Simulation of CTCS-3 Protocol with Temporal Logic Programming

Peng Zhang, Zhenhua Duan* and Cong Tian
Institute of Computing Theory & Technology and ISN Laboratory
Xidian University Xi'an 710071, P.R. China
Email: zhangbestpeng@qq.com, zhhduan@mail.xidian.edu.cn, ctian@mail.xidian.edu.cn

***Abstract*—This paper presents an approach to simulate and verify the CTCS-3 (Chinese Train Control System 3) protocol with a Modeling, Simulation and Verification Language (MSVL) which is an executable subset of Projection Temporal Logic (PTL). First, the syntax and semantics of PTL and MSVL are briefly introduced. Then, CTCS-3 protocol is briefly presented and a simplified CTCS-3 protocol described by an MSVL program is given, and the property to be verified is specified by a Propositional Projection Temporal Logic (PPTL) formula. Finally, the simulation is conducted by means of the interpreter of MSVL. A dynamic graph showing behavior of train is depicted. Based on the graph, we can verify whether or not the protocol satisfies the property.**

***Keywords*—Temporal Logic; MSVL; CTCS-3 protocol; Simulation; Verification**

I. Introduction

The CTCS-3 (Chinese Train Control System 3)[1] is a protocol with the speed of 300-350 kilometers per hour. CTCS-3 protocol uses the advanced technology to monitor and control the speed and protect it against over-speed. CTCS-3 protocol must meet the extremely high safety constrains, otherwise errors would cause serious consequences sometimes. On July 23 in 2011, two trains with high speed collided with each other in WenZhou city of ZheJiang province and this accident made 40 people dead and about 200 people injured. The accident report shows that this accident was caused by design faults of system using CTCS-3 protocol. So it is critical to ensure the correctness and reliability of CTCS-3 protocol. For doing so, simulation and testing are the effective means in practice.

Simulation is a process which is established by an actual system model and the model is used to execute experimental instances. Testing is to design a series of test cases to detect whether a system is right or not. There are many simulation and testing tools, such as $Simulink$ [2], [3], [4] and $Modelsim$ [5], [6]. $Simulink$ is a visual simulation tool based on Matlab, and it is a software package for simulation, dynamic system modeling, which has been widely used in linear systems and digital control systems. $Modelsim$ is considered to be the most excellent Hardware Description Language (HDL). It can provide friendly simulation environment and support Graphical User Interface (GUI). Compared to $Simulink$, the simulation speed of $Modelsim$ is much faster. However, the two methods above are not able to simulate and test the system built upon CTCS-3 effectively. Although $Simulink$ can simulate CTCS-3 protocol it cannot verify the property, so it may not find the design faults of the protocol effectively. $Modelsim$ is often used to simulate the circuit systems, so it's not suitable to simulate and verify the real-time system using CTCS-3 protocol. As a result, more and more researchers consider that the formal engineering methods based on mathematical formulations can give a good solution to solve the problem.

Temporal logic (TL) is one of the mathematical foundation of formal engineering methods. TL was proposed for the purpose of specification and verification of concurrent systems, and was widely applied in verifying software engineering and digital circuits. In particular, Projection Temporal Logic (PTL)[7], [8] is an interval-based temporal logic, which is a useful formalism for reasoning about period of time with hardware and software systems. It can handle both sequential and parallel compositions, and offer useful and practical proof techniques for verifying concurrent systems. The Modeling, Simulation, and Verification Language (MSVL)[9], [10] is an executable subset of PTL and can be used to model, simulate and verify concurrent systems. So we are able to use an MSVL program to describe CTCS-3 protocol and specify its properties by PPTL [9], [10], [11] formulas. In this way, we can simulate the CTCS-3 protocol and prove whether or not the protocol satisfies the property according to the model checking approach. If it doesn't satisfy the desired properties, it can detect design errors which could not be found by the simulation and testing. The main contributions of this paper are as follows: a simplified CTCS-3 protocol described by an MSVL program is given, and its properties are specified by PPTL formulas. The simulation result of the model is shown in a dynamic graph, which is used to verify whether it satisfies the properties or not.

The rest of the paper is organized as follows: the syntax and semantics of PTL and MSVL are described in the next section. In section 3, a simplified CTCS-3 protocol described by the MSVL program is realized and the property of the protocol is verified. Then, the MSVL program is illustrated within the interpreter of MSVL and the simulation and verification results are depicted in section 4. In addition, some related work are compared in section 5. Finally, conclusions are given in section 6.

II. Preliminaries

A. Propositional Projection Temporal Logic

Our underlying logic is Projection Temporal Logic. In the following, we briefly introduce its syntax and semantics. The detail can be found in [12].

