HCSP formal modeling and verification method and its application in the hybrid characteristics of a high speed train control system

J. Lv[1], K. Li[1], T. Tang[2] & L. Chen[2]
[1]*National Engineering Research Center of Rail Transportation Operation and Control System, Beijing Jiaotong University, China*
[2]*State Key Laboratory of Rail Traffic Control and Safety, Beijing Jiaotong University, China*

Abstract

The high speed train control system is a typical hybrid system, which not only contains a continuous evolution process (train position and speed), but also the discrete event between subsystems. Although some formal methods like HUML, HA and DL have already been used in modeling and verification train control systems, they are not good at describing communication behaviors which are in the interactive process of subsystems. To overcome this problem, we introduce a formal modeling and verification method for hybrid systems. First, we use HCSP to model the behavior of the system. Second, we transit the HCSP models to HA models by introducing some transition rules. Finally we input these HA models to PHAVer which is a tool for verifying safety properties of hybrid systems to automatic verification. Based on the simulation and analysis of a Movement Authority scenario in high speed train control system specifications, the method is proven to be validated.
Keywords: high speed train control system, HCSP, Hybrid Automata, Movement Authority scenario.

1 Introduction

The high speed train control system is a typical hybrid system, which not only contains a continuous evolution process (such as train position and speed

WIT Transactions on The Built Environment, Vol 127, © 2012 WIT Press
www.witpress.com, ISSN 1743-3509 (on-line)
doi:10.2495/CR120021

process), but also the discrete event between subsystems (some communication behaviors). Some formal methods have already been used in researching the hybrid property of train control systems. Kirsten proposes a formal language of a hybrid system based on UML 2.0 and a layered structure formal model has been analyzed in the railway crossing [2, 3]. OCL (object constraint language) is introduced to HUML by Ziemann and Gogolla and a reasonable train dynamic behavior model is described using HUML in the BART railway line train control system [4]. Platzer first introduced DL which contains a discrete and differential behavior programming verification in formal modeling and verification train control system [5–7]. Based on the tool Keymera, the ETCS-3 specification is proven to be collision avoidance. An automatic hierarchical design framework has been given by Damm *et al*. [8] and, based on HA, the cooperation protocol in a train control system is proved to be collision avoidance [9].

Referring to the above formal methods, HUML is good at modeling the train control system, but the verification is a hard problem. Although HA and DL have already been successfully used in modeling and verification train control systems, they are not good at describing communication behavior in the interactive process of subsystems. To overcome this problem, we introduce a formal modeling and verification method for a train control system. First, we use the HCSP (Hybrid Communication Sequential Process) [10] to model the behavior of the system and then we transit the HCSP model to the HA model by introducing some transition rules. Finally we input these HA models into PHAVer [11] which is a tool for automatically verifying safety properties. Based on the simulation and analysis of a Movement Authority scenario in high speed train control system specifications, the method is proven to be validated.

2 Model transition

2.1 Assumption

In HCSP, some sequential processes like communication and assignment consume time. We give the following two assumptions:

(1) The assignment process does not consume any time (the time consumed in the assignment process is far less than the time transits in a real physical process)
(2) The communication synchronization process does not consume any time, but the latency time between communications also exists.

In order to automatically verify the property of the HCSP model, we transit the HCSP models to the HA models by introducing some rules.

2.2 Definition

The translation function $HA : HCSP \to HA$ is defined in terms of function $\Phi : B(V) \times B \to HA_{frag}$ that translates a HCSP process term p with a set of jumping continuous variables to a corresponding hybrid automaton fragment HAfrag.

Hybrid automaton fragment: a hybrid automaton fragment HAfrag is a tuple $(V, v_0, inv, flow, done, E, source, target, guard, jump, \sum, event)$, where V, inv, $flow$, E, $source$, $target$, $guard$, $urgent$, $jump$, $\sum$ and $event$ are defined as in the hybrid automation definition. Location $v_0 \in V$ is the initial location of the hybrid automaton fragment, and function done V → $true, false$ assigns to each location $v \in V$ a status (also known as done condition) that partitions the locations into terminating locations (status is true), and non-terminating locations (status is false).

