Research on Formal Modeling and Verification of on-board ATP System

Caiyun Chen[1,a], Qing Luo[1,a] ,Fang Zhang[1,a], Daqing Wang[2,a], Xiaoping Xue[1,a]

[1]School of Electronic and Information Engineering, Tongji University, Shanghai, China

[2]Technology Center, Shanghai Shentong Metro Group Co., Ltd., Shanghai, China

[a]luoqing1962@sohu.com

Keywords: Formal verification; SAT problem; Safety property; SCADE

Abstract. Formal software safety verification is an important issue for on-board ATP (Automation Train Protection) system. A SCADE-based model safety formal verification method is designed in this paper. The extracted safety properties of ATP are expressed by formal automaton machine, which is an unambiguous semantics of the formal method ensuring model-based formal verification mechanisms for system safety. Furthermore, the on-board ATP system and the safety properties module are modeled in SCADE suite, and the safety verification by combination of the two models is done in the Design Verifier using SAT-based Bounded model-checking. The advantages of this method are of completeness and can reduce verification costs.

Introduction

As a safety-critical system, ATP (Automatic Train Protection) protects the train from being in an unsafe state that is one of the core technologies of railway transit. According to the standard EN50128[1], the SIL (Safety Integrity Level) of ATP system is SIL4, which puts forward critically requirements on systems. The conventional developing mode has not adapted to the design procedure of those systems.

In this paper, we focus on the software formal method, which includes formal specification and formal verification. Formalization uses mathematical or logical expression to present the specification and properties of the system and proves the system satisfying the expected properties on the basis of mathematics reasoning. Generally, formal verification methods can be divided into model checking, theorem proving and equivalence checking. With the expanding of scale and function in the software system, the safety verification of the system is becoming more and more difficult. Formal verification is easy to find the designing defects and reveal inconsistency, ambiguity and incompleteness of system. Currently, model-based development of safety-critical software becomes increasingly used, thanks to the application of tools like Matlab/Simulink, UML[2], NuSMV[3] and SCADE[4].

Most of model-checking methods have state-explosion problem. In this paper, we use the Design Verifier, which is built in SCADE and uses SAT-based Bounded model checking, to implement the safety verification. Firstly, the on-board ATP system and the extracted safety properties module are modeled in the graphical editor of SCADE, and then the safety verification is implemented by combination of the two models. If on-board ATP system does not satisfy this safety property, the Design Verifier will give a counter-example. On the contrary, the system meets this expected property. The advantages of this method are the completeness and can reduce the cost of verification.

The remainder of the paper is organized as follows. Section 2 briefly discusses some related works. ATP speed-location function and safety properties are presented in Section 3. Section 4 introduces the process of formal software safety verification in SCADE. Finally, conclusions are given in Section 5.

Related Works

Model checking is an important formal verification method, which mainly creates a FMS (Finite State Machine) for a system or a model in terms of M and formal specification for the expected properties in terms of φ. Model checking judges whether M meets φ by the automatic operation.

Clarke et al.[5] firstly proposed model-checking method that algorithm automatically traversed the state spaces and the φ was satisfied in one of the state spaces. Because the system is abstracted as a finite automaton, it can guarantee convergence in theory when ends the state space. Model checking has the advantage of fully automatic without human interaction. When concluding the system doesn't satisfy the property, model checking will provide a counter-example to locate the error in the designed system. With powerful ability of description of temporal logic, model checking can verify the different kinds of complex time-series properties. The original model-checking algorithms almost verified the property by explicitly enumeration the reachable states of the state space, which usually could only handle a few hundred million numbers of states. It was difficult to overcome the contradiction between exponentially increasing number of states and the limited processing power of the explicit model checking.

Because of the initial model checking having state- explosion problem, there is an important challenge to design a good algorithm and data structure to search large state spaces as far as possible, so that can make the size of the verification range larger. In 1986, Bryant improved the traditional BDD (Binary Decision Diagram) technology and made it into a standard representation of the Boolean expression. McMillan et al.[6] had been a huge success to apply this technique to model-checking and restored the development of model checking again. SMC (Symbolic Model-checking) uses the OBDD (Ordered-BDD) to implicitly express the transition between the states of the finite automaton. Its advantage is to perform the transition relation between a group of states rather than a single, so it greatly alleviates state space explosion to enlarge the scale of system. Reference [7] presents state-space-partitioning-based approaches in SMC, which relieve the state-explosion relative to the fixed-point calculation. There is a great improvement in the size of the validation problem relative to the previous generation of non-OBDD technology model checker, however, it is still restricted in practical verification and the biggest problem is state-explosion.

