Online Verification of Control Parameter Calculations in Communication Based Train Control System

Lei Bu, Xin Chen, Linzhang Wang and Xuandong Li
State Key Laboratory for Novel Software Technology, Nanjing University
Department of Computer Science and Technology, Nanjing University
Nanjing, Jiangsu, P.R.China 210093
{bulei|chenxin|lwang|xld}@nju.edu.cn

Abstract. Communication Based Train Control (CBTC) system is the state-of-the-art train control system. In a CBTC system, to guarantee the safety of train operation, trains communicate with each other intensively and adjust their control modes autonomously by computing critical control parameters, e.g. velocity range, according to the information they get. As the correctness of the control parameters generated are critical to the safety of the system, a method to verify these parameters is a strong desire in the area of train control system.

In this paper, we use our experience learned during verifying a CBTC system to present our ideas of how to model and verify the control parameter calculations in a CBTC system efficiently.

– As the behavior of the system is highly nondeterministic, it is difficult to build and verify the complete behavior space model of the system offline in advance. Thus, we propose to model the system according to the ongoing behavior model induced by the control parameters.

– As the parameters are generated online and updated very quickly, say every 500 milliseconds in the case we met, the verification result will be meaningless if it is given beyond the time bound, since by that time the model will be changed already. Thus, we propose a method to verify the existence of certain dangerous scenarios in the model online quickly.

To demonstrate the feasibility of these proposed approaches, we present the composed linear hybrid automata with readable shared variables as a modeling language to model the control parameters calculation and give a path-oriented reachability analysis technique for the scenario-based verification of this model. We demonstrate the model built for the CBTC system, and show the performance of our technique in fast online verification. Last but not least, as CBTC system is a typical CPS system, we also give a short discussion of the potential directions for CPS verification in this paper.

1 Introduction

Nowadays, as communication has been embedded deeply into our daily life, computation has evolved from locating in one single standalone device to the collaboration of networks of equipments. In such a manner, more and more systems work in open environments, receive signals and stimuli from sensors, actuators and networks, then calculate their control modes and parameters accordingly. The newly generated control
modes and parameters will control the behavior of the system itself and the behavior of other components in the network as well dynamically. These systems have a tight integration of information systems and physical devices, which are named as Cyber-Physical Systems (CPS)\cite{13}.

By combining communication, computation and control (3C), in a CPS system physical devices can have more knowledge of the environment they are working in and the real-time status of the other elements which they are collaborating with. Thus, devices can autonomously generate more accurate instructions and gain advantages like safety, reliability and efficiency.

Public transportation is a typical area where CPS systems are emerging and playing more and more important roles. CBTC is the state-of-the-art technique in the train control area and fundamental for the building and controlling of high speed railway systems. During trains running on railways, the radio block center (RBC) will collect the position of each train periodically and compute the movement authority (MA), which is the distance that the train is authorized to go, for each train. Then the onboard train controller will compute the feasible velocity range by taking account of the movement authority and the current running parameters of the train, e.g., current position, velocity and etc. These are typical procedures of a CBTC system, which is clearly a Cyber-Physical System. One of the most important questions concern the design engineers of CBTC system is whether the parameters generated by the control functions used in the system are correct, e.g. trains will not collide with each other during operation.

In general, if we can build a model for the control parameter calculations and verify it, we can answer the correctness of parameters. Currently, most of the verification works consist of the following two steps: First, build the complete static formal model of the system. Second, verify the correctness of the model under the given property offline using techniques like model checking\cite{8}. For CBTC systems, as the input of the control functions, e.g. current velocity, position, movement authority and etc., are generated and collected online, it is hard to predict the complete behavior space of the system under verification. Thus, it is difficult to build and verify a complete static model of the system’s behavior offline in advance. To overcome this problem, we discuss our opinions about the verification of control parameter calculations in CBTC as follows:

- **Modeling**
  - As discussed above, it is hard to build and verify the complete behavior space model of the control functions offline in advance. We propose that the model should focus on the ongoing static behavior of the system in the short future driven by the current control parameter.
  - For modeling the ongoing behavior of the running CBTC systems, as the system is composed by large number of components, e.g., one control system for each train running on track, the model should be a composed system naturally.
  - Data are transmitted along with communication between components. Thus, the modeling language needs to support the representation of the synchronization among components and the data transmission along with it.

- **Verification**
  - The verification problem will not try to prove whether the control functions are correct or not. The verification procedure will focus on giving answers of the correctness of current parameters.
• As models are generated online, the verification procedure needs to be carried out online. As the model for the system will be updated quickly, it is necessary to give the verification result before the model is changed, which means the verification has to be time bounded and fast.

• A set of parameters can basically induce a series of operation modes in the short future, which can consist several scenarios of the operation of the system. What need to be verified is the existence of certain scenarios in the behavior of the model, which is represented as the reachability of certain paths in the model.

Therefore, from both the point of views of modeling and verification, in this paper we propose a new method to prove the correctness of the parameter calculations online during the CBTC system is in operation, which can result in an additional device deployed on-site, monitoring and guaranteeing the correctness of parameters online.

