null,null,null,f,f,f> |
| noConflictRouteCreat| <null,null,null,null,f,f,f> | <null,null,null,null,f,f,f> |
| conflictRouteExisting| <null,null,null,null,f,f,f> | <null,null,t,null,f,f,f> |
| routeCreatFailed | <null,null,null,null,f,f,f> | <null,null,null,null,f,f,f> |
| turnoutLock       | <t,f,f,f,f,f,f> | <t,f,f,f,f,f> |

Table 2 Pre/post-state vector values of messages in Figure 2

| message          | pre-state vector value | post-state vector value |
|------------------|------------------------|-------------------------|
| turnoutCorrect   | <f,null,null,null,null,f,f,f> | <null,null,null,null,f,f,f> |
| turnoutTransactionError | <f,null,null,null,null,f,f,f> | <null,null,null,null,f,f,f> |
| routeCreatFailed | <f,null,null,null,null,f,f,f> | <null,null,null,null,f,f,f> |
| routeCheck       | <null,null,null,null,f,f,f> | <null,null,null,null,f,f,f> |
| routeOcupied     | <null,null,null,null,f,f,f> | <null,t,null,null,f,f,f> |
| routeCreatFailed | <null,t,null,null,f,f,f> | <null,t,null,null,f,f,f> |
| conflictRouteCheck| <null,null,null,null,f,f,f> | <null,null,null,null,f,f,f> |
| noConflictRouteCreat| <null,null,null,null,f,f,f> | <null,null,null,null,f,f,f> |
| conflictRouteExisting| <null,null,null,null,f,f,f> | <null,null,t,null,f,f,f> |
| routeCreatFailed | <null,null,null,null,f,f,f> | <null,null,null,null,f,f,f> |
| turnoutLock       | <t,f,f,f,f,f,f> | <t,f,f,f,f,f> |

Table 3 FSP model for each component of route creating

| Component          | Corresponding FSP model description |
|--------------------|-------------------------------------|
| Operator           | Operator = (routePreSelect->Operator temp), Operator temp=(routeCreatFailed->stop->Operator |routeCreatOK->Operator). |
| Inter-locking       | InterlockingController=(routePreSelect->Temp), Temp=(turnoutCheck->Temp11 |routeCheck->Temp21 |conflictRouteCheck->Temp31), Temp11=(turnoutCorrect->TurnoutCorrectTemp |turnoutError->turnoutTransaction->TurnoutTransactionTemp), TurnoutTransactionTemp=(turnoutCorrect->TurnoutCorrectTemp |turnoutTransactionError->RouteCreatFailed), RouteCreatFailed=(routeCreatFailed->InterlockingController), TurnoutCorrectTemp=(routeCheck->TCAndrouteCheckTemp |conflictRouteCheck->TCAndCRCrTemp), TCAndrouteCheckTemp=(routeEmpty->conflictRouteCheck->TCAndrouteCheckTemp2 |routeOccupied->RouteCreatFailed), TCAndrouteCheckTemp2=(conflictRouteExisting->RouteCreatFailed |noConflictRouteCreat->TurnoutLock), TCAndCRCrTemp=(conflictRouteExisting->RouteCreatFailed |noConflictRouteCreat->routeCheck->TCAndCRCrTemp2), TCAndCRCrTemp2=(routeOccupied->RouteCreatFailed |routeEmpty->TurnoutLock), Temp21=(routeEmpty->RouteEmptyTemp |routeOccupied->RouteCreatFailed), |
RouteEmptyTemp=(turnoutCheck->Temp211 |conflictRouteCheck ->Temp212),
Temp211=(turnoutCorrect->Temp211TurnoutCorrect |turnoutError->turnoutTransaction->Temp211TurnoutTransaction), Temp211TurnoutCorrect=(conflictRouteCheck->Temp211CRC), Temp211CRC=(conflictRouteExisting->RouteCreatFailed |noConflictRouteCreat->TurnoutLock),
Temp211turnoutTransaction=(turnoutTransactionError->RouteCreatFailed | turnoutCorrect->conflictRouteCheck->Temp211CRC),
Temp212=(conflictRouteExisting->RouteCreatFailed | noConflictRouteCreat->turnoutCheck->Temp212TC),
Temp212TC=(turnoutCorrect->TurnoutLock | turnoutError->turnoutTransaction->Temp212TC2),
Temp212TC2=(turnoutCorrect->TurnoutLock | turnoutTransactionError->RouteCreatFailed),
Temp31=(noConflictRouteCreat->NoConflictRouteCreatTemp | conflictRouteExisting->RouteCreatFailed),
NoConflictRouteCreatTemp=(turnoutCheck->Temp31TurnoutCheck | routeCheck->Temp31RouteCheck),
Temp31TurnoutCheck=(turnoutCorrect->routeCheck->TCAndCRCTemp2 | turnoutError->turnoutTransaction->Temp31TTTemp),
Temp31TTTemp=(turnoutTransactionError->RouteCreatFailed | turnoutCorrect->routeCheck->TCAndCRCTemp2),
Temp31RouteCheck=(routeEmpty->turnoutCheck->Temp31TCTemp | routeOccupied->RouteCreatFailed),
Temp31TCTemp=(turnoutCorrect->TurnoutLock | turnoutError->turnoutTransaction->Temp212TC2),
TurnoutLock=(turnoutLock->turnoutLocked->routeLocked->signalClear->signalCleared->routeCreatOK ->InterlockingController).