BMC (Bounded Model-checking) is another important breakthrough after the model checker SMV, which was proposed by Biere et al.[8]. Nocco and Quer[9] used some features of SAT instances generated by BMC, such as the order of the variables and conflict clause, to optimize the SAT tools. Gupta et al.[10] had success in using BDD model checking to make SAT tool automatic obtain some of the characteristics of SAT instances. BMC transforms the states, transition relation and verified formula into SAT instance, through the SAT solver to solve, not like SMV by using OBDD to present states and transition and using fixed-point iterative calculation to solve problem. Therefore, BMC transforms solving the model-checking problem into solving the SAT. SAT problem has been proved to be NP-complete problem, and the upper bound of SAT problem solving is still the exponent in a worst-case scenario. However, SAT solver succeeds greatly in practical application. The serious of state space explosion problem in SAT solving is less than BDD. Intel's research report showed that BMC had an advantage in the capabilities and efficiency of verification than OBDD-based model checking.

The paper implements the formal verification of model-based ATP speed-location function, by using Design Verifier built-in SCADE development environment. Design verifier is based on SAT-based BMC method.

Modeling of Speed-location Function of ATP System

This sector, we will describe the principle of speed and location function of on-board ATP and its safety property, and explain how to model the system in SCADE, including requirements analysis and modeling. Our development of ATP system is based on the development process recommended by EN50128 standards, which forms the V&V development process [1].

Safety Properties of System**.** We use safety automaton to describe the safety properties of ATP speed-location function. It is defined by

$$M = (Q, \Sigma, \Gamma, T, q_0, S)\,, \tag{1}$$

where Q is the set of internal states, Σ is the set of input information, Γ is the set of output information, T is transition function, q_0 is the initial state and S is the set of safety states.

In this definition, transition function T represents as

$$T: Q\times\Sigma \to Q\times\Gamma\,, \tag{2}$$

Generally, T is the partial function on $Q\times\Sigma$, which is interpreted by safety automaton operator. Transition function T includes the state of current model and the current input information, the result is a new state of model and new output information. Fig. 1 intuitively shows the formal safety property automaton.

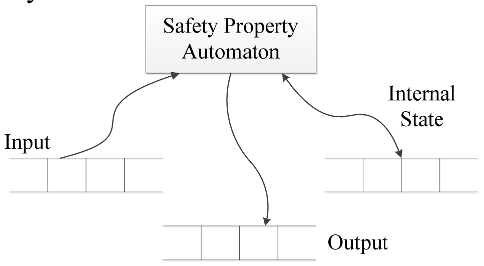


Fig. 1 Formal safety property automaton

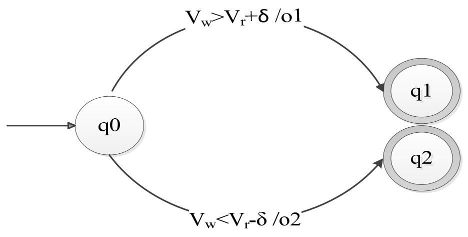


Fig. 2 Automaton of safety property

The system must satisfy function requirements and safety requirements meanwhile. The verification of safety properties is to ensure that the design of the system has achieved the required safety requirements. Safety requirements are essential for the system away from risks, which are extracted in the early phase of the analysis of system risk.

To verify the system designed by formal method, we must define the safety properties formally and extract the safety properties. In this paper, the internal states and input/output information of the on-board ATP speed and location system are defined as shown in Table 1.

According to Eq. 1, $a_i \in \Sigma$, $o_i \in \Gamma$, $q_i \in Q$. The initial state is the normal state.

Table 1 Symbol description

NO.	Symbol	Description
1	a_1	Current speed of wheel sensor
2	a_2	Current speed of radar device
3	o_1	Current go distance of Metro
4	o_2	Current go speed of Metro
5	q_0	Initial state
6	q_1	State of spin
7	q_2	State of slide

The initial state is the normal state.

