Hybrid MARTE statecharts

Jing Liu, Ziwei Liu, Jifeng He, Frédéric Mallet, Zuohua Ding

Hybrid MARTE Statecharts

Jing Liu, Ziwei Liu, Jifeng He[1], Frédéric Mallet[2], Zuohua Ding[3]

1 Shanghai Key Laboratory of Trustworthy Computing, East China Normal University, Shanghai, China
2 Université Nice Sophia Antipolis, Aoste (INRIA/I3S), Sophia Antipolis, France
3 Center of Math Computing and Software Engineering, Zhejiang Sci-Tech University, Hangzhou, Zhejiang, China

Abstract The specification of Modeling and Analysis of Real-time and Embedded Systems (MARTE) is an extension of the UML in the domain of real-time and embedded systems. Even though MARTE Time model offers a support to describe both discrete and dense clocks, the biggest effort has been put so far on the specification and analysis of discrete MARTE models. To address hybrid real-time and embedded systems, we propose to extend statecharts using both MARTE and the theory of hybrid automata. We call this extension Hybrid MARTE statecharts. It provides an improvement over the hybrid automata in that: the logical time variables and the chronometric time variables are unified. The formal syntax and semantics of Hybrid MARTE statecharts are given based on labeled transition systems. As a case study, we model the behavior of a Train Control System with Hybrid MARTE statecharts to demonstrate the benefit.
Keywords UML; MARTE; Hybrid automata; Hybrid MARTE statechart; Train Control System

1 Introduction

The Unified Modeling Language (UML) [1] is a de facto industry standard, and is evolving under the direction of the Object Management Group (OMG). UML extensions, so-called profiles, are increasingly accepted as light-weight solutions to customize UML tools in order to address domain-specific challenges [2]. The ever increasing complexity of Real-Time and Embedded Systems (RTES) requires dedicated solutions that help in their modeling, verification and design space exploration [3]. The UML Profile for Modeling and Analysis of Real-time and Embedded Systems (MARTE) [4] addresses this issue in the context of model-driven development (MDD) with UML. MARTE has been adopted by the OMG in November 2009 and supersedes the existing UML profile for schedulability, performance and time [5].

MARTE provides constructs to support both the modeling and the analysis of RTES. It introduces fundamental concepts for model-based description of RTES by providing an expressive model of time, a model of resources, a support to express non-functional properties and a support to conduct various kinds of analyses. UML notion of Time is very simplistic, however, MARTE provides a rich and expressive time model [6]. MARTE gives a detailed time structure, which is ready to cover both logical/chronometric discrete/dense time. Multiform logical time has been made popular by Synchronous languages [7]. It has been introduced in MARTE [8] to cover a wide range of reactive systems including transportation systems. Multiform time uses independent time bases to measure time flows of events. Each time base may use a different unit (second, meter, angle). Naturally, to support multiform time modeling, the time-related concepts and the behavioral diagrams in UML are extended. MARTE Time model contains main time entities, such as clock, clocktype and clock constraints. The Clock constraint specification language (CCSL), defined in a non-normative annex of MARTE, provides a concrete syntax to define multiform logical time constraints and support time analysis [9, 10]. Most of the works on MARTE and CCSL focus on discrete models and the dense aspect of clocks remains unexploited so far.

A hybrid system is a dynamic system, which is composed of the interactions of continuous subsystems and discrete subsystems. Discrete systems are systems with countable number of states. Such systems deal with discrete variable whose definition domain is discrete. Continuous systems, however, deal with continuously varying quantities and uncountable sets. Dynamic characteristic of continuous subsystems changes over time, and the dynamic evolution of discrete subsystems is activated by events [11]. Researches on hybrid systems have a broad engineering background. Most of the complex industrial control systems are composed of dynamic evolutions of the physical layer, which is described by continuous variables, and the high-level coordination of the optimization process, which is characterized by symbolic operations and discrete control decisions. Many industrial control systems have been modeled as hybrid systems such as robot systems, automobiles, air-

Full text available from the Publisher: https://link.springer.com/article/10.1007/s11704-012-1301-1