Translation function HA Function HA is now defined as follows: Let $\Phi_J(p) = (V_p, v_{0p}, inv_p, flow_p, done_p, E_p, source_p, target_p, urgent_p, guard_p, jump_p, \sum_p, event_p)$ be the hybrid automaton fragment of p with the set of jumping continuous variables *J*, and *X* as defined all the variables in HA. More information is shown in reference [12].

2.3 Transition rules

2.3.1 Atom process

(1) *Stop*

Since process term *stop* can neither perform any action transitions, nor time transitions, $\Phi_J(Stop)$ has one location with flow condition false, invariant true, and no outgoing edges. Let

$\Phi_J(Stop) = (\{v_0\}, v_0, inv, flow, done, \phi, \phi, \phi, \phi, \phi, \phi, \phi, \phi)$ be the transition rule, where: $inv(v_0) = true$, $flow(v_0) = false$, $done(v_0) = false$

(2) *Skip*

Like process term *stop*, the process term *skip* can also neither perform any action transitions, nor time transitions, $\Phi_J(Skip)$ has one location with flow condition false, invariant true, and done true. It means the process terminates immediately with no effect.

$$\Phi_J(Skip) = (\{v_0\}, v_0, inv, flow, done, \phi, \phi, \phi, \phi, \phi, \phi, \phi, \phi)$$

(3) $v := e$

$$\Phi_J(v := e) = (\{v_0, v_1\}, v_0, inv, flow, done, E, source, target, urgent, guard, jump, \tau, event)$$

where:

$$inv(v_0) = true, source(E) = v_0, flow(v_0) = true, flow(v_1) = false,$$
$$target(E) = v_1, done(v_0) = false, done(v_1) = true, urgent(E) = true, guard(E) = true,$$
$$jump(E) = (x, x = e), event(E) = \tau$$

v: = *e* assigns the value of expression *e* to *v* and then terminating. $\Phi_J(v := e)$ behaves as the action predicate $v := e$ with initial state v_0, where the flow condition true, invariant true, and done false. An instantaneous change from state v_0 to another state v_1 with flow condition false, invariant true, done true, guard true, jump true and event true.

Figure 1: Assignment.

(4) $ch!e$

$$\Phi_J(ch!e) = (\{v_0, v_1\}, v_0, inv, flow, done, E, source, target, urgent, guard, jump, ipa(ch), event)$$

where:

$$inv(v_0) = true \; inv(v_1) = true, \; source(e) = v_0, \; flow(v_0) = true, \; flow(v_1) = false \; target(E) = v_1,$$
$$done(v_0) = false, \; done(v_1) = true, \; guard(E) = true, \; jump(E) = \{X, e = ch'\}, \; event(E) = isa(ch)$$

$ch!e$ is willing to send the value of the expression e to the output channel ch. $\Phi_J(ch!e)$ behaves as an undelayable sending of expression e via channel ch from initial state v_0 to another state v_1 with flow condition false, invariant true, done true, guard true, jump true and event true if another process $ch?x$ is prepared. Otherwise, it performs arbitrary time transitions.

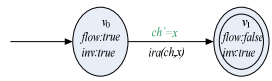

Figure 2: Send process.

(5) $ch?x$

$$\Phi_J(ch?x) = (\{v_0, v_1\}, v_0, inv, flow, done, E, source, target, urgent, guard, jump, isa(ch), event)$$

where:

$$init = (\phi, true), \; inv(v_0) = true, \; inv(v_1) = true \; source(E) = v_0, \; flow(v_0) = true,$$
$$flow(v_1) = false, \; target(E) = v_1, \; done(v_0) = false, \; done(v_1) = true, \; urgent(E) = true,$$
$$guard(E) = true, \; jump(E) = \{X, ch' = x\}, \; event(E) = ira(ch, x)$$

$ch?x$ is willing to accept a message from the channel ch and assigns it to a variable x, which could be discrete or continuous. $\Phi_J ch?x$ behaves as undelayable receiving information of e via channel ch into variable x from initial state v_0 to another state v_1 with flow condition false, invariant true, done true, guard true, jump true and event true if another process $ch!e$ is prepared. Otherwise, it performs arbitrary time transitions.

