The Verification of Temporary Speed Restriction of Train Control System Based on Timed Automata*

Lei Yuan[1], Junfeng Wang[1] and Renwei Kang[1]
[1]State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, Beijing, China
{LYUAN & JFWANG & 10120291}@bjtu.edu.cn

Abstract - Temporary speed restriction (TSR) refers to the speed restriction within given time bounds beyond the required speed restriction by the lines. TSR system is a typical real-time system. There are strict logical order relations and precise time constraints for the settings, execution, confirmation and cancel of TSR command. So it is important to ensure real-time performance of TSR system. In this paper we focused on the verification of the existing specification of the TSR system of Chinese Train Control System (CTCS) which was used in high-speed railway. We expected to find some imperfections of the working process or the configuration parameters of TSR system. Therefore, a model of the timed automata for the working process of the TSR system was established on the basis of the specification which was published by the administrator of the railway. UPPAAL verification tool was applied to verify the safety and bounded liveness properties of TSR system. The result of the verification showed that some time-related configuration parameters cannot achieve real-time requirements in the existing technical specifications. We also recommended how to correct this imperfection. It showed that the modeling and verification methods we proposed can be useful to study real-time performance of train control system of high-speed railway.

Index Terms - Train Control System, TSRS, Timed Automata, UPPAAL.

1. Introduction

Temporary speed restriction refers to the speed restriction within given time bounds beyond the required speed restriction by the lines. It includes planning speed restriction caused by construction or maintenance and outbreak speed restriction caused by disasters or failure of equipments, etc. Chinese Train Control System (CTCS) which was equipped in Chinese high-speed railway include two levels, L2 and L3. In CTCS-L3, the onboard equipment normally receives the control commands, including temporary speed restriction, via wireless network from the wayside equipment, Radio Block Centre (RBC). But when the wireless network or RBC failed, the onboard equipment will switch to the backup system and will receive the control information from track circuits and balises. So the temporary speed restriction system was configured in CTCS L3 to manage, store, maintain the temporary speed restriction command [1] and the temporary speed restriction working process was specified clearly by the administrator of the railway to ensure that the TSR can be executed correctly through the two ways.

Temporary speed restriction system is a typical real-time system. Once TSR command is issued, the feedback of the implementation results must be receipt within the specified time by the TSR system. Otherwise, it would endanger the railway system. Therefore, it is crucial to ensure real-time performance of TSR system. Up until now, the research on temporary speed restriction mainly focused on the introduction of the application status [1] and the analysis of the effect on train running [2], etc. There is still no research on the verification of the real-time performance of the working process of TSR.

Formal methods which are based on mathematics are often used for modeling and verification of specifications [3], such as the timed automata [4][5]. Timed automata were introduced by R. Alur and D. Dill as an operational model of real-time systems [4]. The real-time system can be described effectively using the timed automata and the function and the properties of system can be verified automatically using UPPAAL[3][4]. UPPAAL, based on the theory of timed automata, is a tool for modeling, simulating, and verifying real-time systems. The name is an acronym for the universities of Uppsala, Sweden and Aalborg, Denmark [4]. There have been relevant cases in the domain of train control system. For example, researchers built the timed automata model to study the safety of train control system [6] and verified the safety and bounded liveness properties of train control system using UPPAAL tool [3]. In this paper, the timed automata network model for the working process of TSR was built and the verification of the model was made in UPPAAL tool to verify the correctness of the specification of the system.

2. Timed Automata in UPPAAL

A timed automata (TA) is a structure TA= (L,A,C,I,E,l0) [4][5]. L is a finite set of locations. A is a set of actions. C is a finite set of clocks. $I{:}L\to\Phi(C)$ is a mapping that assigns to each location a clock constraint, its invariant. $E \subseteq L\times A\times\Phi(C)\times 2^C\times L$ is a the set of directed edges. An element $(l,\alpha,\varphi,Y,l')\in E$ describes an edge from location l to location l' labeled with the action α, the guard φ, and the set Y of clocks that will be reset. $l_0\in L$ is the initial location.