Based on this scheme, we present a formal model named as Hybrid Automata with Readable Shared Variable to model the control parameter calculations of the CBTC system, and a path-oriented reachability verification technique to verify the reachability property along with a path set in the model to achieve the goal of fast online verification. To demonstrate the feasibility of this scheme, the model for the control parameter calculations of the CBTC system is given in the paper, and several case studies are conducted on the model to illustrate the performance of the fast online verification.

Structure of The Paper. This paper is organized as follows. In the next section, we give a brief description of the running example of our study: Communication based train control system and summarize the requirements of verifying the control parameter calculations in a CBTC system. In Sec.3, we present our modeling language for the CBTC system: Composed Hybrid System with Readable Shared Variables, give the model we built for the CBTC system and show how to verify the existence of given critical scenarios on the system by the path-oriented reachability analysis method. Sec.4 verifies the existence of the dangerous scenarios in the model we built for the CBTC system and demonstrates the process ability of the path-oriented reachability method in online verification of CBTC systems. Sec.5 summarizes the related works on the verification of train control systems and proposes several potential directions in the verification of CPS systems based on our experience in verifying the CBTC system. Finally, the conclusion is stated in Sec.6.

2 Motivating Example: CBTC System

2.1 Communication Based Train Control System

A train control system is the heart for the safe and efficient operation of train systems. There are many organizations and projects devoted to the research and development of the train control system with high dependability. Many standards are proposed to give detail and comprehensive rule sets and guidances for the operation of railway systems for inter-vehicle and vehicle to infrastructure cooperation, like European Train Control System(ETCS)[5], and Chinese Train Control System (CTCS)[6]. According to different infrastructure utilities, data transmission methods and train control methodologies, ETCS/CTCS is divided up into several different equipmental and functional levels.
Communication based train control system (CBTC), on the high level of ETCS/CTCS, is believed to be the most advanced signaling technique and the fundamental method underlying the latest high speed railway systems. It uses data communication between trains and various control facilities to guarantee the safety and efficiency of train operation. It can be abstractly divided into two main parts: ground systems and onboard systems. Ground systems can track the runtime status of all the trains periodically. The radio block center (RBC) will send the needed information, e.g., movement authority, to the onboard systems on the train. Then the onboard systems will compute the velocity curve autonomously by taking account of the movement authority they received and the current operation status of the train. Ideally, the movement authority basically indicates a End-of-Authority (EOA) point which is with rear safe distance to the end of the train ahead. During the train operation, it also needs to guarantee that there is enough space for the train to completely stop by emergency braking before touching the EOA point, which is named as “Safe braking distance” (SBD). A simple illustration of the communication and movement authority granting is shown in Fig. 1.

Our running example is a typical CBTC system which is supposed to be used in a Urban Railway System in China. As the system is still under designing and debugging, our team join into the project to help verifying the correctness of the design of the ATP module of the CBTC system. One of the most interested question which bothers the engineers from the area of railway system is how to guarantee the absence of certain dangerous scenarios, e.g., train collision. We will use some of these scenarios to show the motivation of this paper, and introduce our thoughts about the verification of control parameter calculations in CBTC system.

In the design, all the trains need to communicate with RBC in 500 milliseconds period. RBC will grant the movement authority to each train by telling them the position of the EOA points. After that, the onboard computer will start to calculate the legal operation speed by taking account of the current speed of train, the limitation of the train and the track and so on. The train is free to move under the generated operating speed before reaching the safe braking distance point (SBD) which is with safe braking distance away from the EOA point. Once a SBD point is reached, the train will brake immediately to try to stop completely before move beyond the EOA point. When the train has not receive any signal from RBC for 5 seconds, the automatic train protection
(ATP) module of the CBTC system will take over the control of the train operation and ask the train to brake urgently as well. What the designers are worried about is whether the train can stop safely under the control parameters without beyond the movement authority and collide with the train ahead under certain scenarios.

2.2 Requirements For Verifying Control Parameter Calculations In The CBTC System

For verifying the control parameter calculations in the CBTC system, we need to build the formal model for the system and summarize the characteristics of the CBTC system at first:

- **Modeling**
  - **Static Model For Time-Bounded Behavior.** As the behavior of the CBTC system is highly nondeterministic, e.g., the input of the control functions including lots of runtime dynamic parameters, it is difficult to predict the complete behavior of the system or even verify it. Thus, the modeling and verification effort should focus on the ongoing behavior after synchronization or receiving signals which is the deterministic part of the entire nondeterministic behavior space.
  - **Compositional Verification.** The running CBTC system has all the operating trains on track and RBCs included. These components communicate with each other intensively, which is a concurrent system naturally. For each component, due to the dynamical behavior of system, it should be a hybrid automata. Thus, the model for the system should be a composed hybrid system.
  - **Shared Label and Variable.** The modeling language has to support the representation of the synchronization between components. Data are also transmitted during the synchronization. This is natural, because a component running in a system can not read the other components’ running parameters at anytime. We propose to use shared labels to represent the synchronization among component automata, and communication with other components are represented as shared variables in transition guards and reset actions on shared labels.