2.3.2 Composition process

(1) $p;q$

$$\Phi_J(p;q) = (V, v_{0p}, inv_{pq} \mid V, flow_{pq} \mid V, done_{pq} \mid V \blacklozenge E_{pq}, source_{pq}, target \cup target_q, urgent_{pq}, guard_{pq}, jump_{pq}, \sum_{pq}, event_{pq})$$

where:

$$V = \{v \mid v \in V_p, \neg done_p(v)\} \cup V_q, dom(target)_{pq} = E_p$$

$$\forall_{e \in E_p} : target(e) = \begin{cases} v_{0q} & if \; done_p(target_p(e)), \\ target_p(e) & otherwise, \end{cases}$$

$p;q$ behaves like p first and if p terminates then it behaves like q. The initial location of $\Phi_J(p;q)$ is the initial location of $\Phi_J(p)$. The end-points of the edges that go to terminating locations of $\Phi_J(p)$ are reconnected to the initial location of $\Phi_J(q)$.The behavior of $\Phi_J(p;q)$ is straightforward: first p is executed, then q. It follows that if the hybrid automaton fragment for p has no terminating locations, i.e. $v \in V_p, \neg done_p(v)$, then the locations from $\Phi_J(q)$ are unreachable.

(2) $p \,||\, q$

$$\Phi_J(p \,||\, q) = (V, (v_{0p}, v_{0q}), inv \cup inv_{pq}, flow \cup flow_{pq}, done \cup done_{pq}, E \cup E_{pq}, source \cup source_{pq}, target \cup target_{pq}, urgent \cup urgent_{pq}, guard \cup guard_{pq}, jump \cup jump_{pq}, \sum_{pq}, event U event_{pq})$$

where:

$$E = (E_p \times (V_q \setminus V_q^{done})) \cup (E_q \times (V_p \setminus V_p^{done})) \cup E_p \cup E_q) \cup \{(e_p, e_q) \in E_p \times E_q\}$$

$$dom(source) = dom(target) = dom(urgent) = dom(guard) = dom(jump) = dom(event) = E \setminus E_{pq}$$

$p \,||\, q$ is a parallel process which behaves as if two processes are working independently except that all communications along channel *ch* between p and q to be synchronized. An additional terminating location V^{done} is introduced. Action transitions from the components are interleaved, apart from the synchronization of matching send and receive actions, in which they are executed simultaneously. $\Phi_J(p \,||\, q)$ means that if the synchronization of matching send and receive actions can terminate successfully, the control of the hybrid automaton ends up in the terminating location V^{done}.

(3) $\mu X.(p;X)$

$$\Phi(\mu X.(p;X)) = (V, v_{0p}, inv_p | V, flow_p | V, done_p | V, E_p, source_p, target, urgent_p, guard_p, jump_p, \sum_p, event_p)$$

where:

$$V = \{v \mid v \in V_p, \neg done_p(v)\}, dom(target) = E_p$$

$$\forall_{e \in E_p} : \; target(e) = \begin{cases} v_{0p} & if \; done_p(target_p(e)), \\ target_p(e) & otherwise. \end{cases}$$

$\mu X.(p;X)$ is a tailed recursive process which represents infinite repetition of term p. $\Phi(\mu X.(p;X))$ behaves as that if the process p terminates in and end-point location, it will reconnect to the initial location v_{0p}.

