

A Verification Method for Software Safety Requirement by Combining Model Checking and FTA

Congcong Chen^{1,a}, Fuping Zeng^{1,b}, Minyan Lu^{1,c}

¹School of Reliability and System Engineering, Beihang University, Beijing 100191, China

^aemail: ccchappiness@yeah.net, ^bemail: zfp@buaa.edu.cn, ^cemail: lmy@buaa.edu.cn

Keywords: Model Checking; Software Safety; Formal Method; Fault Tree Analysis

Abstract. Formal verification for safety-critical software requirements is used to improve the safety of software system. Model checking precisely verifies the related properties of software system, but there still exist limitations in properties extraction which is traditionally obtained by artificial definition. In this paper, a combined technology of model checking and FTA (Fault Tree Analysis) is applied to the software safety requirements verification, mainly to solve the problem of properties extraction in model checking. First the software system model is described in finite state machine (FSM). Second the safety properties are achieved by combining FTA and Computational Tree Logic (CTL) of model checking. Next the open source tool of model checker NuSMV is applied to implement the verification of software safety requirements. Finally this methodology is illustrated with an application to an ABP (Alternating Bit Protocol) instance. The application results can show the effectiveness of the method.

Introduction

Among all the phases in software development, the requirements analysis phase is generally considered to play the most critical role in determining the overall software safety, because mistakes made during the requirements analysis phase easily introduce faults that subsequently lead to accidents [1]. The verification and validation (V&V) process in the requirements phases checks the correctness of the software requirements specifications. However, the conventional requirements verification methods such as manual review, analysis and testing for safety analysis of software requirements have problems in terms of correctness and efficiency. Formal method [2] has been proved to be an effective method to reduce errors and an important way to improve the software dependability. Moreover it has been gradually applied into safety-critical software systems in some aerospace research institutions.

There are two major verification approaches, theorem proving and model checking. Compared with the theorem proving, model checking has the advantages of high automation and high detection efficiency [3][4][5][6]. The application of model checking can break through the subjective limitation of human brain and eliminate the subjective differences among people. Hence model checking is a powerful and effective way of achieving strict verification of software safety requirements. In recent years, although model checking has been a huge success applied in the field of formal verification, it is not mature to be applied in the field of safety-critical software especially in the application of safety requirements verification. The extraction of software safety properties is more performed by experience [7][8].

Fault tree analysis is one of the most frequently safety analysis techniques applied in the development of safety-critical industrial system [9][10]. It represents the interaction of failures and other events within a system graphically. If the minimum cut sets obtained from the FTA are regarded as the system safety properties to be verified, it is bound to play a vital role to improve the safety of safety-critical system.

Therefore in this paper, we put forward a combined method of model checking and FTA for verifying software safety requirements, focusing on solving the problem of safety properties extraction by combining FTA and Computational Tree Logic (CTL) of model checking.

The remaining of this paper is organized as follows. Section 2 briefly summarizes the method. In

section 3, we use a protocol example to verify the feasibility of the method and give the related verification results and analysis. Finally, summary and conclusion are provided in section 4.

Method

2.1 Basic idea

Model checking is a formal method which can precisely prove that a system can work properly in accordance with the intended target. Its basic idea can be concluded as below: First, we need to build the system model for system, and then use a formalized language such as temporal logic expressions to describe system properties, finally use the model checking technology to analyze whether the system model (M) meets system properties(S), namely $M \models S$. The general process is shown in Figure 1.

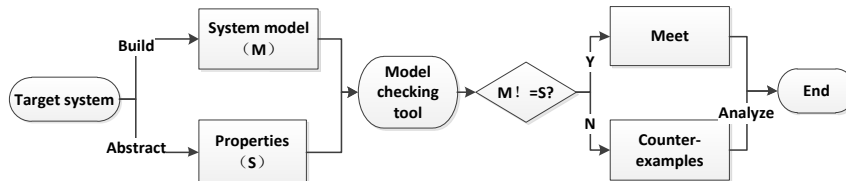


Fig.1. The flow chart of model checking

In this paper, model checking technology is applied to the verification of software safety requirements. First, the finite state machine (FSM) will be built based on the related software documents. Second, software safety properties will be obtained by combining the fault tree analysis (FTA) and the computation tree logic (CTL)of model checking. Finally, the automatic model checker will be used to verify whether the system model meets the safety properties. The method flow chart is shown in Figure 2.