- **Verification**
  - **Online and Fast Verification.** As the environment of the CBTC working in is changing quickly, 500 ms in the running example, if we cannot give answer to the verification questions in 500 ms, the result will be meaningless. Thus, once a set of control parameters is calculated, the verification module needs to give a quick answer of whether this set of parameters will violate certain properties, e.g., safety. Therefore, The verification should be online and fast.
  - **Control Parameters Driven Verification.** As the verification procedure needs to be online and fast, it will not try to determine the correctness of the complex control functions beneath the system, but only give a quick answer to the correctness of the parameters generated.
  - **Time-Bounded and Scenario-Based Verification.** According to the requirements of designers of the certain CBTC system, the verification problem they concern most is checking whether certain bad scenarios will happen in the control modes induced by the current parameters. The scenario will be translated
as a sequence of control modes in the model, which constitutes a path. Thus, what needs to be verified is the reachability of certain property along with the path/scenario.

In summary, we think the modeling language for the control parameter calculations in CBTC system is composed hybrid automata with support of shared labels and shared variable reading. The model of the system should be a small static model induced by generated control parameters. The verification procedure should be scenario based path-oriented compositional reachability analysis.

3 Verifying The Control Parameter Calculations in CTBC systems

3.1 Modeling of The CBTC System

For a CBTC system in a train, the set of control parameters includes the current velocity range, the target velocity range, the location of the end of movement authority (EOA) point, the location of the safe braking distance (SBD) point, the position of the train itself and so on. Thus, this set of parameters shows a clear dynamic behavior of the train along with time, which can be modeled as a hybrid automaton (HA) naturally.

Now, let’s raise the field of our view from a single train to a series of trains running in a track. We will see that trains will communicate with RBCs and other trains during operation periodically. Data, e.g., the location of the train ahead, are transmitted to each train along with the communication. Thus, the model for the complete system should be a composed hybrid automata. Furthermore, in the composed system, component HAs synchronize with each other using shared labels, and a component can only read the value of the variable of other components on shared labels.

Based on the above discussion, we give the definition of the class of HA we proposed for CBTC systems as following:

Definition 1. A hybrid automaton with readable outer variables (HA\textsubscript{RV}) is a tuple $H = (X^l, X^s, \Sigma^l, \Sigma^s, V, V^0, E, \alpha, \beta, \gamma)$, where

- $X^l$ is a finite set of real-valued variables which belongs to $H$; $X^s$ is a finite set of real-valued variables which don’t belong to $H$, but can be read by $H$ in certain position; $X^l \cap X^s = \emptyset$
- $\Sigma^l$ is a finite set of local event labels which belongs to $H$ only; $\Sigma^s$ is a finite set of event labels which belongs to several $HA\textsubscript{RV}$; $\Sigma^l \cap \Sigma^s = \emptyset$
- $V$ is a finite set of locations; $V^0 \subseteq V$ is a set of initial locations.
- $E$ is a transition relation whose elements are of the form $(v, \sigma, \phi, \psi, v')$, where $v, v'$ are in $V$, $\sigma \in \Sigma^l \cup \Sigma^s$ is a label, $\phi$ is a set of transition guards of the form $f(y) \leq a$, and $\psi$ is a set of reset actions of the form $x := f(y)$, where $x \in X^l$, $a \in \mathbb{R}$, and
  - if $\sigma \in \Sigma^l$, $y \in X^l$;
  - if $\sigma \in \Sigma^s$, $y \in X^l \cup X^s$, if $y \in X^s$, we say $\sigma$ is $y$ related;
- $\alpha$ is a labeling function which maps each location in $V$ to a location invariant which is a set of variable constraints of the form $f(y) \leq a$ where $y \in X^l$, $a \in \mathbb{R}$. 


- $\beta$ is a labeling function which maps each location in $V$ to a set of flow conditions which are of the form $\dot{x} = g(y)$ where $x \in X^1$. For any $v \in V$, for any $x \in X^1$, there is one and only one flow condition $\dot{x} = g(y)$, where $x, y \in X^1$.

- $\gamma$ is a labeling function which maps each location in $V_0$ to a set of initial conditions which are of the form $x = a$ where $x \in X^1$ and $a \in \mathbb{R}$. For any $v \in V_0$, for any $x \in X^1$, there is at most one initial condition definition $x = a \in \gamma(v)$.

If each $f(y)$ is a linear expression, and $g(y) = [a, b]$, where $a, b \in \mathbb{R}$, we say this $HA_{RV}$ is a $LHA_{RV}$ (linear hybrid automaton with readable outer variables).

For a group of $HA_{RV}$, their composition $CHA_{RV}$ is defined as a product $HA_{RV}$ generated by synchronizing all the components with respect to the shared labels.

**Definition 2.** Let $H_1 = (X_1^1, X_1^2, \Sigma_1^1, \Sigma_1^2, V_1, V_0^1, E_1, a_1, \beta_1, \gamma_1)$ and $H_2 = (X_2^1, X_2^2, \Sigma_2^1, \Sigma_2^2, V_2, V_0^2, E_2, a_2, \beta_2, \gamma_2)$ be two $HA_{RV}$s, where $X_1^1 \cap X_2^1 = \emptyset, \Sigma_1^1 \cap \Sigma_2^2 = \emptyset$. The composition of $H_1$ and $H_2$, denoted as $H_1 || H_2$, is a $HA_{RV}$ $N = (X^1, X^2, \Sigma^1, \Sigma^2, V, V^0, E, a, \beta, \gamma)$ where