978-1-4673-6085-2/13/$31.00 ©2013 IEEE

1) Syntax: Let Π be a countable set of propositions, and V be a countable set of typed static and dynamic variables. Terms e and formulas p of the logic are defined as follows:

$$e ::= v \mid \bigcirc e \mid \ominus e \mid \mathit{beg(e)} \mid \mathit{end(e)} \mid f(e_1, \ldots, e_m)$$
$$p ::= \pi \mid e_1 = e_2 \mid P(e_1, \ldots, e_m) \mid \neg p \mid p_1 \wedge p_2 \mid \exists\, v : p \mid$$
$$\bigcirc p \mid (p_1, \ldots, p_m)\ \mathit{prj}\ p \mid (p_1, \ldots, p_m)\ \mathit{prj}\ p$$

where v is a static or dynamic variable, and π is a proposition. In $f(e_1, \ldots, e_m)$ and $P(e_1, \ldots, e_m)$, where f is a function and P is a predicate. It is assumed that the types of the terms are compatible with those of the arguments of f and P. A formula (term) is called a state formula (term) if it does not contain any temporal operators (i.e., $\bigcirc$, $\ominus$ and *prj*), otherwise it is a temporal formula (term).

2) Semantics: A state s is a pair of assignments (I_{var}, I_{prop}) which, for each variable $v \in V$ gives $s[v] = I_{var}[v]$, and for each proposition $\pi \in \Pi$ gives $s[\pi] = I_{prop}[\pi]$. Each $I_{var}[v]$ is a value of the appropriate type or nil (undefined), whereas $I_{prop}[\pi]$ is true or false.

An interval $\sigma = \langle s_0, s_1, \ldots \rangle$ is a non-empty (possibly infinite) sequence of states. It is assumed that each static variable is assigned the same value in all the states over σ. The length of σ, denoted by $|\sigma|$, is defined as ω if σ is infinite; otherwise it is the number of the states in σ minus one. The concatenation of a finite interval $\sigma =< s_0, \ldots, s_{|\sigma|} >$ with another interval $\sigma' =< s'_0, \ldots, s'_{|\sigma'|} >$ (may be infinite) is denoted by $\sigma \cdot \sigma'$ and $\sigma \cdot \sigma' =< s_0, \ldots, s_{|\sigma|}, s'_0, \ldots, s'_{|\sigma'|} >$. To have an uniform notation for both finite and infinite intervals, we will use extended integers as indices. That is, we consider the set $\mathbb{N}_0$ of non-negative integers with added ω, $\mathbb{N}_\omega = \mathbb{N}_0 \cup \{\omega\}$, and extend the standard arithmetic comparison operators ($=$, $<$ and $\leq$) to $\mathbb{N}_\omega$, by setting $\omega = \omega$ and $n < \omega$, for all $n \in \mathbb{N}_0$. Furthermore, we define $\preceq$ as $\leq -\{(\omega, \omega)\}$. To simplify definitions, we will denote σ as $\langle s_0, \ldots, s_{|\sigma|} \rangle$, where $s_{|\sigma|}$ is undefined if σ is infinite. With such a notation, $\sigma_{(i..j)}$ (for $0 \leq i \preceq j \leq |\sigma|$) denotes the sub-interval $\langle s_i, \ldots, s_j \rangle$ and $\sigma^{(k)}$ (for $0 \leq k \preceq |\sigma|$) denotes $\langle s_k, \ldots, s_{|\sigma|} \rangle$.

To define the semantics of the projection operator we need an auxiliary operator. Let $\sigma = \langle s_0, s_1, \ldots \rangle$ be an interval and $r_1, \ldots, r_h$ be integers ($h \geq 1$) such that $0 \leq r_1 \leq r_2 \leq \ldots \leq r_h \leq |\sigma|$. $\sigma{\downarrow}(r_1, \ldots, r_h) = \langle s_{t_1}, s_{t_2} \ldots, s_{t_l} \rangle, (t_1 < t_2 < \ldots < t_l)$. The projection of σ onto $r_1, \ldots, r_h$ is the interval, called projected interval, where $t_1, \ldots, t_l$ are obtained from $r_1, \ldots, r_h$ by deleting all duplicates. In other words, $t_1, \ldots, t_l$ is the longest strictly increasing subsequence of $r_1, \ldots, r_h$. For example $\langle s_0, s_1, s_2, s_3, s_4, s_5 \rangle \downarrow (0, 2, 2, 2, 3, 4, 4, 5) = \langle s_0, s_2, s_3, s_4, s_5 \rangle$.

An interpretation for a PTL term or formula is a tuple $\mathfrak{I} = (\sigma, i, k, j)$, where $\sigma = \langle s_0, s_1, \ldots \rangle$ is an interval and $i, j, k \in \mathbb{N}_\omega$ are integers such that $i \leq k \preceq j \leq |\sigma|$. Intuitively, we use (σ, i, k, j) to mean that a term or formula is interpreted over a subinterval $\sigma_{(i..j)}$ with the current state being s_k. Then, for every term e, the evaluation of e relative to $\mathfrak{I}$, denoted by $\mathfrak{I}[e]$, is defined by induction on terms in the following way:
$\mathfrak{I}[v] = s_k[v] = I^k_{var}[v]$ if v is a variable