It is necessary to determine whether the wheel speed is in the error range of radar speed when the Metro is in a state of normal. Metro is in a state of spin if Vw>Vr+ δ that should rectify the going speed and going distance, i.e. using the Vr as the going speed. On the other hand, Metro is in a state of slide if Vw<Vr- δ that should rectify the going speed and going distance, i.e. using the Vr as the going speed. Vw and Vr refer to the speed value measured by wheel sensor and radar device respectively, and δ is the error of radar device.

Transition formulas are $T(q0, a1)=(q1, o1)$, if Vw>Vr+δ, and $T(q0, a1)=(q2, o2)$, if Vw<Vr-δ. Fig. 2 illustrates automaton of that properties.

Modeling of Speed and Location Function**.** Speed-location function used to provide a train's speed and position information is one of the important equipment in on-board ATP. The function measures speed by a compound mode of wheel speed sensor and radar speed device. The key of this method is to judge whether there is happened slide or slip, by a compound mode of wheel speed sensor and radar speed device, in addition adjust the speed and ranging of train according the speeding values. Train revises the slide or slip by using the speed measured by radar.

We transform the above safety property into SCADE model, and analyze requirements of speed-location function firstly.

1) Input. Wheel_Speed refers to the value measured by wheel sensor, and Radar_Speed refers to the value measured by radar device.

2) Output. Distance refers to the value of Metro going; Speed refers to the value of Metro's current speed, and Out_State (enumerated type) means the state of Metro, including State_Normal, State_Spin and Stat_Slide.

3) When the Metro is in State_Normal, if $Vw>Vr+\delta$, Metro turns to be in State_Spin that should rectify the going speed and going distance.

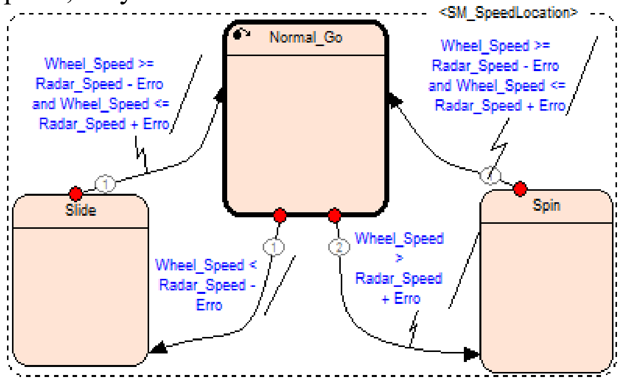
4) When the Metro is in State_Spin, if $Vw>Vr-\delta$ and $Vw<Vr+\delta$, Metro regains to be in State_Normal

5) When the Metro is in State_Normal, if $Vw<Vr-\delta$, Metro turns to be in State_Slide that should rectify the going speed and going distance.

6) When the Metro is in State_Slide, if $Vw>Vr-\delta$ and $Vw<Vr+\delta$, Metro regains to be in State_Normal.

SCADE provides graphical editor to build formal models and to specify safety properties. It is based on the concept of synchronous programming, using strict mathematical theory. SCADE offers two sets of mechanisms for graphical modeling, which are data flow and safety state machine. The mechanisms have the strict mathematical semantics, which can guarantee the model be of accuracy, completeness, consistency and unambiguous. So we can create the model and the requirements formally.

According to the above requirements, the speed and location function is modeled in SCADE graphical editor using the safety state machine, as shown in Fig. 3, which conceals specific operation process in the state machine and illustrates the transition relation among these three states. At some point, only one of the three states can be true.



<SM_SpeedLocation>
Normal_Go
Wheel_Speed >= Radar_Speed - Erro and Wheel_Speed <= Radar_Speed + Erro
Wheel_Speed >= Radar_Speed - Erro and Wheel_Speed <= Radar_Speed + Erro
Slide
Spin
Wheel_Speed < Radar_Speed - Erro
Wheel_Speed > Radar_Speed + Erro