- $X^1 = X_1^1 \cup X_2^1$; $X^2 = X_1^2 \cup X_2^2$;
- $\Sigma^1 = \Sigma_1^1 \cup \Sigma_2^1$; $\Sigma^2 = \Sigma_1^2 \cup \Sigma_2^2$;
- $V = V_1 \times V_2$; $V^0 = V_0^1 \times V_0^2$;
- $\alpha((v_1, v_2)) = \alpha(v_1) \cup \alpha(v_2); \beta((v_1, v_2)) = \beta(v_1) \cup \beta(v_2); \gamma((v_1, v_2)) = \gamma(v_1) \cup \gamma(v_2)$;
- $E$ is defined as follows:
  - for $a \in \Sigma_1^1 \cap \Sigma_2^1$, for every $(v_1, a, \phi_1, v'_1)$ in $E_1$ and $(v_2, a, \phi_2, v'_2)$ in $E_2$, $E$ contains $((v_1, v_2), a, \phi_1 \cup \phi_2, \psi, (v'_1, v'_2))$;
  - for $a \in \Sigma_2^1 \cup \Sigma_2^2$, for every $(v, a, \phi, \psi, v')$ in $E_1$ and every $t$ in $V_2$, $E$ contains $((v, t), a, \phi, \psi, (v', t))$;
  - for $a \in \Sigma_1^2 \cup \Sigma_2^2$, for every $(v, a, \phi, \psi, v')$ in $E_2$ and every $t$ in $V_1$, $E$ contains $((t, v), a, \phi, \psi, (t, v'))$.

For all $m > 2$, the composition of $HA_{RV} H_1, H_2, \ldots, H_m$, denoted as $H_1 || H_2 || \ldots || H_m$, is a $HA_{RV}$ which is defined recursively as $H_1 || H_2 || \ldots || H_m = H_1 || H' \ldots || H_m$ where $H' = H_2 || H_3 || \ldots || H_m$.

Using the formal language defined above, we build a set of models of the system which includes nonlinear control functions as shown below. These models consist of two main parts:

- $n$ trains running on the track, the automaton for each train is shown in Fig. 2A.
- $m$ RBC centers, the automaton for each RBC is shown in Fig. 2B.

From Fig. 2, we can see the behavior of the system consists of the following aspects:

- Trains and RBCs communicate by two labels $updateMA$ and $syn$.
- After the global synchronization $syn$, an RBC will get the running parameters from the related train. Then the RBC will perform preprocessor job before it starts to compute and assign the latest MA to the related trains.
- After preprocessing, RBC will compute the new MA for the related trains using complex function $k$ and send them to the related trains by shared label $updateMA$.
- When received the new MA, each train will compute the local velocity and SBD using control function $f()$ and $g()$, and it will start to adjust the running velocity from current value $[c_i, c'_i]$ to the latest value range $[n_i, n'_i]$. 
During the adjustment period, we abstract the velocity to the mean of the old and new value of the velocity range.

- After the train $Train_i$ is running under the new velocity range, it will keep checking the current position to make sure it has not move beyond the safe braking point.
- Once the safe braking point is touched, the train will start to brake normally to try to stop completely before touching the end of the movement authority.
- And if the train has safely operated for 5 seconds without receiving any communication signal, the train will assume the communication channel is broken and an emergency braking will be executed immediately.
- Once a train starts the procedure of braking, it must stop completely in less than 5 seconds.

Considering a system with dozens of subsystems, e.g., trains and RBCs, and with complex nonlinear functions $f()$, $g()$, $k()$ included, it will be very difficult to verify properties on the model, as widely reported in literature [21]. Furthermore, many parameters used in functions $f()$, $g()$ and $k()$ are collected and generated online nondeterministically, e.g., temporary speed limitation, wind speed, mass of the train and etc, even there is a method to verify the complex nonlinear function, as these critical parameters cannot be predicted ahead precisely, the offline verification of the system is still very difficult.

3.2 Scenario Based Verification and Path-oriented Reachability Analysis

For the verification of the control parameter calculations in CBTC systems, one of the problems which the designers concern most is when $Train_i$ starts to brake, whether it can stop completely before passing the EOA point or even collide with the ahead train under the generated parameters. This problem indicates an execution scenario of the behavior of each train in the system, from location compute to $Ebraking$, and a target property to verify: the physical position of $Train_i$ equals with the ahead one $Train_{i-1}$.
Scenario-Based Automata. As discussed in the last section, the verification of the given scenario-based property on the models given in Fig. 2 is very difficult. On the other hand, the control parameters generated by the control functions can induce a static control model of the behavior of the CBTC system in the short future before the generation of the next set of parameters.

Based on this idea, we simplify the models given in the last section. As the control parameters are already calculated and saved, e.g., $ma_c$ and $sbd_c$ for the movement authority and safe braking distance of $Train_i$, the control functions can be dismissed in the new scenario-based model. The component RBCs can also be dismissed from the system, because the scenario-based automaton stands for the behavior of the system after the latest MA is already granted and before the next communication, during that period the train $Train_i$ doesn’t have to communicate with any RBC. As a result, we build the scenario-based static running automata for $Train_i$ to a $LHA_{RV}$ as below in Fig. 3.