$$\mathfrak{I}[f(e_1, \ldots, e_m)] = \begin{cases} \mathfrak{I}[f](\mathfrak{I}[e_1], \ldots, \mathfrak{I}[e_m]) & \text{if } \mathfrak{I}[e_i] \neq nil \\ nil & \text{otherwise} \end{cases}$$

$$\mathfrak{I}[\bigcirc e] = \begin{cases} (\sigma, i, k+1, j)[e] & \text{if } k < j \\ nil & \text{otherwise} \end{cases}$$

$$\mathfrak{I}[\ominus e] = \begin{cases} (\sigma, i, k-1, j)[e] & \text{if } i < k \\ nil & \text{otherwise} \end{cases}$$

$$\mathfrak{I}[\mathsf{beg}(e)] = (\sigma, i, i, j)[e]$$

$$\mathfrak{I}[\mathsf{end}(e)] = \begin{cases} (\sigma, i, j, j)[e] & \text{if } j \neq \omega \\ nil & \text{otherwise} \end{cases}$$

The satisfaction relation for formulas, $\models$, is inductively defined as follows.

1. $\mathfrak{I} \models \pi$ iff $s_k[\pi] = I^k_{prop}[\pi] = \mathsf{true}$.

2. $\mathfrak{I} \models P(e_1, \ldots, e_m)$ iff $\mathfrak{I}[e_i] \neq nil$ and $\mathfrak{I}[P](\mathfrak{I}[e_1], \ldots, \mathfrak{I}[e_m]) = \mathsf{true}$, for all $1 \leq i \leq m$.

3. $\mathfrak{I} \models e_1 = e_2$ iff $\mathfrak{I}[e_1] = \mathfrak{I}[e_2]$.

4. $\mathfrak{I} \models \neg p$ iff $\mathfrak{I} \not\models p$.

5. $\mathfrak{I} \models p \wedge q$ iff $\mathfrak{I} \models p$ and $\mathfrak{I} \models q$.

6. $\mathfrak{I} \models \bigcirc p$ iff $k < j$ and $(\sigma, i, k+1, j) \models p$.

7. $\mathfrak{I} \models \exists v : p$ iff $(\sigma', i, k, j) \models p$ for some $\sigma' \overset{v}{=} \sigma$.

8. $\mathfrak{I} \models (p_1, \ldots, p_m)$ prj p iff there are $k = r_0 \leq r_1 \leq \ldots \leq r_m \leq j.$ such that $(\sigma, i, r_0, r_1) \models p_1$ and $(\sigma, r_{l-1}, r_{l-1}, r_l) \models p_l$ for all $1 < l \leq m$ and $(\sigma', 0, 0, |\sigma'|) \models p$ for σ' given by :
- if $r_m < j$ then $\sigma' = \sigma{\downarrow}(r_0, \ldots, r_m) \cdot \sigma_{(r_m+1..j)}$
- if $r_m = j$ then $\sigma' = \sigma{\downarrow}(r_0, \ldots, r_h)$ for some $0 \leq h \leq m$.

9. $\mathfrak{I} \models (p_1, \ldots, (p_i, \ldots, p_l)^{\oplus}, \ldots, p_m)$ prj p iff one of the following cases holds:
- $1 \leq i \leq l \leq m$ and there exists an integer $n \geq 1$ and $\mathfrak{I} \models (p_1, \ldots, (p_i, \ldots, p_l)^{(n)}, \ldots, p_m)$ prj p, or
- $1 \leq i \leq l = m, j = w$ and there exist infinitely many integers $k = r_0 \leq r_1 \leq \ldots \leq r_k \preceq \omega$ and $\lim_{k \to \infty} r_k = \omega$ such that $(\sigma, i, r_0, r_1) \models p_1$ and $(\sigma, r_{x-1}, r_{x-1}, r_x) \models p_x$ for all $1 \leq x \leq i-1$ and $(\sigma, r_{i+t(l-i+1)+n-1}, r_{i+t(l-i+1)+n-1}, r_{i+t(l-i+1)+n}) \models p_{i+n}$, for all $t \geq 0$ and $0 \leq n \leq l-i$, and $\sigma{\downarrow}(r_0, r_1, \ldots, r_h, \omega) \models p$ for some $h \in N_\omega$.

A formula P is satisfied by an interval σ, denoted by $\sigma \models P$, if $(\sigma, 0, 0, |\sigma|) \models P$. A formula P is called satisfiable if $\sigma \models P$ for some σ. A formula P is valid, denoted by $\models P$, if $\sigma \models P$ for all σ. A formula p is equivalent to another formula q, denoted by $p \equiv q$, if $\models \Box(p \leftrightarrow q)$.