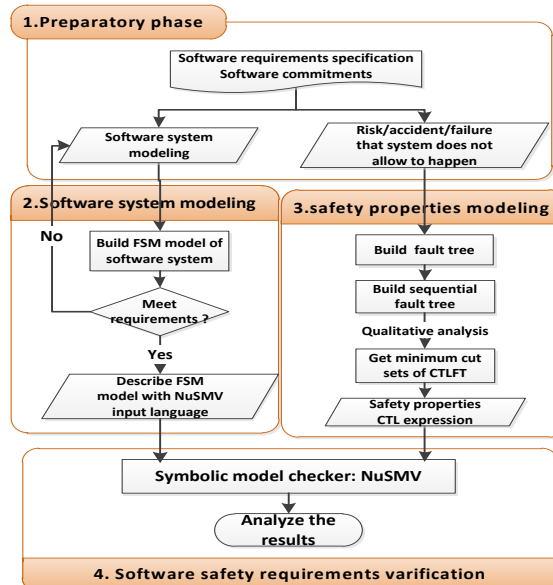


Fig.2. The flow chart of software safety requirements verification method

2.2 Software system modeling based on FSM

Finite state machine [11] is a mathematical model which denotes the finite states and transitions between states of system. FSM consists of five-element set: $FSM=(Q, \Sigma, \delta, q_0, F)$, where: Q is a set of all possible states of system; Σ is a set of triggering events that the system may encounter; δ is state transition functions which determine subsequent state that the system will be transferred to when triggering event occurs. $q_0: Q \times \Sigma \rightarrow Q$; $q_0 \in Q$, where: q_0 is system initial state; $F \subseteq Q$ is a set of system end states, the system will end its behavior once it is transferred to any end state. Otherwise the system will continue to run.




Software system modeling is made based on FSM. According to the relevant documents such as software requirement specifications, software system states and the transition relationships between

states are extracted to build the five-elements set of FSM which is a formal system model of the target software system.

2.3 Combining of CTL and FTA for software safety properties modeling

The system safety properties are obtained by a combined method of FTA and CTL of model checking. First, the software system is analyzed, and the hazards the software system does not want to happen are as the top event, followed by constructing a fault tree which will be used for safety analysis. Hence CTL temporal logic gates are introduced into the fault tree to construct the CTLFT (Computational Tree Logic Fault Tree). During the process, all the events and logic gates of traditional fault tree are remained. In the meantime, the temporal logic gates such as “G”, “F”, “U” which are corresponded to CTL temporal conjunction are introduced to specify description of the fault tree. The temporal logic gates and their meanings are shown in table 1. Second, the minimum cut sets of CTLFT are obtained by adopting qualitative analysis method of FTA. Finally, the formal analysis is conducted by describing the minimum cut sets in CTL logic language. Top event of system fault tree will never happen in any condition on all paths. Thus software safety requirement specification (SRS) is formalized with CTL expression as follows, $SRS = AG \neg TopEvent$.

Table 1 the temporal logic gates and theirs meanings

CTL temporal conjunction	CTLFT temporal logic gate	Meanings	
G		“G” gate	“G” (Global) indicates “all future states”. If all the input fault events under “G” gate are true in the whole system running cycle, then the output event happens.
F		“F” gate	“F” (Future) indicates “some future states”. If the input fault events under “F” gate are true at a certain moment in the whole system running cycle, then the output event happens.
U		“U” gate	“U” (Until) indicates “Until”. “U” gate has two input events (A and B) and one output event (C), if the condition “AUB” is established, then event C happens.

2.4 Safety verification by model checker NuSMV

SMV is a representative model checker for finite state systems against specifications written in temporal logic such as computation tree logic (CTL). It has its own language to describe system model and specifications. The input and the property that is being tested are then converted to the internal representation of SMV. The representations are passed to the model checking algorithm. The result is either a claim that the property is true or else a counterexample showing that the property is false. The result can be analyzed by the software engineer to refine the model of the specification, the property, or even the specification itself.

The above software system and properties models are described with input language of model checker NuSMV [12][13]. A state transition diagram is regarded as a module, each state is declared with a keyword VAR, the transition relations are indicated with keyword ASSIGN, the properties input of model checker is described with a keyword SPEC, the .smv program obtained from above steps is regarded as model checker’s input. The system model is to be modified if the results contain counterexamples, which the system model does not meet the safety.