![Scenario-Based LHA RV](image)

This scenario on a single $LHA_{RV}$ $Train_i$ is presented as an evolution of the system from locations to locations, e.g., $\langle comp, cv \rangle \rightarrow \langle adjust, op \rangle \rightarrow \langle cruise, EBrake \rangle$ in the automaton $Train_i$. Using the same notion given in [11], we name such a sequence of locations as a path. By assigning each location with a nonnegative real number, we can get a timed sequence in the form of

$$\left[ \begin{array}{l} \text{comp} \\ \delta_0 \\ v_1 \\ \delta_1 \\ v_2 \\ \delta_2 \\ \vdots \\ v_n \end{array} \right]$$

This timed sequence represents a behavior of the model such that the system starts at location $compute$, stays there for $\delta_0$ time units, then jumps to location $adjust$ by transition $cv$ and stays at $adjust$ for $\delta_1$ time units, and so on.

Let $N = H_1 || H_2 || \ldots || H_m$ be a $CLHA_{RV}$ where $H_i = (X_i^l, X_i^r, \Sigma_i^l, \Sigma_i^r, V_i, V_0^i, E_i, \alpha_i, \beta_i, \gamma_i)$ $(1 \leq i \leq m)$ is an $LHA_{RV}$ and $\rho$ be a path in $N$ of the form $\rho = \langle v_0 \rangle \begin{array}{c} \delta_0 \\ \sigma_0 \end{array} \langle v_1 \rangle \begin{array}{c} \delta_1 \\ \sigma_1 \end{array} \ldots \begin{array}{c} \delta_n \\ \sigma_n \end{array} \langle v_n \rangle$. It follows that $v_i = (v_{i1}, v_{i2}, \ldots, v_{im})$ $(0 \leq i \leq n)$ where $v_{ik} \in V_k$ $(1 \leq k \leq m)$. For any $k$ $(1 \leq k \leq m)$, we construct the sequence $\rho_k$ from $\rho$ as follows: replace any $v_i$ with $v_{ik}$ $(0 \leq i \leq n)$, and for any $\langle \phi, \psi \rangle$ $(1 \leq i \leq n)$, if $(v_{i-1k}, \sigma_{i-1}, \phi, \psi, v_{ik}) \in E_k$, then replace it with $\langle \phi, \psi \rangle$ $(v_{ik})$, otherwise remove it. It follows that $\rho_k$ is a path in $H_k$. We say that $\rho_k$ is the projection of $\rho$ on $H_k$. Intuitively, $\rho_k$ is the execution trace of $N$ on $H_k$ when $N$ runs along $\rho$. Thus, the complete scenario is a path set for the system, consisting of one path for each component.
Reachability Specification. Now, let us look at the reachability specification: During braking, Train \(_i\) collide with the ahead train Train\(_{-1}\), which means the position of Train\(_i\) is the same with the ahead one Train\(_{-1}\) in the location EBraking. This property can be formally translated as Train\(_i\).x = Train\(_{-1}\).x in location EBraking.

For an LHA \(H = (X', X^i, \Sigma^i, \Sigma', V, V^0, E, \alpha, \beta, \gamma)\), a reachability specification, denoted as \(R(v, \varphi)\), consists of a location \(v\) in \(H\) and a set \(\varphi\) of variable constraints of the form \(a \leq c_0 x_0 + c_1 x_1 + \cdots + c_i x_i \leq b\) where \(x_i \in X' \cup X^i\) for any \(i (0 \leq i \leq l)\), \(a, b\) and \(c_i (0 \leq i \leq l)\) are real numbers.

**Definition 3.** Let \(H = (X, \Sigma, V, V^0, E, \alpha, \beta, \gamma)\) be an LHA\(_{RV}\), and \(R(v, \varphi)\) be a reachability specification. A behavior of \(H\) of the form \(\langle v_0 \delta_0 \rangle_{c_0} \langle v_1 \delta_1 \rangle_{c_1} \cdots \langle v_n \delta_n \rangle_{c_n} \) satisfies \(R(v, \varphi)\) iff \(v_n = v\) and each variable constraint in \(\varphi\) is satisfied when the automaton has stayed in \(v_n\) for delay \(\delta_n\), i.e. for each variable constraint \(a \leq c_0 x_0 + c_1 x_1 + \cdots + c_i x_i \leq b\) in \(\varphi\), \(a \leq c_0 \zeta_0(x_0) + c_1 \zeta_1(x_1) + \cdots + c_i \zeta_i(x_i) \leq b\) where \(\zeta_n(x_i)\) \((0 \leq k \leq l)\) represents the value of \(x_i\) when the automaton has been at \(v_n\) for the delay \(\delta_n\). \(H\) satisfies \(R(v, \varphi)\) iff there is a behavior of \(H\) which satisfies \(R(v, \varphi)\).

**Definition 4.** Let \(N = H_1 \| H_2 \| \ldots \| H_m\) be a CLHA\(_{RV}\), \(P = \{\rho_1, \rho_2, \ldots, \rho_m\}\) be a path set, where \(\rho_i\) is a finite path in \(H_i\) \((1 \leq i \leq m)\), and \(R(v, \varphi)\) be a reachability specification. \(P\) satisfies \(R(v, \varphi)\) if and only if there is a path \(\rho\) of \(N\) that the projection of \(\rho\) on \(H_i\) is \(\rho_i\) \((1 \leq i \leq m)\), and there is a behavior of \(N\) which satisfies \(R(v, \varphi)\).