(4) $\trianglerighteq io \rightarrow p$

$$\Phi(\trianglerighteq io \rightarrow p) = (V_p, v_{0p}, inv_p, flow_p, done_p, E_p, source_p, target_p, urgent_p, guard_p, jump_p, \sum_p, event_p)$$

where:

$$\forall_{e \in E_p} : \; urgent(e) = \begin{cases} true & if \; event_p(e) = io \\ urgent_p(e) & otherwise. \end{cases}$$

$\trianglerighteq io \rightarrow p$ is an interruptive process that if event io occurs, the previous term will immediately terminate and then behaves as p. $\Phi(\trianglerighteq io \rightarrow p)$ means that once

$event_p(e)$ equals io, $urgent(e)$ becomes true and the previous term will immediately terminate and then behaves as p.

3 Case study

3.1 The MA scenario

3.1.1 Description

According to the general technical programme of CTCS-3 level train control system in railways for passengers [13], the moving authority scenario is divided into three parts: Arrival-departure Route Scenario, Through-Route Scenario and Block Section Scenario. In this paper, we mainly focus on the Block Section Scenario. As is described in figure 3, the trains are running on the railway line which is made up of several Block Sections. We assume that all the Block Sections have their length and limited speed. Balise are layout on each Block Section in order to adjust the position of the train.

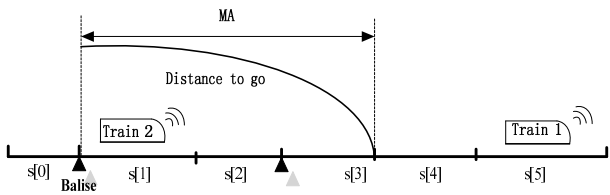

Figure 3: The sketch map of MA scenario.

3.1.2 The interaction between subsystems

The data flow in the Block Section Scenario is shown in figure 4. The state of each Block Section is sent from TCC to RBC. The onboard equipment sends its position to RBC. Three types of Moving Authority (SMA UEM CEM) will be replied to the onboard equipment which contains the state of Block Section, the speed and the route information. A distance to go curve is computed to supervise the safe running of a train.

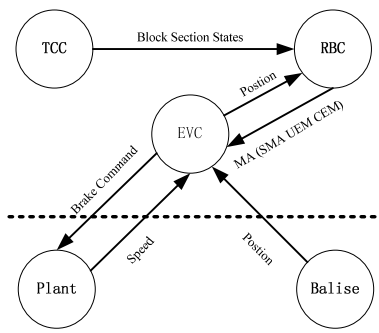

Figure 4: The data flow between subsystems.

3.1.3 The safety requirement

(1) Normal MA supervision
- The position of the train is always less than EOA (End of Authority).
- Once in the ceiling supervision limits mode, the speed of the train is less than the limited speed of Block Section.
- Once in the target supervision limits mode, if the speed exceeds the EBI (Emergency Brake Intervention), the emergency brake command must be trigged immediately.

(2) Short MA supervision
- Once the position of the train exceeds the EOA of SMA, the emergency brake command must be trigged immediately.
- Once the position of the train is not beyond the EOA of SMA, the message of SMA must be ignored.

(3) UEM supervision
- Once having received the UEM, the emergency brake command must be trigged immediately.

(4) CEM supervision
- If the position of train is between the stop point of CEM and minimal brake point, the emergency brake command must be trigged immediately.
- If the position of train is between the minimal brake point and maximal brake point, the service brake command must be trigged immediately.
- Once the position of the train exceeds the EOA of CEM, the CEM message must be ignored.