Timed automata engage in transitions from locations to locations when certain timing conditions are satisfied. These transitions either perform input and output actions on channels that will synchronize with other timed automata

* This work is partially supported by the 863Program of China No. 2012AA112801 and the Project of the Fundamental Research Funds for Central Universities No. 2011JB2004 and the Projects of State Key Laboratory of Rail Traffic Control and Safety No.RCS2011ZZ001.

working in paralleled or perform internal actions that are invisible from the outside [4].

TSR system consists of a number of components that work in parallel but also interact with each other from time to time. To model such system, networks of timed automata is built up from single timed automata by two composition operators: parallel composition and restriction. The parallel composition $TA_1||TA_2$ of two timed automata $TA_i = (L_i, A_i, C_i, I_i, E_i, l_{0,i})$ where i=1,2, with disjoint sets of clock C1 and C2 yields the timed automaton [4]

$$TA_1 \,||\, TA_2 = (L_1 \times L_2, A_1 \vee A_2, C_1 \vee C_2, I, E, (l_{0,1}, l_{0,2}))$$

The model-checker tool, UPPAAL, is based on the theory of timed automata and its modeling language offers additional features such as bounded integer variables and urgency. The query language of UPPAAL, used to specify properties to be checked, is a subset of the Timed Computation Tree Logic(TCTL) [4].

3. Modeling and Verification Using UPPAAL

To model the working process of TSR system, we describe the structure and functions of TSR system first. Then we show the models of the TSR system and explain how the model was built. Finally, we discuss how to verify the model and what the result of the verification means.

3.1 The Structure and Functions of TSR System

The components of the TSR system and the transmission channels of the TSR command in the high-speed railway train control system can be seen in literature[7]. Temporary Speed Restriction Server (TSRS), being set near Centralized Traffic Control (CTC) in the operation center, transfer TSR information to Train Control Center (TCC) and RBC. When RBC received TSR command, it will send the TSR command to the trains which approach the TSR area via wireless network, GSM for Railway (GSM-R). When TCC received TSR command, it will write these messages into the related balises through Line side Electronic Unit (LEU). The train which passed that balise can receive the TSR information via balise. The TSRS must ensure the conformity of the TSR information in RBC and in TCC.

In CTCS L3, the formulation, setting and cancel of TSR command are achieved through CTC, TSRS, RBC and TCC together. CTC generate the TSR command according to the input of the dispatcher and send this command to TSRS. TSRS is responsible for the validation and the transmission of the TSR command. RBC and TCC mainly accomplish checking the validity of the TSR command, generating the TSR data packet and returning the execution results of the TSR command to TSRS.

3.2 Working Process of TSR

TSR working process is divided into three parts: the formulation, the setting and the cancel of TSR. Detailed process is seen in reference [7]. Here we explain the working process of the setting TSR, as an example, which was described as a sequence diagram of the UML model.

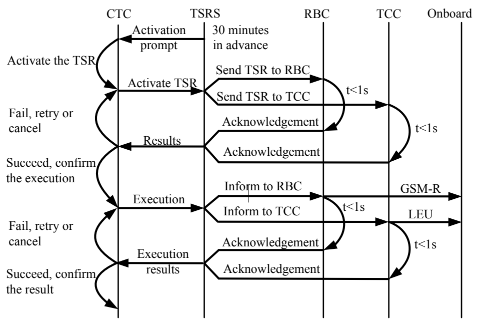

Fig. 1 Working process of the setting of TSR

First, the TSRS will continuously check the stored TSR commands and will prompt the dispatcher to activate the TSR command which are about to be executed. After the activation of the dispatcher, the TSRS will calculate which are the related RBCs and TCCs according to the area of the TSR command and then send the command to them. The related RBCs and TCCs must return the acknowledgement information to TSRS within one second after checking the correctness of the TSR command. The TSRS handles the acknowledgement information. It will send the TSR command back to CTC operating terminal if the acknowledgements from RBCs and TCCs are valid and consistent or send warning information to CTC if the acknowledges are not consistent. The TSRS will inform the related RBCs and TCCs to execute the TSR command when it received the acknowledgement message from the dispatcher. So, the TSR information will be sent to the on-board equipment of the approaching trains by RBC via the GSM-R and will be wrote into the related balises by TCC via LEU. The TSRS will also check the consistency of the acknowledgement message of RBCs and TCCs within one second. If any inconsistency of the execution is found, the TSRS will inform CTC and the dispatcher will retransmit the TSR command or cancel the previous command.