Path-Oriented Reachability Analysis. In this paragraph, we will show how to verify the reachability specification along with a path set in a CLHA\(_{RV}\) system using linear programming efficiently.

Generally speaking, the model checking problem for hybrid systems is very difficult. Even for a single LHA, the reachability analysis problem is undecidable \([1][2][3]\). The performance of existing techniques for compositional analysis of LHA systems is even worse. The state-of-the-art tool HYTECH \([4]\) and its improvement PHAVer \([7]\) need to compute the composition of the whole system into a unique global automaton then use expensive polyhedra computation for reachability analysis, which will suffer the problem of state explosion and greatly restrict the solvable problem size.

To overcome this drawback, in study\([11]\) we presented an efficient approach for the path-oriented reachability analysis of LHA compositions. This technique checks a group of paths at a time, one path for each LHA, all of the paths are transformed into a group of linear constraints automatically. Then, a few constraints about the system integration according to the synchronization events in each path will be added to ensure that the components cooperate correctly. It follows that the reachability problem along those specific paths can be reduced to a linear program. Using this method both the path length and the number of participant automata checked can be scaled up greatly to satisfy practical requirements. This approach of symbolic execution of paths can be used by design engineers to check critical paths, and thereby increases the faith in the system correctness. This path-oriented technique can be easily scaled to use in CLHA\(_{RV}\) systems. We will use a simple example to illustrate our idea below.
The reachability specification is whether the property $\text{global location (path)} \to \text{linear constraints}$ that represents all the timed runs corresponding to the path. Take the location $\langle \delta_1, \delta_2, \delta_3 \rangle$ as an example in label $e$ of the shared labels, for example, in label $e$ of the system $T$, the transition guard is $s + t > k$. The reachability specification is whether the property $s + 2t - 3k = 0$ can be satisfied at the global location $(s_3, t_5, k_5)$.

In our path-oriented approach, for each of these three paths we generate a group of linear constraints that represents all the timed runs corresponding to the path. Take the path $\langle t_1 \rangle \to \langle t_2 \rangle \to \langle t_3 \rangle \to \langle t_4 \rangle \to \langle t_5 \rangle$ of the system $T$ for example:

- Use $\left\{ t_i, \delta_i \right\}$ to indicate that the system has stayed in location $t_i$ for time delay $\delta_i$ (non-negative variable). The behavior of the system is represented by $\left\{ t_1, \delta_1 \right\} \to \left\{ t_2, \delta_2 \right\} \to \left\{ t_3, \delta_3 \right\} \to \left\{ t_4, \delta_4 \right\} \to \left\{ t_5, \delta_5 \right\}$ where $\delta_1, \delta_2, \delta_3, \delta_4, \delta_5$ must satisfy all the time constraints enforced by the system, which forms a group of linear constraints.

  - For each location $t_i$, two variables $\gamma_i(t)$ and $\zeta_i(t)$ are generated to represent the valuation of $t$ when entering $t_i$ and leaving $t_i$ after stay there by $\delta_i$ time units.
  - Take the location $t_3$ for example, according to the flow condition, $1.1\delta_3 + \gamma_3(t) \geq \zeta_3(t) \geq 0.9\delta_3 + \gamma_3(t)$.
  - For the transition guard $t < 5$ on the local transition $g$, we have $\zeta_4(t) < 5$.
  - For the reset action $t = 2$ on the local transition $d$, we have $\gamma_4(t) = 2$.

- Synchronization constraints will be added to ensure that these three components cooperate accurately according to the synchronization events, which are illustrated by the dashed lines and $\Theta_{(event)}$ in Fig 4(B).

  - For the event $b$ shared by $S$ and $T$, we have $\delta_1 = \delta_1 + \delta_2$.
  - For the transition constraints including outer variable reading, e.g., $s + t > k$ in $e$, we have $\zeta_3(s) + \zeta_3(t) > \zeta_2(k)$.
  - All the components have spent exact the same time, e.g., for $S$ and $T$, we have $\delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5 = \delta_1 + \delta_2 + \delta_3 + \delta_4 + \delta_5$.

  - For reachability specification $s + 2t - 3k = 0$, we get $\zeta_3(s) + 2\zeta_3(t) - 3\zeta_3(k) = 0$. 

Fig. 4. Sample Automata And The Path-oriented Reachability Encoding
Above all, the path-oriented reachability analysis problem is transformed to a feasibility problem of a set of linear constraints. It is well-known that the feasibility problem of linear constraints can be solved by linear programming (LP) technique efficiently. Utilizing LP solver, we can develop an efficient tool for path-oriented reachability analysis of $CLHARV$ where the length of the path, the size of each $LHARV$, and the number of components are all close to the practical problem scales. Thus, we can gain the objective of fast verification of the existence of certain scenarios in the model of control parameter calculations in CBTC systems.

4 Experimental Evaluation

To demonstrate the modeling and verification techniques for control parameter calculations in the CBTC system proposed in this paper and show the ability of fast verification of the path-oriented reachability method, we verify the train collision scenario given in the last section using the model built in Sec.3.