Turnout

| Outcome                              |
|--------------------------------------|
| Turnout=(turnoutCheck->TurnoutTemp1), |
| TurnoutTemp1=(turnoutCorrect->TurnoutTemp2 | turnoutError->turnoutTransaction->TurnoutTemp3), |
| TurnoutTemp2=(turnoutLock->turnoutLocked->Turnout), |
| TurnoutTemp3=(turnoutCorrect->TurnoutTemp2 | turnoutTransactionError->Turnout). |

Track Section

| Outcome                              |
|--------------------------------------|
| TrackSection=(routeCheck->SectionTemp1 | conflictRouteCheck->SectionTemp2 ), |
| SectionTemp1=(routeEmpty->conflictRouteCheck->SectionTemp1a | routeOccupied->routeLock->trackSection | conflictRouteExisting->trackSection), |
| SectionTemp1a=(noConflictRouteCreat->routeLock->trackSection | conflictRouteExisting->trackSection), |
| SectionTemp2=(noConflictRouteCreat->SectionTemp2a | conflictRouteExisting->trackSection), |
| SectionTemp2a=(routeCheck->SectionTemp2b), |
| SectionTemp2b=(routeEmpty->routeLock->routeLocked->trackSection | routeOccupied->trackSection). |

System Mode

| Outcome                              |
|--------------------------------------|
| SystemModel=(Operator || InterlockingController || Turnout || TrackSection|| SignalLight). |
4. Formal modeling of railway station interlocking system

According to Figure 3, this paper uses FSP model to describe components of route creating as shown in Table 3. Transition condition "stop" indicates that route creating is stopped temporarily, specific meaning of other transition condition is shown in Section 2.

Formalized model of the system is SystemModel, which is described in Table 3. After performing combinatorial operation in the model checking tool for LTSA\cite{14}, 108 states and 141 transitions are acquired in SystemModel, and the error flag "-1" don’t appear, which indicates that there is no death lock caused by the loop waiting during the route creating. Generally, every function and behavior of the system can be realized. By function simulation in LTSA, it is found that formalized model of the system can meet functional requirements under normal conditions. Results of the test also show that no route can be set up under the circumstances of route is occupied, conflict route is set up, turnout is not in correct location, turnout locking is failed, track section is occupied, and conflict signal is opened, etc. Which are in accordance with the technical conditions that are stipulated in the railway industry standard TB/T 3027-2002\cite{15}. So the proposed formal modeling method in this paper is correct and feasible.

5. Conclusion

This paper uses OCL pre/post-condition to analyse operation scenarios of railway station interlocking system, and provide a method for transforming multiple composite UML scenarios to system’s FSP model. Besides, a formal modeling method based on multiple composite scenarios analysis is proposed. Proven by experiments, the method is correct and feasible, which not only provides an
effective way to solve the problem for formal modeling in the field of safety critical, but also provides a new thought for generating formal model for formal verification and analysis. Design and development of safety critical software can be improved by the proposed method, which is an important part of model-based formal development of software.

Acknowledgment
This work has been supported by the national natural science foundation of China(No. 61462034,No.61561024,No.11461031), Jiangxi Provincial Natural Science Foundation of China(No. 20151BAB207035), and Scientific Research Fund of Jiangxi Provincial Education Department,China(No. GJJ160632, No. GJJ170517).

References
[1] N. Storey, Safety Critical Computer Systems, (Addison Wesley, 1996)
[2] EN 50128, Railway Applications: Communications, signalling and processing systems-- Software for railway control and protection systems (CENELEC, 2001)
[3] IEC61508, Functional Safety of electrical/ electro-nic/programmable electronic safety related systems-part3: software requirements ( IEC, 2000)
[4] F. Scippacercola, R. Pietrantuono, S. Russo, CCIS, 580, 375 (2015)
[5] I. Mutlu, T. Ovatman, M. Turan Söylemez, TMEE (IEEE, 2012)
[6] X.S.Wu, Acta Technica, 62, 313 (2017)
[7] H. Jiang, X. Bian, Y. Chen, et al. China Civil Engineering, 48, 85 (2015)
[8] W. Xi, X. Zhong-wei, M. Meng, Wuhan publishing house, 56, 156 (2010)
[9] E. Dincel, O. Eris, S. Kurtulan, Antennas Propagat. Mag., 55, 308 (2013)
[10] S. Busard, Q. Cappart, C. Limbrée, etc, Proc. ESSS 2015, 184 (2015)
[11] A. Bonacchi, A. Fantechi, QUATIC 2014, 226 (2014)
[12] C. Limbré, Q. Cappart, C. Pecheur, etc. Springer International Publishing, 134 (2016)
[13] W. Xi, X. Zhong-wei. Journal of the china railway society, 34, 67 (2012)
[14] J. Magee, J.Kramer, Concurrency: state models & Java programs( John Wiley & Sons, Inc. 2000)
[15] Ministry of Railways of the People's Republic of China, Technical conditions for TB5027--2002 computer interlocking (China railway publishing house, 2002)