3.2 The HCSP model

3.2.1 Model

The HCSP model of the scenario is defined by six parallel processes as follows:

$$Process \triangleq Plant \parallel Driver \parallel EVC \parallel RBC \parallel Balise \parallel TCC$$

The process is as follows:

(1) Plant

The Plant is defined as a recursive process $p\,\mathrm{Proc}$. We assume that the dynamic behavior of train expresses by the continuous process: $< \dot{s} = v, \dot{v} = a >_{(s=0,v=0,a=0)}$, where s is the position, v is the speed and a is the acceleration with both initial value are zero. The train stays in number i Block Section in the beginning. Once it enters or leaves a Block Section, it will send synchronization enter or leave message to TCC.

$$P_0 \triangleq < \dot{s} = v, \dot{v} = a >_{(s=0,v=0,a=0)} \triangleright (ch_{p2e}!v; (s = s_1) \to press!)$$

$$p\,\mathrm{Proc} \triangleq \triangleright (i := 1; enter!i; leave(i-1)!; i = \left[\frac{p}{d}\right]; P_0 \triangleright (ch_{d2p}?a) \triangleright (ch_{e2p}?a; (a = b_e) \to A_{EB}))$$

(2) Driver

The Driver is defined as a recursive process $d\,\mathrm{Proc}$. It periodically receives data (v,v_p) from EVC. After receiving (v,v_p), the driver first selects a random

acceleration in a interval $[-b_e,0]$ or $[-b_e,A]$ depending on the Boolean value of ($v \geq v_p$), and then sends it to Plant.

$$d\,\mathrm{Pr}oc \triangleq \triangleright\,(ch_{e2d}?(v,v_p);(v \geq v_p) \rightarrow (\exists a \in [-b_e,0].ch_{d2p}!a);(v \leq v_p) \rightarrow (\exists a \in [-b_e,A].ch_{d2p}!a))$$

(3) Balise

The Balise process is defined as a sequential process $b\,\mathrm{Pr}oc$. The position of Balise is a constant value d. When it receives a synchronization message press from Plant, it will immediately send its position to EVC.

$$b\,\mathrm{Pr}oc \triangleq s := d; press?; ch_{b2e}!s$$

(4) TCC

The TCC process is also defined as a sequential process $t\,\mathrm{Pr}oc$. When TCC receives synchronization message enter or leave from Plant, it will change the state of the related Block Section, and send all its updated states to EVC.

$$t\,\mathrm{Pr}oc \triangleq s[0] := 0; s[1] := 0; s[2] := 0; s[3] := 0; s[4] := 0; s[5] := 1; \cdots; s[14] = 0; B := 0;$$
$$\triangleright\,(enter?i \rightarrow (s[i] := 1); leave?j \rightarrow (s[j] := 0); B := 1; (B = 1) \rightarrow (ch_{t2r}!s; B := 0));$$

(5) RBC

The RBC process is defined as a recursive process $r\,\mathrm{Pr}oc$.When the RBC process receives *pos* and *s*, it then computes the value $eoa := (i - \left[\frac{p}{d}\right]) \cdot d - p$, and assigns the max limited speed $vr := v0$. The RBC process sends the message (eoa, vr) to EVC.

$$r\,\mathrm{Pr}oc \triangleq m[0] := 0; m[1] := 0; m[2] := 0; m[3] := 0; m[4] := 0; m[5] := 0; \cdots m[14] = 0; p := 0; eoa := 0$$
$$; vr := 0; i := 5; v0 := 0.03; d := 1; neoa = 10; A := 0; B := 0; C := 0; \triangleright\,(ch_{e2r}?pos; p := pos; ch_{t2r}?s$$
$$; m := s; A := 1;\ \ B := 1; (A = 1) \wedge (B = 1) \rightarrow (eoa := (i - \left[\frac{p}{d}\right]) \cdot d - p; vr := v0; C := 1; (C = 1) \rightarrow$$
$$(ch_{r2e}!(eoa, vr); C := 0; A := 0; B := 0); (v0 = 0.05) \rightarrow A_{SMA}; (s = 4) \rightarrow A_{CEM}; (s = 5) \rightarrow A_{UEM})$$