The scenario we selected to verify is if the communication channel fails during train operation, whether all the train can stop safely without collide with each other, and the corresponding scenario-based automata we built for each train is shown in Fig.3. The model represents the path: $\langle \text{compute} \rangle \rightarrow \langle \text{adjustment} \rangle \rightarrow \langle \text{cruise} \rangle \rightarrow \langle \text{EBraking} \rangle$ for each train $Train$, and the reachability specification is the positions of two nearby trains are equal with each other, for example $train_1.x = train_2.x$. Since the system is still under simulation and debugging, we use a group of traditional running values for the parameters in the model from our colleagues in the railway area.

The experiments are conducted in an ongoing version of BACH$^{[9,10]}$, which is a toolset for building LHA models and verifying the bounded reachability property of LHA systems, and can be downloaded from [http://seg.nju.edu.cn/BACH/](http://seg.nju.edu.cn/BACH/). On a DELL workstation (Intel Core2 Quad CPU 2.4GHz, 4GB RAM), we evaluate the potential of the path-oriented reachability analysis method presented in this paper using the CBTC model shown in Fig.3.

The experiment data is shown in Table 1. The largest problem BACH can solve in 500 ms consists of 16 trains which is a very complex system and enough for a running urban railway system. According to the consultation to the engineers in the urban railway company, it is expected that the number of trains under operation on a normal track is around 15 to 20. Thus, the technique presented in this paper is applicable to be used in daily operation. The parameter we used in the model is proved to be safe by verification, which means certain path-oriented reachability specifications are not satisfied. Meanwhile, the runtime memory overhead of the computation, which is not listed in the table, is very small.

The data in Table 1 gives a clear demonstration of the process ability of fast verification of the bad scenario in the model for control parameter calculations. It also strengthens our belief that this technique can be used online during system in operation to guarantee the correctness of the important control parameters. As the linear programming solver underlying BACH is a free collection of Java classes for research [12], we believe if the linear programming package is replaced by an advanced commercial one, the performance will be even better.
Table 1. Experimental Data on the CBTC System

| Path | Constraint | Variable | Time  |
|------|------------|----------|-------|
| 8    | 1208       | 96       | 0.175s|
| 10   | 1550       | 120      | 0.335s|
| 12   | 1908       | 144      | 0.328s|
| 14   | 2282       | 168      | 0.404s|
| 16   | 2672       | 192      | 0.469s|

5 Related Work and Further Discussion

5.1 Related Work

The verification of the train control system has been intensively studied. Study [17] gives a method to generate the high level requirements from a subset of the specification of an ETCS[5] system, and use method in [18] to verify the consistency between requirements. These two works belong to the category of requirement engineering, which don’t touch real time behaviors of the system.

Study [19] models the communication in train control systems with Live Sequence Chart (LSC), then validate the LSC by model checking and testing. Study [20] models the behavior of train control systems by timed state transition systems and verify the given property by bounded model checking and compositional reasoning. These studies all give high level models for behaviors of the system without considering the dynamic behavior of the movement of train.

Study [16] models a fully parametric ETCS system using differential dynamic logic and verify the system by logical deductive verification. Study [15] builds different complex models for different layers of an ETCS system and verify these models using layer-specific technologies. These works build static model for the ETCS system without considering the system as a dynamic system which works in open environment. Thus, they only include rather limit parameters used in the control functions in the model.

5.2 Verification of CPS Systems

The new CPS computing paradigm brings new challenges and requirements to the research community, like how to guarantee the qualities of service, how to generate the formal models for the system and so on, which are proposed and summarized in many studies like [13][14]. The CBTC system is a typical CPS system which combines communication, computation and control tightly. From the experience we learned during verify the CBTC system, we think Control Parameter Calculations Verification could be an emerging topic in the verification of CPS systems. Furthermore, we summarize following subtopics we think is worth studying and paying attention to:
– **Modeling Language.** CPS systems are running under dynamic environments. They receive signals from each other and the environment in an unpredictable way. How can the nondeterminism be modeled and verified? For CBTC systems, we choose to use linear hybrid automata as the modeling language and focus on the modeling of the ongoing static behavior of the system once the control parameters are generated. How about for general CPS system, do we need to introduce a new language?

– **Time Bounded Verification.** Compared with classical verification which try to prove the correctness of the complete behavior of the system, the verification of CPS system focuses more on the correctness of the behavior in given time bound, e.g., will the train collide with the ahead one in 500 milliseconds in this paper. This is a new direction of Bounded Model Checking\cite{BMC}, where the term “bound” means time, rather than “steps” used in classical Bounded Model Checking.

– **Online and Fast Verification.** As the control parameters of CPS system are changing quickly, the verification module needs to give a quick answer of the correctness of the new generated set of parameters. We think it is necessary to investigate how to build fast and low-overhead online verification techniques for CPS systems.

6 Conclusion

In this paper, we introduce our experience in modeling and verifying the control parameter calculations in a CBTC system which is a typical CPS system. Based on our study of this system, we propose our ideas of the requirements for modeling and verifying control parameters in a CBTC system. For modeling language, we think it should be a composed hybrid system with support of component communication and data transmission. For verification technique, we insist the verification for CBTC systems should be online and fast verification of the ongoing behavior in the short future, and the problem needed to be verified is the existence of certain dangerous scenarios.