The abbreviations true, false, $\vee$, $\rightarrow$ and $\leftrightarrow$ are defined as usual. In particular, $\mathsf{true} \equiv p \vee \neg p$ and $\mathsf{false} \equiv p \wedge \neg p$ for any formula p. We also use the following abbreviations:

$$\begin{array}{ll}
\mathsf{empty} \overset{\text{def}}{=} \neg \bigcirc \mathsf{true} & \mathsf{skip} \overset{\text{def}}{=} \bigcirc \mathsf{empty} \\
p\,;\,q \overset{\text{def}}{=} (p, q)\ \mathsf{prj\ empty} & \Diamond p \overset{\text{def}}{=} \mathsf{true}\,;\,p \\
\odot\, p \overset{\text{def}}{=} \mathsf{empty} \vee \bigcirc p & \Box p \overset{\text{def}}{=} \neg\Diamond\neg p \\
\mathsf{more} \overset{\text{def}}{=} \neg \mathsf{empty} & x := e \overset{\text{def}}{=} \bigcirc x = e \wedge \mathsf{skip} \\
\mathsf{fin}(p) \overset{\text{def}}{=} \Box(\mathsf{empty} \rightarrow p) & p^{+} \overset{\text{def}}{=} (p^{\oplus})\ \mathsf{prj\ empty} \\
p^{*} \overset{\text{def}}{=} (p^{\circledast})\ \mathsf{prj\ empty} & \mathsf{halt}(p) \overset{\text{def}}{=} \Box(\mathsf{empty} \leftrightarrow p) \\
\mathsf{keep}(p) \overset{\text{def}}{=} \Box(\neg \mathsf{empty} \rightarrow p) & \\
p||q \overset{\text{def}}{=} p \wedge (q\,;\,\mathsf{true}) \vee q \wedge (p\,;\,\mathsf{true}) &
\end{array}$$

A number of logic laws of PTL can be found in [7], [12].

B. Modeling, Simulation and Verification Language

The Modeling, Simulation and Verification Language with frame technique called MSVL is an executable subset of PTL and used to model, simulate and verify a system. In addition, the variables within a program can refer to their previous values.

1) Syntax: As an executable subset of PTL, MSVL consists of expressions and statements. Expressions can be regarded as PTL terms, and statements as PTL formulas. The arithmetic expression e and boolean expression b of MSVL are inductively defined as follows:

$$e ::= n \mid x \mid \bigcirc x \mid \ominus x \mid e_0 \; op \; e_0 \; (op ::= + \mid - \mid * \mid / \mid mod)$$
$$b ::= true \mid false \mid e_0 = e_1 \mid e_0 < e_1 \mid \neg\, b \mid b_1 \wedge b_2$$

where n is an integer and x is a variable. The elementary statements in MSVL are defined as follows:

1. Assignment $x = e$. 2. P-I-Assignment $x \Leftarrow e$.
3. Conjunction $p \wedge q$. 4. Selection $p \vee q$.
5. Next $\bigcirc p$. 6. Always $\Box p$.
7. Termination empty. 8. Sequential $p\,;\,q$.
9. Local variable $\exists x : p$. 10. State Frame $\mathsf{lbf}(x)$.
11. Interval Frame $\mathsf{frame}(x)$.
12. Projection $(p_1, \ldots, p_m)$ prj q.
13. Parallel $p \parallel q \overset{\text{def}}{=} p \wedge (q\,;\,\mathsf{true}) \vee q \wedge (p\,;\,\mathsf{true})$.
14. Conditional $\texttt{if}\ b\ \texttt{then}\ p\ \texttt{else}\ q \overset{\text{def}}{=} (b \rightarrow p) \wedge (\neg b \rightarrow q)$.
15. While $\texttt{while}\ b\ \texttt{do}\ p \overset{\text{def}}{=} (b \wedge p)^* \wedge \Box(\mathsf{empty} \rightarrow \neg b)$.
16. Await $\mathsf{await}(b) \overset{\text{def}}{=} (\mathsf{frame}(x_1) \wedge \cdots \wedge \mathsf{frame}(x_h)) \wedge \Box(\mathsf{empty} \leftrightarrow b)$ where $x_i \in V_b = \{x | x \text{ appears in } b\}$.

where x denotes a variable, e stands for an arbitrary arithmetic expression, b a boolean expression, and $p_1, \ldots, p_m$, p and q stand for programs of MSVL. The assignment $x = e$, $x \Leftarrow e$, empty, $\mathsf{lbf}(x)$, and $\mathsf{frame}(x)$ can be regarded as basic statements, and the others composite ones; $\mathsf{frame}(x)$ means that variable x always keeps its old value over an interval if no assignment to x is encountered.

2) Normal Form: Since temporal logic formulas are interpreted over intervals and temporal logic programming language is a subset of the corresponding temporal logic, programs should be executed over a sequence of states which is a marked feature of temporal logic programming. So execution of programs is implemented by reduction since a program can be reduced to two parts: present and remain. That fact can be illustrated by the normal form of programs given in [13].

Definition 1 (Normal Form of MSVL program): An MSVL program q is in Normal Form if

$$q \overset{\text{def}}{=} \bigvee_{i=1}^{k} q_{ei} \wedge \mathsf{empty} \vee \bigvee_{j=1}^{h} q_{cj} \wedge \bigcirc q_{fj}$$

where $k + h \geq 1$ and the following hold:
• q_{fj} is an internal program in which variables may refer to the previous states but not beyond the first state of the current interval over which the program is executed.
• each q_{ei} and q_{cj} is either true or a state formula of the form $p_1 \wedge \ldots \wedge p_m$ $(m \geq 1)$ and each p_l $(1 \leq l \leq m)$ is either $(x = e)$ with $e \in D$, $x \in V$, or p_x or $\neg p_x$.