Fig. 3 The SCADE mode of speed-location system

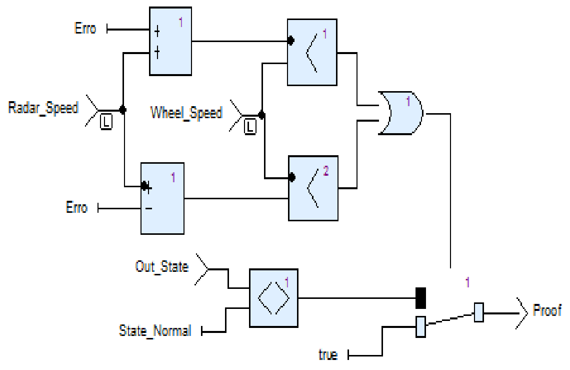


Fig. 4 Safety property observer

In this paper, the safety property of the speed-location function verified is that train will be in State_Spin or in State_Slide, if V_W is not in the range of $[V_r-\delta, V_r+\delta]$.

Formal Verification based on SCADE

In the past, the cost of creating separate analysis model is high and it is difficult to maintain these models. Those are the reasons why formal methods are not widely used. Using the model-based design tools, such as SCADE suite, can effectively eliminate the barriers and provide the basis for the wide application of formal verification. SCADE uses Design Verifier to automatically verify all safety properties. The Design Verifier provides SAT-based BMC method, which solves problem by using SAT solver. The following will introduce the related basis in order to implementing the model checking, such as SAT problem and BMC method.

The SAT problem is to confirm whether there is a group of true assignment (t) that makes the CNF (Conjunctive Normal Form) in Boolean variable set (U) true. If there is an assignment making CNF form true, it indicates that SAT problem is satisfiable.

The SAT Solver is a solver for SAT problem. It is to input a formula into solver and output a solution for the formula. Most SAT solvers are based on Dvis-Putnam- Logemann-Loveland (DPLL) that implements solution through eliminating variables of the CNF formula. DPLL process itself is a big demand for space, and SAT problem is an NP-complete problem, the worst case complexity is O(n2), where n is the number of variables in the formula. Therefore, SAT solver improves the solving efficiency by integrating heuristic local search strategy or Boolean constraint propagation strategy. At present, the GRASP, SATO and zChaff solver are widely used.

BMC method is based on SAT technology, which mainly includes four steps. Firstly, it models the finite automaton M for the system to be verified, simulates the operation of the system by the transit relation between the states of FSM. And it uses CLT (Computation Tree Logic) to represent the specification to be verified. Secondly, sets the upper boundary K. Thirdly, transforms the transit relation between FSM states and negation paradigm of CTL into BMC conversion formula. Lastly, codes the BMC conversion formula into SAT instance, and verifies the formula by using SAT solver. If it has a solution, find out the counter-example. On the contrary, the formula isn't satisfiable and it means the system is safe and error-free when the system runs to the K phase.

The SAT process does not appear the potential state explosion problem as OBDD, which can generate counter-example quicker and needs smaller memory space.

Moreover, it doesn't need to manually define the variable or time-consuming dynamic reordering. Therefore, SAT-based BMC is an important supplement for OBDD-based symbolic model checking.

We design the safety properties and a "property observer" for the created SCADE model when using SCADE to formal verification. Then, the tool can automatically verify the safety property of the model using SAT solver. If the model is safe, it will give a safety proof. Instead, if the model is not safe, it will give a counter-example to help check errors in the system model or safety model. Therefore, to a great extent, formal verification makes sure the safety of the target system.

Formal Verification of Safety Property**.** In this section, we will introduce the process that we verify safety properties of speed-location function based- model using SCADE. The safety property we extract is that train will be in State_Spin or in State_Slide rather than be in State_Normal, if V_w is not in the range of $[V_r-\delta, V_r+\delta]$.

According to this property, we build the verification model called "property observer" in SCADE as shown in Fig. 4.

Firstly, V_w compares with $V_r-\delta$ and $V_r+\delta$ respectively, doing the "Or" operation for the results. Secondly, if the computation is true that indicates V_w not in the range of $[V_r-\delta, V_r+\delta]$, so the Out_State is State_Normal; if false, the Out_State is not State_Normal. Lastly, put the Proof (Boolean type) as analysis object of formal verification, i.e. insert a Proof Objective for the output.