To demonstrate our ideas, we introduce a notion Composed Linear Hybrid Automata with Readable Shared Variables to model the behavior of the CBTC system induced by the control parameters. We also present a path-oriented reachability analysis method to achieve the objective of the online scenario-based verification. The experiment results support our belief a lot by showing the great process ability of fast solving of a system consists of 16 trains in less than 500 milliseconds which is the period of parameter generation in the CBTC system.

Currently, with the help of our colleagues from railway areas, we are trying to implement this technique into a standalone device which can be integrated and deployed into the onboard ATP module as a part of the CBTC system to check the correctness of the velocity range given by ATP. Safety critical scenarios can be enumerated by CBTC engineers ahead, the model pattern corresponding to these scenarios can be designed in advance also. Then the device can catch the latest generated parameter set, build the related models using the pattern and verify them online. It is supposed to work as a runtime monitor/checker on the train under experimentation to guarantee the safety of the control parameters before the parameters are utilized.
References

1. T. Henzinger. The theory of hybrid automata. In Proc. of the 11th Annual IEEE Symposium on Logic in Computer Science, pp. 278-292.
2. R. Alur, C. Courcoubetis, N. Hallbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, S. Yovine. The algorithmic analysis of hybrid systems. In Theoretical Computer Science, 138(1995), pp.3-34.
3. T. Henzinger, P. Kopke, A. Puri, and P. Varaiya. What’s Decidable About Hybrid Automata? In Journal of Computer and System Sciences, 57:94-124, 1998.
4. T. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: a model checker for hybrid systems. In Software Tools for Technology Transfer, 1:110-122, 1997. 13.
5. ERTMS User Group, UNISIG: ERTMS/ETCS System requirements specification. http://www.era.europa.eu (2002) Version 2.2.2.
6. B.Ning, T.Tang, K.Qiu, C.Gao, and Q. Wang. CTCS-Chinese Train Control System. In Computers in Railway IX, 2004
7. G. Frehse. PHAVer: Algorithmic Verification of Hybrid Systems past HyTech. In Proc. of Hybrid Systems: Computation and Control’05, LNCS 2289, pp.258-273.
8. E. Clarke, O. Grumberg, and D. Peled. Model Checking. The MIT Press, 1999.
9. L. Bu, Y. Li, L. Wang and X. Li. BACH : Bounded ReachAbility CHecker for Linear Hybrid Automata. In Proc. of FMCAD’08, IEEE Computer Society Press, 2008, pp.65-68.
10. L. Bu, Y. Li, L. Wang, X. Chen and X. Li. BACH 2: Bounded ReachAbility CHecker for Compositional Linear Hybrid Systems, In Proc of DATE’10, pp. 1512-1517, 2010.
11. L. Bu, and X. Li. Path-Oriented Bounded Reachability Analysis of Composed Linear Hybrid Systems, In Software Tools Technology Transfer, 2010, DOI:10.1007/s10009-010-0163-9.
12. OR-Objects of DRA Systems. http://OpsResearch.com/OR-Objects/index.html.
13. E. Lee. Cyber-Physical Systems– Are Computing Foundations Adequate? In Position paper for NSF Workshop on Cyber-Physical Systems: Research Motivation, Techniques and Roadmap, 2006.
14. E. Clarke, B.Krogh, A.Platzer, and R.Rajkumar. Analysis and Verification Challenges for Cyber-Physical Transportation Systems. In National Workshop for Research on High-Confidence Transportation Cyber-Physical Systems: Automotive, Aviation & Rail, 2008
15. W. Damm, A. Mikschl, J. Oehlerking, E.R. Olderog, J. Pang, A. Platzer, M. Segelken, and B. Wirtz. Automating verification of cooperation, control, and design in traffic applications. In Formal Methods and Hybrid Real-Time Systems. LNCS 4700, Springer, 2007.
16. A. Platzer, and J. Quesel. European Train Control System: A Case Study in Formal Verification. In Proc. of the 11th International Conference on Formal Engineering Methods, ICFEM, Rio de Janeiro, Brasil, LNCS 5885, pp. 246-265. Springer, 2009.
17. A. Chiappini, A. Cimatti, L. Macchi, O. Rebollo, M. Roveri, A. Susi, S. Tonetta, and B. Vittorini. Formalization and Validation of a subset of the European Train Control System. In Proc. of ICSE’10, pp.109-118, ACM, 2010.
18. A. Cimatti, M. Roveri, and S. Tonetta. Requirements Validation For Hybrid Systems. In Proc. of CAV’09, LNCS 5643, Springer, Grenoble, 2009.
19. J. Bohn, W. Damm, H. Wittke, J. Klose, and A. Moik. Modeling and Validating Train System Applications Using Statemate and Live Sequence Charts. In Proc. of IDPT2002, Society for Design and Process Science, 2002.
20. J. Peleska, D. Groe, A. Haxthausen, and R. Drechsler. Automated verification for train control systems. In Proc. of FORMS/FORMAT. (2004)
21. B. I. Silva, O. Stursberg, B. H. Krogh and S. Engell. An Assessment of the Current Status of Algorithmic Approaches to the Verification of Hybrid Systems, In Proc. of 40th Conference on Decision and Control. 2001, volume 3, 4-7, pages 2867–2874.
22. A. Biere, A. Cimatti, E. M. Clarke, O. Strichman, Y. Zhu. Bounded Model Checking. In *Advance in Computers*, Vol.58, 2003.