Theorem 1: For each MSVL program p there is a program p' in Normal Form satisfying $p \equiv p'$

Theorem 1 tells us that for each MSVL program, there is an equivalent program in normal form. The proof of the theorem can be found in [7], [12].

III. Simulation of CTCS-3 protocol

Learning from the European Train Control System (ETCS)[14], [15] protocol, CTCS protocol has been made. According to functional requirement and device configuration [1], the CTCS protocol is divided into five levels. The CTCS-3 level is an important protocol and it is the core technique to ensure high-speed train to run safely, reliably and efficiently. This paper focuses on the CTCS-3 protocol.

A. Architecture of CTCS-3 Protocol

The architecture of CTCS protocol is divided into four layers as follows: the railway transport management layer, the network transport layer, the ground equipment layer and vehicle equipment layer. The architecture of the four layer is shown in figure 1. The railway transport management layer controls the train by the communication network to ensure the train to run safely; the network transport layer is to transfer the data among the four layers; the ground equipment layer and vehicle equipment layer contain the track circuit and a variety of communication equipments, which are the control center when the train is running.

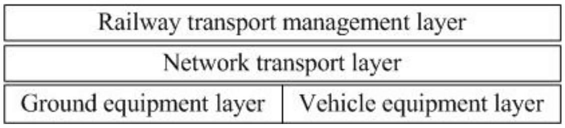

Fig. 1: Architecture of CTCS protocol

B. Data Flow Diagram of CTCS-3 Protocol

Based on the architecture and working processes of CTCS-3 protocol [16], there are four sub-systems as follows: the Centralized Traffic Control System (CTC) in the railway transport management layer, the Radio Block Center (RBC) in the network transport layer, the Railway Interlocking System (RIS) in the ground equipment layer and the Vehicle Control System (VCS) in the vehicle equipment layer. By the collaboration among the four sub-systems, CTCS-3 protocol can operate safely, orderly and reliably.

Of the four sub-systems, RBC is the data processing center of CTCS-3 protocol, almost all the data that other three sub-systems need is transferred by RBC. The Data Flow Diagram (DFD) of CTCS-3 protocol is shown by figure 2.

C. A Simplified CTCS-3 Protocol

When a train begins to run using CTCS-3 protocol, it must be registered in the nearest RBC. Then the track will be divided into lots of intervals. At the same time, all the RBC systems begin to work and each RBC covers several intervals. In these intervals, all the data is transferred by the same RBC. That is to say, the VCS sends data by one RBC in this interval and it will be switched to another RBC in another interval.

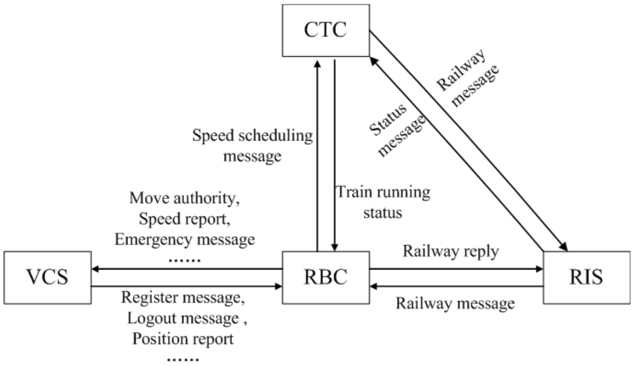

Fig. 2: DFD of CTCS-3 protocol

When the train is running in the intervals, Move Authority (MA) is an important method to ensure the train to run safely. There is no doubt that the train will be safe to move from current interval to the interval which is marked by MA. If it reaches the interval marked by MA, the train must stop no matter whether it arrivals at the station or not. RBC periodically interacts with the VCS, RIS and CTC and MA is determined through the train's current position, driving direction and the states of the track.

In addition, the principle of feedback has introduced to further modify the MA. The feedback signal is used to mark whether the interval is occupied or not. If the interval of the track is occupied by a train or destroyed by some other reasons, the flag of this interval signal is set to 1, otherwise it is set to 0. Then the feedback signal will be sent to RBC and all the trains in the track will share this signal too. According to the feedback signal, the RBC and VCS will adjust the original MA. In this way, CTCS-3 protocol can guarantee several trains to run safely and orderly in the same track, even if a certain interval of the track is occupied or destroyed.

To be described by an MSVL program, the track should also be simplified. A mathematical form of the track is defined as shown in figure 3. It is assumed that each RBC covers two intervals. A binary string is given to represent the interval information of the track and 0 is used to represent the station. As we know, there are three kinds of stations: original station, middle station, terminal station. The string 11, 10, and 01 represent the intervals of the track. Obviously, the string 00 is considered as the wrong interval since if the train runs in the track with the string 00, it will encounter a series of problems and will be forced to stop immediately.