Method Application

Communication interconnection protocol is used broadly on Internet. However, in the process of protocol connections, message transmission may make a series of mistakes or unsafe issues which will cause catastrophic accidents in some fields. For instance, train collision accident is caused by malfunction of communication protocol, that is, the staff on the ground lost contact with the train control system. Alternating bit protocol (ABP) [14][15] is a simple network protocol operating at the data link layer that retransmits lost or corrupted messages. Hence we choose ABP messaging

process as an example to study the formal verification process of model checking.

3.1 Constructing system model

The system FSM model is stratified into the sender FSM model and receiver FSM model. The two model diagrams are shown as follows in Figure 3 (a) and (b).

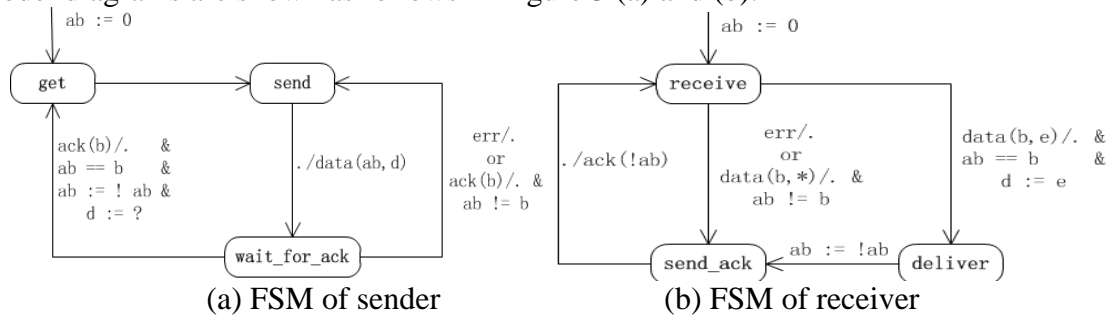


Fig.3. FSM of ABP

After the FSM model and formal description of the ABP instance are obtained, safety properties are established in order to achieve the safety, effectiveness of the protocol.

3.2 Achieving safety properties model

1) Construction of a sequential fault tree of ABP

The main steps of setting up the ABP sequential fault tree are as follows:

Step 1 Obtain the top event. “Data sent from the sender to the receiver failure” is regarded as the top event.

Step 2 Analyze the cause of top event step by step and connect each event with temporal logic gates, and determine the basic gates.

a) Event “Data sent from the sender does not reach the receiver”, event “Order error of data sent from the sender to the receiver” and event “Protocol is in deadlock” which cause the top event are connected with top event by an OR-gate.

b) Analyze the event “Data sent from the sender does not reach the receiver”, make event “The sender sends data error”, event “The receiver receives data error” and event “The sender can’t send data” to be its sub-events and FTA’s basic events. Make event “Data sent from the sender is not the same as the receiver received” to be the sub-event of “Order error of data sent from the sender to the receiver”. Analyze the event “protocol is in deadlock” and make it to be a basic event of FTA because it can directly be formalized description.

Step 3 Analyze all the paths which are from an intermediate event to a basic event and insert related temporal logic gates when necessary. For example, an “A” gate is inserted on this path which is from event “Data sent from the sender does not reach the receiver” to event “The sender sends data error”, because event “The sender sends data error” occurs on all information transmission channels. Thus, the sequential fault tree is obtained which attributes to facilitating conversion from sequential fault tree to CTL formalization. Therefore the ABP sequential fault tree is built and shown in Figure 4 as follows.

Step 4 The minimum cut sets of sequential fault tree are obtained and described them with CTL temporal logic expressions which are safety properties.

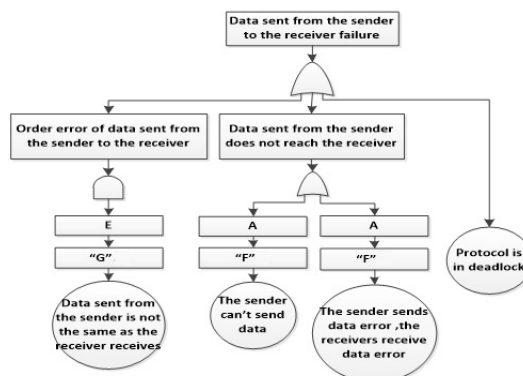


Fig.4. ABP sequential fault tree

2) Safety properties specification

The obtained minimum cut sets of the sequential fault tree are defined as: {Data sent from the sender to the receiver is not the same as the receiver receives}, {The sender can't send data}, {Protocol is in deadlock}, {The sender sends data error, the receiver receives data error}, the safety properties specifications are as follows:

```
!EF(AG(receiver.state = receive -> s2r_in.data != s2r_out.data))
```

```
!EF(AF(sender.state = send & !(s2r_in.tag = mt)))
```

```
AG(AF((sender.state=get->sender.state=send->sender.state=wait_for_ack)&(receiver.state = receive -> receiver.state = deliver -> receiver.state = send_ack))
```

```
!EF(AF(sender.state = send & s2r_in.tag = error)|(receiver.state = receive & s2r_out.tag = error))
```