3.3 Model of TSR System

The TSR commands are processed by CTC, TSRS, RBC and TCC together, as shown in Figure 1, and the procedures of CTC, TSRS, RBC and TCC are executed separately. So we first built the sub-models of CTC, TSRS, RBC and TCC one by one, i.e. TA_{CTC}, TA_{TSRS}, TA_{RBC} and TA_{TCC}, and then we expressed the model of the TSR system using the parallel composition of the four models, i.e.

$$TA = TA_{CTC} \,||\, TA_{TSRS} \,||\, TA_{RBC} \,||\, TA_{TCC}$$

The set of clocks of TA_{CTC}, TA_{TSRS}, TA_{RBC} and TA_{TCC} represent the clock variables in the working process of TSR, which are defined as $C=\{T0,T1,T2\}$.

$T0$ which equals 10 minutes represents that the TSRS must prompt CTC to activate the TSR command every 10 minutes in 30 minutes before the planned time of the activation.

$T1$ represents the time interval of TSRS from the execution command was issued to the acknowledgement for this execution command was received by TSRS.

*T*2 represents the time interval of TSRS from the execution command was issued for the first time to the acknowledgement for this TSR command was received by TSRS.

In the TSR technical specifications [7], it was specified that RBC will inform TSRS about the results of the command process after TSRS send TSR command to RBC. Here we explain the semantics of the TA model using this example. Two models were built to describe the behavior of RBC and TSRS as shown in Figure 2.

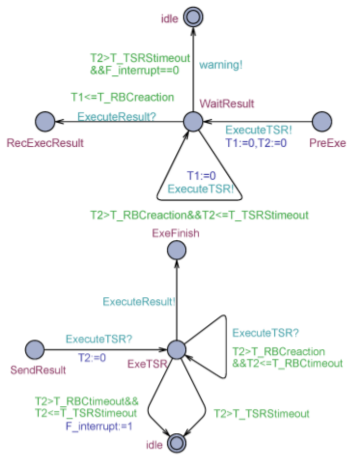

Fig. 2. The sub-model of TSRS and RBC

(1) The location which marks as *PreExe* ($PreExe \in L_{TSRS}$) in the model of TSRS represent the status of TSRS before sending TSR execution command. Also, the location in the model of RBC, *SendResult* ($SendResult \in L_{RBC}$), represent the status before sending the acknowledgement of TSR command.

(2) Note the transition of TSRS model, e_1, and the transition of RBC model, e_2, as e_1=<*S_PreExe,ExecuteTSR, , {T1, T2},S_WaitResult*> and e_2 = <*S_SendResult, ExecuteTSR, , {T2}, S_ExeTSR*>

The transition e_1 will be triggered first. The transition e_1 will generate an action *ExecuteTSR* which means sending TSR command from TSRS to RBC and then the transition e_2 will be triggered synchronously because of the action *ExecuteTSR*. Also, clocks *T1* and *T2* are reset. After the transitions e_1 and e_2, the location transfers to *WaitResult* $\in L_{TSRS}$ and *ExeTSR* $\in L_{RBC}$ respectively.

(3) If RBC reports the execution result of TSR command to TSRS within the time *T_RBCreaction*, the transitions e_3 in RBC model and e_4 in TSRS model will be triggered, which are defined as e_3=<*S_ExeTSR,ExecuteResult, , ,S_ExeFinish*> and e_4=<*S_WaitResult,ExecuteResult,T1<=T_RBCreaction, ,S_RecExecResult*>

(4) If TSRS cannot receive the execution result of TSR command within the time *T_RBCreaction*, TSRS will resend TSR command *ExecuteTSR* within the time *T_TSRStimeout*. Then transitions e_5 in TSRS model and e_6 in RBC model are triggered, which are defined as e_5=<*S_WaitReslut,ExecuteTSR, T2>T_RBCreaction&&T2<=T_TSRStimeout,{T1},S_WaitResult*> and e_6=<*S_ExeTSR,ExecuteTSR,T2>T_RBCreaction && T2 <= T_RBCtimeout, ,S_ExeTSR*>

(5) RBC will consider that the communication channel between RBC and TSRS is already interrupted if RBC does not receive any information from TSRS within the time *T_RBCtimeout*. It can be represented as the transitions e_7 and e_8 in RBC model which are defined as e_7=<*S_ExeTSR, , T2>T_RBCreaction&&T2<=T_TSRStimeout, ,idle*> and e_8= <*S_ExeTSR, ,T2>T_TSRStimeout, idle*>, where $idle \in L^0_{RBC}$.