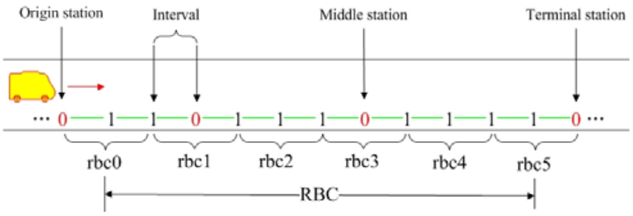

Fig. 3: Mathematical form of track

D. Simulation Process of the Simplified CTCS-3 Protocol

Based on the simplified CTCS-3 protocol, CTCS-3 protocol can be simulated by an MSVL program. The simulation processes are shown in Figure 4 as follows.

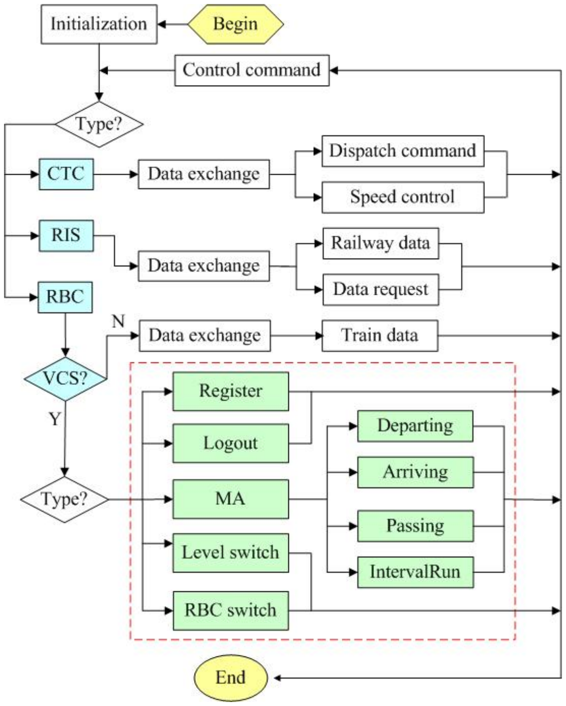

Fig. 4: Simulation process of CTCS-3 protocol

As shown in Figure 4, the simulation processes are as follows: firstly, a series of data should be initialized and the train is in the state of waiting for the command to start. When the control command is received, it needs to determine the type of the command. If the command is related to CTC, it will exchange the data among the other sub-systems, including the dispatch command and speed control command; if it is related to RIS, it will exchange the railway data and send some data request; if the command is related to RBC, it will determine whether the command is related to VCS or not. If it is not related to VCS, it also exchange the data of trains; otherwise, it will enter the module of VCS, which has realized the interaction of the data that the train needs when it is running. In this part, it realizes the register and logout operation of the train, the switching of RBC, the switching of the train level and the obtaining of the MA. The operation of MA includes departing process, arriving process, passing process and the process of interval running. Obviously, the above process is a cyclical process.

IV. Simulation and Verification Results

An interpreter named MSV for Temporal Logic Programming Language MSVL has been developed in the operation system of Linux. The interpreter MSV can be used for simulating a program described by MSVL, verifying the property specified by a PPTL formula.

A. Simulation Results

As it is given in figure 4, the simplified CTCS-3 protocol has been modeled by an MSVL program. In this model, the track is simplified as 0110111011110 based on the mathematical form of figure 3. The track is divided into 12 intervals. When the trains are moving in the track, the simulation graph can show the processes of running and give some running information of each train. There are four conditions of the simulation results, the details of each condition are shown as follows.

The first condition is that one train runs safely in the normal track. The feedback signal which marks whether the interval is occupied or not is initially set to 000000000000. This condition is simple and the train will stop normally in the terminal station. The simulation result is shown in figure 5. Here we only give one of the simulation states.

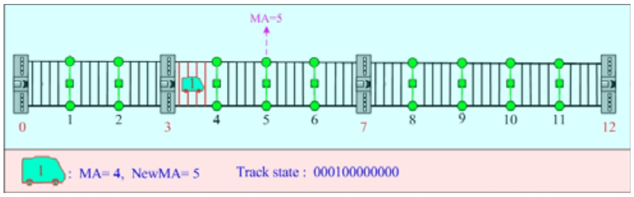

Fig. 5: One train runs in the normal track

The second condition is as follows: when one train is running in the track, one interval of the track is occupied or destroyed, so it must be stopped in this interval. The feedback signal is initially set to 000000000100, which indicates the tenth interval is abnormal. The simulation result can show how the train to stop before the abnormal interval, and figure 6 gives the last state of the train.