3.3 Properties verification and the results analysis

After the formal expression of FSM model and safety properties specification which can be accepted by the model checker NuSMV are obtained, model checker is run and the simulation results are shown in Figure 5.

```
The computation of reachable states has been completed.
The diameter of the FSM is 19.
-- specification AG (AF ((sender.state = get -> (sender.state = send -> sender.s
tate = wait_for_ack)) & (receiver.state = receive -> (receiver.state = deliver -
> receiver.state = send_ack)))) is true
-- specification !(EF (AF (sender.state = send & !(s2r_in.tag = mt)))) is true
-- specification !(EF (AG (receiver.state = receive & !(receiver.abp = TRUE & s
2r_out.tag = data0) ! (receiver.abp = FALSE & s2r_out.tag = data1)))) is true
-- specification !(EF (AF (sender.state = send & s2r_in.tag = error) ! (receiver
.state = receive & s2r_out.tag = error))) is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 1.1 <-
```

Fig.5. Verification results

Both true and false verification results are presented in the above figure. The safety property of false verification result corresponds to the minimum cut set {The sender sends data error, the receiver receives data error}. The false result shows that the FMS model does not meet the safety requirement since the top event of sequential fault tree will happen. The counterexample paths can also be obtained in the results.

Summary and Conclusion

In this paper, a combined method with FTA and CTL of model checking is proposed to verify software safety requirement, focusing on solving the problem of the extraction of software safety properties. The fault tree is improved by adding the CTL temporal logic gates, and sequence fault tree CTLFT is constructed by taking the risk or failure as top event. The qualitative analysis method is applied to obtain the minimum cut sets, followed by the conversion from the minimum cut sets to software safety properties. The software safety properties obtained by the proposed extraction method are more reasonable and complete than only by experience and thus the credibility of software safety verification results can be improved effectively.

References

- [1] Leveson, N.G., Software safety: why, what, and how. ACM Comput. Surv., 1986. 18(2): p. 125-163.
- [2] Meenakshi, B., et al. Formal safety analysis of mode transitions in aircraft flight control system. in Digital Avionics Systems Conference, 2007. DASC'07. IEEE/AIAA 26th. 2007: IEEE.
- [3] Clarke, E.M., O. Grumberg and D.A. Peled, Model checking. 1999: MIT press.
- [4] Clarke, E.M. and S. Berezin, Model checking: Historical perspective and example. 1998, Springer. p. 18-24.
- [5] Clarke, E., et al. Symbolic model checking. 1996: Springer.

- [6] Berard, B., et al., Systems and Software Verification: Model-Checking Techniques and Tools. 2010: Springer Publishing Company, Incorporated. 196.
- [7] Norman, G., et al., Model checking probabilistic and stochastic extensions of the π -calculus. Software Engineering, IEEE Transactions on, 2009. 35(2): p. 209-223.
- [8] Batsayan, D., D. Sarkar and S. Chattopadhyay. Model checking on state transition diagram. in Design Automation Conference, 2004. Proceedings of the ASP-DAC 2004. Asia and South Pacific. 2004.
- [9] Younghee, L., et al. A verification of fault tree for safety integrity level evaluation. in ICCAS-SICE, 2009. 2009.
- [10] Shu-Yuan, S., L. Li-Jun and X. Shi-Jian. Study and Application of FTA Software System. in Machine Learning and Cybernetics, 2007 International Conference on. 2007.
- [11] Chenthati, D., et al. Verification of Web Services Modeled as Finite State Machines. in Mathematical/Analytical Modelling and Computer Simulation (AMS), 2010 Fourth Asia International Conference on. 2010.
- [12] NuSMV 2.5 User Manual.
- [13] NuSMV 2.5 Tutorial.
- [14] http://en.wikipedia.org/wiki/Alternating_bit_protocol
- [15] <http://www.macs.hw.ac.uk/~pjbk/nets/protocolsimulations/abp.html>