The transition e_7 is modeled for the situation *T2 > T_RBCtimeout* and *T2 <= T_TSRStimeout*, and the transitions e_8 is for the situation *T2 > T_TSRStimeout*.

(6) If TSRS does not receive any information from RBC within the time *T_TSRStimeout*, TSRS will consider that the communication channel between RBC and TSRS is already interrupted and will send a warning message $warning \in A_1$ to CTC. It can be represented as the transition e_9 and e_{10} in TSRS model which are defined as e_9=<*S_WaitResult,warning, T2>T_TSRStimeout, ,idle*>, where $idle \in L^0_{TSRS}$.

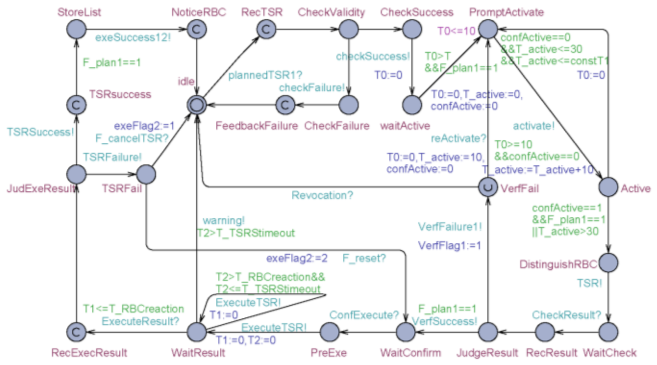

Fig 3. TA model of the working process of TSR

So far, the sub-model of TSRS and RBC are built. Following the above steps, the timed automata model for the working process of TSR system can be modeled. The TSRS model are shown here as an example in Figure 3. The set of edges between locations with an action are determined. The circle marked with 'U' is urgent location in which time is not allowed to pass. The circle marked with 'C', as a committed position, in which time is not allowed to pass and the next transition must involve an outgoing edge of at least one of the committed locations [4].

3.4 Simulation and Verification on the Model of TSR System

The safety property of a model states that *something bad must never happen*. It states what may or may not occur, but do not require that anything ever does happen. The liveness properties of a model state what must occur. The simplest form of a liveness property guarantees that *something good eventually does happen* [4]. The bounded liveness properties of TSR system are guaranteed by location invariant and edges with guard conditions. Safety verification on TSR system can be attributed to reachability analysis of timed automata. TCTL language is applied to verify the safety and bounded liveness properties of system [3].

Taking RBC return execution results to TSRS within the time of T_RBCreaction as example, the verification process are as follows. In order to verify this property, we should find run sequence r_1 and r_2, such that $\left(ExeTSR,\nu\right)\xrightarrow{r_1}\left(ExeFinish,\nu\right)$ and $\left(WaitResult,\nu\right)\xrightarrow{r_2}\left(RecExecResult,\nu\right)$ and $T1 <=$ T_RBCreaction, where $\{ExeTSR,\ ExeFinish\}\subseteq L_{RBC}$ and $\{WaitResult,\ RecExecResult\}\subseteq L_{TSRS}$ and ν is a clock interpretation.

So the verification language of TCTL is that

A<>((RBC.ExeTSR imply RBC.ExeFinish) and (TSRS.WaitResult imply TSRS.RecExecResult) and (TSRS.T1<=T_RBCreaction)).

It means that RBC can send execution feedback to TSRS within the time of T_RBCreaction if the property passed verification. In other words, there exists a run sequence so that the property is satisfied.