(6) EVC

The EVC process is defined as a recursive sequential process $e\,\mathrm{Pr}oc$. It describes as: When it receives message *v* from Plant, it will update its position *pos*:=*pos*+*vT*, which may be interrupted by a more priority message *S* from Balise. When it receives the MA message (*eoa,vr*) from RBC, it will compute the distance between *eoa* and de, the target speed and the permit speed.

$$e\,\mathrm{Pr}oc \triangleq pos := 0; v := 0; eb := -be; Ve := 0; de := 0; Vp := 0; a := 0; T := 1; Vq := 15; \triangleright\,(ch_{p2e}?v; (pos := pos + vT)$$
$$\trianglerighteq [(ch_{b2e}?s; pos := s)]; ch_{r2e}?(eoa, vr); de := eoa - v^2 / 2be - T \cdot v; Ve := \sqrt{2be(eoa - de)}; Vp := Ve - v \cdot T;$$
$$((v > (vr + Vq)) \vee ((v > Ve) \wedge (pos < eoa) \wedge (pos > de)) \rightarrow (ch_{e2p}!eb); ch_{e2d}!(v, Vp); ch_{r2e}?sma$$
$$\rightarrow A_{PSMA}; (ch_{r2e}?cem \vee ch_{r2e}?uem) \rightarrow A_{EM})$$

3.3 Model transition and verification

3.3.1 Model transition

According to the former transition rules, the continuous and discrete variables are unchanged whereas the definition of the channel will be renamed. Using the above transition rules in section 2, we transit the HCSP model to HA model as in Table 1.

Table 1: Transition rules.

HCSP	HA
chp2e	v
che2p	eb
chd2p	ad
che2d	v,vp
cht2r	s[0],s[1],s[2], s[3],s[4],s[5]……s[14]
chb2e	S
chr2e	eoa,vr
che2r	pos
enteri	Enteri
leave(i-1)	Leave(i-1)
press	Press

Then we put the HA model into PHAVer. For example, the input language of Plant automaton is described as:

```
automaton plant
contr_var : i,a,v,s,t;
input_var : ad,eb;
synclabs : tau,Press,Enteri,Leavei,sendV,receiveU,EB;
loc normalRun:
        while a>=bs & a<=A wait {s'==v & v'==a & t'==1}
        when ture sync EB do {a'==eb} goto EmergencyBrake;
        when s==d sync Press  goto normalRun;
        when t==T sync sendV  do {t'==0} goto normalRun;
        when ture sync Enteri goto normalRun;
        when ture sync Leavei goto normalRun;
        when ture sync receiveU do {a'==ad} goto normalRun;
        when ture sync tau goto normalRun;
loc EmergencyBrake;
        while a==-be & v>0 wait {s'=v & v'=-be & t'=1}
        when v==0 sync tau goto stop;
        when ture sync goto EmergencyBrake;
loc stop;
        while true wait {true}
        when ture sync tau goto stop;
```

3.3.2 Verification

According to the reachability analysis strategies in PHAVer, we give the following forbidden states:

(1) Normal MA supervision

forbidden1=sys.{$~$~$~normal~$~$ & v>vr+Vq}

forbidden2=sys.{$~$~$~normal~$~$ & v>vr+Vq}

forbidden3=sys.{$~$~$~normal~$~$& v>Ve}

(2) Short MA supervision

forbidden4=sys.{$~$~$~normal~$~$ & pos>neoa-de}

(3) CEM supervision

forbidden5=sys.{$~$~$~normal~$~$ & pos>neoa-de}

(4) UEM supervision

forbidden6=sys.{$~$~$~normal~$~$ & uem}

Based on the simulation and verification, all of the forbidden states are not reachable. It means that HCSP model of Block Section Scenario meets the safety requirement.