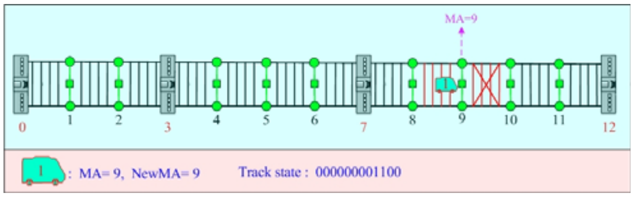

Fig. 6: One train runs in the abnormal track

The third condition is that two trains are running at the same time with one after the other. The track is normal so the feedback signal is also initially set to 000000000000. If the two trains keep a certain distance and the speeds of two trains are appropriate, both of them can run safely and orderly in the track. The following figure 7 shows how two trains move. Here we only give one of the simulation states.

The fourth condition is based on the third condition. The feedback signal is initially set to 000000000010 , which indicates the eleventh interval is abnormal. In this condition, the train ahead must stop in the eleventh interval because this interval is destroyed; then the train behind this train should stop immediately. Otherwise, it will collide the train ahead and cause serious consequences. So figure 8 is the last state of this condition.

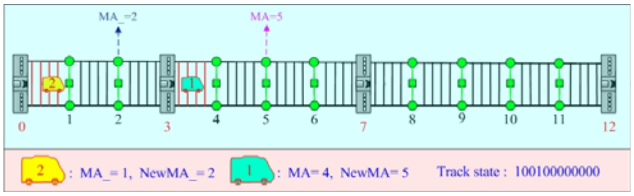

Fig. 7: Two trains run in the normal track

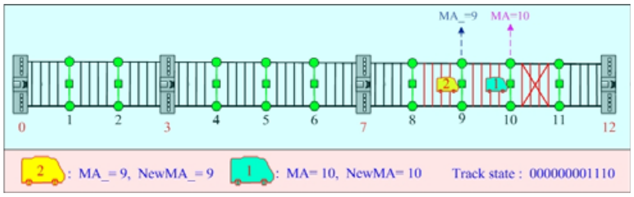

Fig. 8: Two trains run in the abnormal track

B. Verification Results

Besides simulating the model, we can also verify the desired property of CTCS-3 protocol with the interpreter MSV. Here we give the method to verify the property of the third and fourth condition above. When two trains are running at the same time, we expect that both of them must run safely and orderly with no collision in any case. So we can describe this property by a PPTL formula as shown below:

$$\begin{array}{l} </ \\ \quad define\quad check\_MA : MA\_ < MA; \\ \quad always\quad check\_MA \\ /> \end{array}$$

where MA is the move authority of the train ahead and $MA\_$ is that of the train behind it.

This property can detect that the MA of the train ahead is always greater than the train's MA behind it. In each state, if this property is satisfied, the two trains cannot collide each other. Figure 9 is the verification result of a simulation state when two trains run in the normal track. Figure 10 is the verification result of the last simulation state when two trains run in the abnormal track. It will verify every state to detect whether the protocol satisfies this property or not. The verification results of other states are omitted here. According to the verification results, we can easily verify whether or not the CTCS-3 protocol satisfies this property.

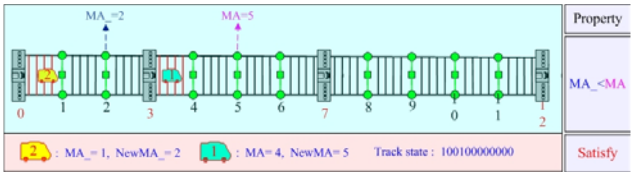

Fig. 9: Verification result of two trains running in the normal track

V. Related Work

Many researchers and organizations have explored how to simulate and verify the protocol of TCS (Train Control System)

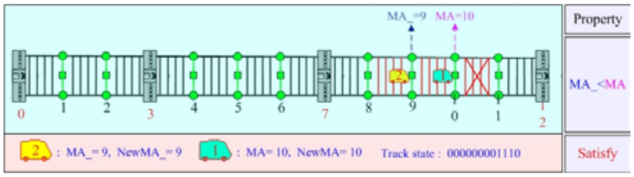

Fig. 10: Verification result of two trains running in the abnormal track

efficiently and some methods have been proposed to solve the problem. Since 1990's, OOM (Object-Oriented Method) is widely used to simulate the protocol of TCS [17]. The researchers in the National University of Singapore realized the simulation of the protocol using OOM. In 2004, the University of Florence in Italy developed a testing system of ATP/ATC based on the simulation technology [18]. Some Chinese researchers use the Petri Net theory to model the protocol of TCS [19], [20], [21], [22], which provides a framework for simulation and modeling of the protocol. However, it still does not have a comprehensive verification means and its simulation theory and method also need to be improved further.

As an executable subset of PTL, MSVL is a useful formalism to describe system behaviors [23]. MSVL program is of distinct advantages because it is a temporal logic programming language with three modes: modeling, simulation and verification. With such a framework, the system descriptions, including the models for systems and environments, and the properties to be verified are all in the same language, which makes the simulation and verification easier. Based on the simulation method with MSVL, we can easily detect the design faults of the protocol of TCS.