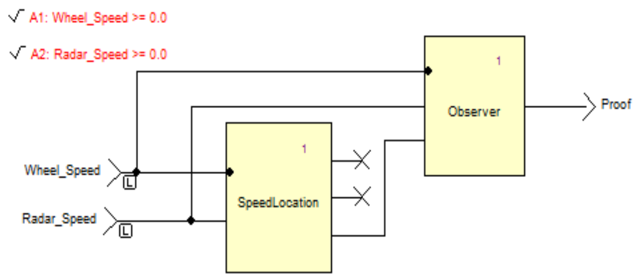
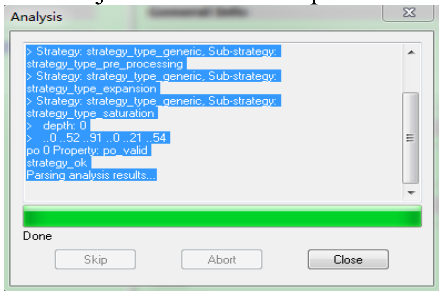


Fig. 5 Safety observer of speed-location system



```
> Strategy: strategy_type_generic, Sub-strategy:
strategy_type_pre_processing
> Strategy: strategy_type_generic, Sub-strategy:
strategy_type_expansion
> Strategy: strategy_type_generic, Sub-strategy:
strategy_type_saturation
>   depth: 0
>   ..0 ..52 ..91 ..0 ..21 ..54
po 0 Property: po_valid
strategy_ok
Parsing analysis results...
```

Fig .6 Verification result of safety property

After safety property observer and verified model (the SCADE speed-location model) are created, we combine the two models to achieve complete formal verification, as shown in Fig. 5 When analyzing the model, SCADE invokes the Prover SLTM engine, automatically to take logical reasoning or exhaustive search method according to the circumstance. Analysis results may be valid or falsifiable. If falsifiable, it gives a counter-example to help analysis.

Fig. 6 shows the result of verification indicating the speed-location function satisfy the safety property.

Conclusions

The paper designs a method to guarantee safety of ATP speed-location function, which focuses on formal modeling and model-based verification. We present a modeling approach to express safety properties by using safety automaton, and extract the main property of the system. What's more, how to model safety property and do formal verification in SCADE is discussed. The main advantage of this method is being of completeness. It allows complete verification of the safety properties, and improves verification efficiency. Error can be detected in the early phase within the design cycle, it reduces design time and verification costs. Another advantage is that this approach can effectively relieve the state-explosion problem.

References

[1] BS EN 50128: 2011, Railway applications communications, signaling and processing systems software for railway control and protection systems, European Committee for Electrotechnical Standardization, June 2011.

[2] C. H. Huang, P. A. Hsiung, Model-Based verification and estimation framework for dynamically partially reconfigurable system, IEEE Transaction on Industrial Informatics, Vol. 7, No. 2, 2011, pp. 287-300.

[3] A. Ferrante et al., A NuSMV extension for graded-CTL model checking, Computer Aided Verification 2010, Vol. 6174 of LNCS, 2010, pp. 670-673.

[4] P. A. Abdulla et al, Designing safe, reliable systems using SCADE, In Proceedings of ISOLA 2004, Vol. 4313 of LNCS, 2006, pp. 115-129.

[5] E. Clarke, O. Grumberg and D. Peled, Model Checking, The MIT Press, Cambridge, 1999, pp. 61−87.

[6] K. Mcmillan, Symbolic Model checking-an approach to the state explosion problem, Ph.D. Dissertation, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. 1992.

[7] S. Iyer, D. Sahoo, E. A. Emerson and J. Jain, On partitioning and symbolic model checking, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 25, No. 5, 2006, pp. 780–785.

[8] A. Biere, A. Cimatti, E. Clarke and Y. Zhu, Symbolic model checking without BDDs, In Tools and Algorithms for the Constructions and Analysis of Systems, Vol. 1579 of LNCS, 1999, pp. 193-207.

[9] S. Nocco, S. Quer, A novel SAT-based approach to the task graph cost-optimal scheduling problem, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 29, No. 12, 2010, pp. 2027-2039.

[10] A. Gupta et al., Learning from BDDs in SAT-based bounded model checking, Proceedings of the 40th Conference on Design Automation, Anaheim, 2-6 June 2003, pp. 824-829.

Acknowledgment

This work is supported by the Open foundation of the State Key Laboratory of Rail Traffic Control and Safety(Contract No. RCS2010K007), Beijing Jiaotong University.