4 Conclusion

The high speed train control system is a typical hybrid system, in which it not only contains the continuous evolution process (train position and speed), but also the discrete event between subsystems. According to the hybrid characteristics of a high speed train control system, a formal modeling and verification method is introduced. The behavior of a train control system is described by HCSP. The HCSP models are transited into HA models by introducing some transition rules in order to verify the hybrid property of a train control system. The HA models are automatically verified by a model checking tool PHAVer. The Moving Authority Scenario is taken as a case study and a Movement Authority Scenario HCSP model is built. The Movement Authority Scenario HCSP model is transited into HA model by the above transition rules. Different MA supervision states which must not be reachable are verified using PHAVer. Based on the simulation and analysis of the Movement Authority Scenario in a high speed train control system specification, the method is proven to be validated.

Acknowledgements

The research of the work reported here were supported by the project of National High-tech R&D Program of China: safety certification and assessment technology of a high-speed railway signal system (2012AA112801); the Fundamental Research Funds for the Central Universities: Research of Formal Design and Development Method in a High-Speed Train Control System (2012JBM024); the State Key Laboratory of Rail Traffic Control and Safety (Contract No. RCS2011K010), Beijing Jiaotong University.

References

[1] XU Tianhua. Research of Probabilistic Model Checking and Its Application in the Train Control Systems [D]. Beijing: Beijing Jiaotong University, 2008. In Chinese.
[2] Kirsten Berkenkotter Stefan Bisanz Ulrich Hannemann Jan Peleska. The HybridUML profile for UML 2.0[J]. International Journal on Software Tools for Technology (2006) 8(2): 167–176.
[3] Kirsten Berkenkotter Stefan Bisanz Ulrich Hannemann Jan Peleska. HybridUML profile for UML2.0. SVERTS Workshop at the UML 2003 Conference, October 2003. http://www.verimag. imag.fr/EVENTS/2003/ SVERTS/.
[4] Paul Ziemann and Martin Gogolla. Validating OCL specifications with the USE tool - an example based on the BART case study. In Thomas Arts and Wan Fokkink, editors, Proc. 8th Int. Workshop Formal Methods for Industrial Critical Systems (FMICS'2003), volume 80 of ENTCS. Elsevier, 2003.
[5] André Platzer. A Temporal Dynamic Logic for Verifying Hybrid System Invariants [J]. Logical Foundations of Computer Science, Springer Berlin / Heidelberg 2007, pp. 457-471.
[6] André Platzer, Jan-David Quesel. Logical Verification and Systematic Parametric Analysis in Train Control [J]. Lecture Notes in Computer Science [J], Springer Berlin 2008, pp. 646-649.
[7] André Platzer, Jan-David Quesel. European Train Control System: A Case Study in Formal Verification [J]. Lecture Notes in Computer Science [J], Springer Berlin 2009, pp. 246-265.
[8] Thomas A. Henzinger, The Theory of Hybrid Automata Logic in Computer Science [A]. LICS'96, Proceedings of Eleventh Annual IEEE Symposium on [C]. 1996. 278 –292.
[9] Werner Damm, Alfred Mikschl, Jens Oehlerking, Ernst-Rüdiger Olderog, Jun Pang, André Platzer, Marc Segelken, and BorisWirtz. Automating Verification of Cooperation, Control, and Design in Traffic Applications [J]. Springer Berlin 2007, pp. 115-169.
[10] Zhou Chaochen, Wang Ji and Anders P. Ravn A Formal Description of Hybrid Systems [C]. in the Proc of the DIMACS/SYCON workshop on Hybrid systems III: verification and control: verification and control Springer-Verlag New York, Inc. Secaucus, NJ, USA, 1996.
[11] Goran Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In Manfred Morari and Lothar Thiele, editors, Hybrid Systems: Computation and Control (HSCC'05), Mar. 9–11, 2005, Zürich, CH, volume 2289 of LNCS. Springer, 2005.
[12] LV Jidong. Hierarchical Formal Modeling and Verification Train Control System [D]. Beijing：Beijing Jiaotong University, 2011. In Chinese.
[13] Shuguang Zhang, The general technical Programme of CTCS-3 level train control system in railways for passengers [M], China Railway Publishing House, 2008. In Chinese.