VI. Conclusion

In this paper, we briefly present a simplified CTCS-3 protocol. Then the protocol is described by an MSVL program, and the property is specified by a PPTL formula. As we can see, simulation of CTCS-3 protocol with MSVL interpreter can well simulate the process of train's running, and give a series of important parameters when the train is running. The property described by a PPTL formula has been verified, and the simplified CTCS-3 protocol is correct. That is to say, if a train runs as the situations described by the MSVL program, it will be safe and there will be no disasters.

However, because the CTCS-3 protocol is real-time control system, we need to do some further studies for the simplified CTCS-3 protocol and add the real-time control command. This is a challenge to us in the near future. To do so, we also need to develop a new interpreter of MSVL to manage the simulation and verification of real-time CTCS-3 protocol.

Acknowledgement

The paper is supported by the NSFC Grant Nos. 61133001, 61202038, 61003078, 61272117, 61272118 and 973 Program Grant No. 2010CB328102. * Corresponding author: Z. Duan.

References

[1] ShuGuang Zhang, *Overall technical program of the CTCS-3*, Beijing China Railway Publishing House, 2008.

[2] Lech Znamirowski, Olgierd A.Palusinski and Sarma B.K. Vrudhula, *Programmable analog/digital arrays in control and simulation*, Analog Integrated Circuits and Signal Processing, 2004,39: 55-73.

[3] Linjing Hu, Zhengshun Sun, *Build and encapsulate custom block in Simulink*, Acta Simulata Systematica Sinica, 2004,3: 488-491.

[4] Jihua Ye, Dengwen Gan, Xiaohong Qiu and Jie Xue, *Simulink simulation of computer interface circuit*, Acta Simulata Systematica Sinica ,2007,6: 1234-1237.

[5] *Simulation of Designs C Modelsim Tool*, Digital VLSI Systems Design, 2007.

[6] J.E.Petersen, W.M.Kemp, V.S.Kennedy, W.C.Dennison and P.Kangas, *Tools for Design and Analysis of Experiments* , Enclosed Experimental Ecosystems and Scale, 2009.

[7] Zhenhua Duan, *An Extended Interval Temporal Logic and A Framing Technique for Temporal Logic Programming*, PhD Thesis, University of Newcastle upon Tyne (1996).

[8] Z. Duan, M. Koutny, *A Framed Temporal Logic Programming Language*, Journal of Computer Science and Technology 19 (2004)341–351.

[9] Zhenhua Duan, Xiaoxiao Yang and Maciej Koutny, *Semantics of Framed Temporal Logic Programs*, Proceedings of ICLP 2005, Springer-Verlag LNCS 3668(2005) 356–370.

[10] Zhenhua Duan, Xiaoxiao Yang and Maciej Koutny, *Framed Temporal Logic Programming*, Science of Computer Programming, 70(1): 31–61, 2008.

[11] Y. Kwon, G. Agha, *LTLC: Linear temporal logic for control*, Hybrid Systems: Computation and Control (HSCC), volume 4981 of Lecture Notes in Computer Science, (2008) 316–329.

[12] Zhenhua Duan, *Temporal Logic and Temporal Logic Programming*, Science Press, Beijing (2005).

[13] Xiaoxiao Yang, Zhenhua Duan, *Operational semantics of Framed Tempura*, J. Log. Algebr. Program. 78(1): 22-51 (2008).

[14] R.J ill, *Electric railway traction Part 5 Train detection, communication and supervision*, Power Engineering Journal, April 1996.

[15] *ERTMS/ETCS Functional Requirements Specification FRS. V*: 4.29. UIC. March 1 999

[16] Rong Guo, *Design and implementation of Radio Block Center in High-speed Railway*, Beijing Jiaotong University: 13-30, 2009.

[17] P. diTommaso, F. Flammini, A. Lazzaro, R. Pellecchia and A. Sanseviero,*The Simulation of Anomalies in the Functional Testing of the ERTMS/ETCS Traekside System*, Proceedings of the Ninth IEEE International SymPosium on High-Assurance Systems Engineering(HASE'05), 200535100.

[18] L. Jansen, M. Meyer and E. Sehnieder, *Technical issues in modeling the European train control system*, In Proc 1st CPN Workshop, DAIMIP-B532, Aarhus Uiversity, 1998:105115.

[19] K. Jensen, L.M. Kristensen, L. Wells, *Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems*, International Journal on Software Tools for Technol-ogy Transfer, Vol.9, 213C254 (2007).

[20] K. Jensen, *Coloured Petri Nets: A High Level Language for System Design and Analysis*, In: G. Rozenberg (ed.): Advances in Petri Nets 1990, LNCS 483, pp.342C416. Springer-Verlag(1991).

[21] M. Koutney, *Synthesis of Petri Nets with Localities*, Sci. Ann. Comp. Sci. 19, 1C23 (2009).

[22] M. Koutney, *A Compositional Model of Timed Petri Nets*, In: Proceedings of ICATPN 2000, pp.303C322 (2000).

[23] Z. Duan, M. Koutny and C. Holt, *Projection in Temporal Logic Programming*, Proceedings of Logic Programming and Automatic Reasoning, Springer-Verlag, F.Pfenning (ed.), LNAI 822 (1994) 333–344.