Specific verification content and results are shown in Table 1. As we can see in Table 1, the safety and bounded liveness properties of TSR system were satisfied but the deadlock was found in the model. It was found in the simulation that the TSR system made a deadlock during the process of the exchange information between TSRS and RBC. We analyzed the cause of the deadlock is as follow. TSRS judges communication interruption with RBC if it does not receive any information from RBC within the time of T_TSRStimeout, while RBC judges communication interruption with TSRS if it does not receive any information from TSRS within the time of T_RBCtimeout. But according to the configuration of parameters which was defined in the specification of TSR system[7], T_TSRStimeout is less than or equal to 5 seconds and T_RBCtimeout is less than or equal to 3 seconds.

Table 1. Model Verification Content and Verification Results

Serial number		Verification Content	Verification Language	Verification Results
Safety	1	System model without deadlock.	A[]not deadlock	fail
	2	CTC can formulate set and cancel TSR command.	E<>(((CTC.idle)imply(CTC.PreparedTSR))and((CTC.idle)imply(CTC.ExeSuccess)))	pass
	3	Each sub-system check the validity of received TSR information properly.	E<>((TSRS.CheckValidity)or(TSRS.JudgeResult)or(TSRS. JudExeResult) or(RBC.CheckValidity))	pass
	4	TSRS must validate TSR information before the execution.	A[](((TSRS.WaitCheck)imply(TSRS.WaitResult))or(TSRS.CheckValidity imply TSRS.Active))	pass
	5	TSRS sends operating instructions to RBC or TCC properly.	E<>(((TSRS.DistinguishRBC)imply(TSRS.WaitCheck))and((RBC.idle)imply(RBC.CheckValidity)))	pass
bounded liveness properties	6	RBC can return execution results within T_RBCreaction after receiving TSR command.	A<>(((RBC.ExeTSR)imply(RBC.ExeFinish)and((TSRS.WaitResult)imply (TSRS.RecExecResult))and(TSRS.T1<=T_RBCreaction)))	pass
	7	TSRS can resend TSR command if not receiving feedback information within T_TSRStimeout.	A<>((TSRS.WaitResult)imply(TSRS.PreExe)and(T2>T_RBCreaction)and (T2<=T_TSRStimeout)and((RBC.ExeTSR)imply(RBC.SendResult)))	pass

In other words, RBC judges communication interruption from 3 seconds to 5 seconds, but TSRS continues to send information to RBC. At this moment, there have no transitions in the timed automata model. Therefore, the model of system made a deadlock. If the values of the parameters T_TSRStimeout and T_RBCtimeout are equal, the system will avoid deadlock. We found some imperfections in the existing technical specifications of TSR system by this modeling and verification methods proposed, which can optimize the existing technical specifications further.

4. Conclusions

In this paper, we first puts forward a UPPAAL-based modeling and verification methods for the working process of TSR system. Then, with the timed automata theory, the network model for the working process of the TSR in the high-speed railway train control system is established. Third, UPPAAL verification tool successfully verifies the safety and bounded liveness properties of the system. Last but not least, this modeling and verification methods can be useful to study real-time performance of TSR system.

References

[1] GUO Zhen, 2011, "Exploration of Temporary Speed Restriction Technology Application at all Levels in CTCS", Journal of Science & Technology Information, Vol16: 111-112.

[2] HUANG Yuan-yuan, DONG Yu, ZHAO Yu-kun, 2011, "Simulation Study of Temporary Speed Restriction on Train Operation", Journal of Railway Signalling & Communication, Vol1: 13-15.

[3] LU Ji-dong, TANG Tao, JIA Hao, 2010, "Modeling and Verification of Radio Block Center of CTCS-3 Train Control System for Dedicated Passengers Lines",Journal of the China Railway Society,Vol32(6):34-42.

[4] E. -R. Olderog and H.Dierks, 2008, "Real-Time Systems", Cambridge University Press, London, pp.137-146.

[5] Alur R. and D. L. Dill, 2004, "A theory of timed automata", Journal of Theoretical Computer Science, Vol126: 193-194.

[6] ZHOU Qing-lei, JI Li-xia, 2004, "On Automatic Turnout Control Based on Timed Automata", Journal of Control Engineering of China, Vol11 (supplement): 146-147.

[7] Ministry of Railways, 2010, "Technical Specifications of temporary speed restriction in Train Control System for Dedicated Passengers Lines(Version 2.0)".