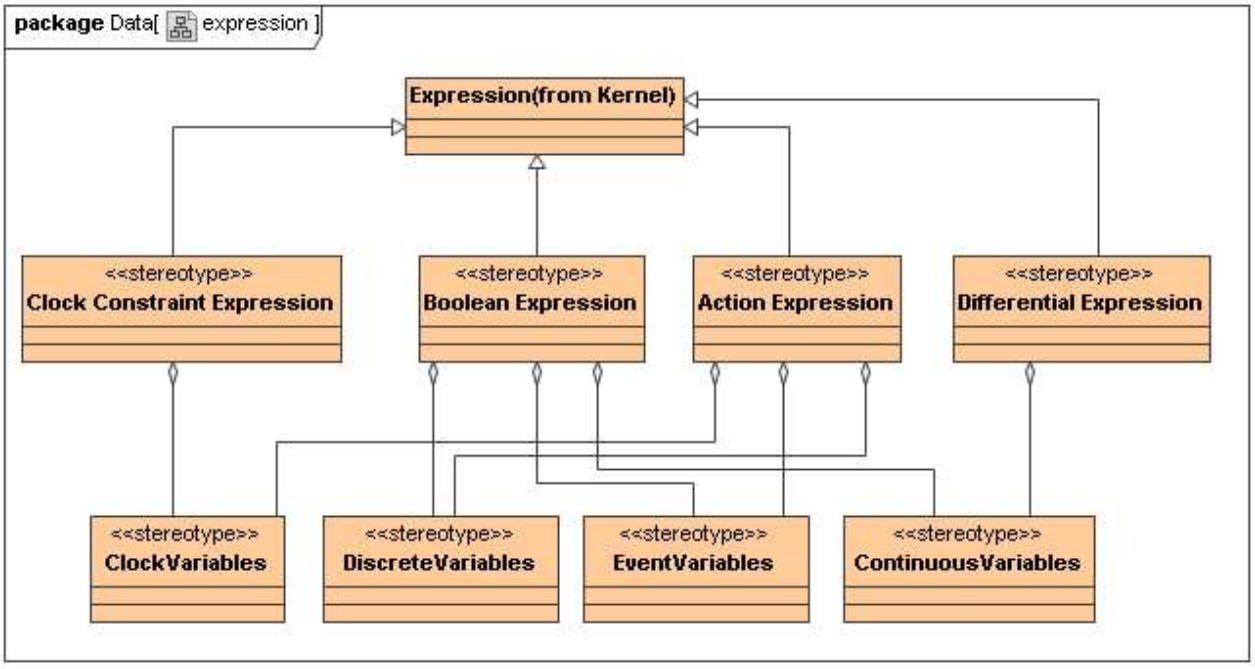

Fig. 1 Four extensions of UML expressions dedicated to Hybrid MARTE Statecharts

crafts, and chemical process control systems [12]. Hybrid automata can be used as a general framework of formal description and algorithm analysis of hybrid systems. Hybrid automata are extensions of timed automata, where the continuous dynamic behavior in the system is described by a set of differential equations. It has been successfully used to solve some practical verification, controllability and optimization problems [13].

MARTE Time model was designed to deal with both discrete and dense time. It provides primitive constructs to discretize dense clocks, so-called chronometric clocks. However, the dense models were introduced to deal with rational numbers (dense but countable) rather than proper continuous variables. An extension of MARTE is therefore required to address control-based hybrid system design and provide additional operators to deal with dense clocks. Proposing such an extension is the main contribution of this paper.

Our extension is based on Statecharts. Statecharts were introduced by D. Harel in 1987, and provide compact and expressive visual formalisms for reactive systems. They have been widely used as a modeling tool and have largely inspired the definition of the UML State Machines. They are fundamental to model the dynamic behavior of objects [14]. Since statecharts is an extension of the finite state automata model, there is a natural mapping between statecharts and automata [15]. This paper mainly focuses on the extension of UML State Machines with MARTE and hybrid automata concepts. Other UML elements, such as classifiers, activities or interactions, are out of the scope of this paper.

Our proposition is to extend MARTE statecharts with hybrid automata. We mainly focus on providing a formal definition for the syntax and semantics of the Hybrid MARTE statecharts. By importing hybrid automata into MARTE statecharts, we get a unified time model: Hybrid MARTE statecharts, where two different time concepts (continuous time and multiform logical time) are used jointly. Moreover, we have also developed a tool for system simulation. With the simulation results, we can use our Hybrid MARTE statechart model in system design.

This paper is organized as follows. Section 1 presents a brief summary of Hybrid MARTE statecharts. Section 2 describes the formal models of Hybrid MARTE statecharts: expressions (Section 2.1), formal syntax (Section 2.2), and formal semantics (Section 2.3). Section 3 gives an application example of an Automatic Train Protection system (ATP) that prevents trains from over-speeding. This illustration example shows a specific usage of Hybrid MARTE statecharts in the field of Train Control

System. Simulation results are also presented in Section 3.

2 Hybrid MARTE Statecharts

Hybrid MARTE statecharts is an extension of MARTE statecharts based on hybrid automata. Through importing Hybrid automata, a set of differential equations are introduced to MARTE statecharts to represent the continuous dynamic behavior of the system.

Statecharts In the Hybrid MARTE statecharts, we use modes to describe the behavior. By augmenting state descriptions using *Invariant* and *Activity*, Hybrid MARTE statecharts extend the variables of MARTE statecharts. *Invariant* is a formal statement to describe the properties unchanged in a mode. *Activity* is used to describe the time-continuous variable evolutions. *Activity*, which is the conjunction of several *Differential Expressions*, takes place while the system stays in the respective mode. Like hybrid automata, variation of discrete variables in Hybrid MARTE statecharts is specified by *Actions* labeled on transitions. Transitions in Hybrid MARTE statecharts can be triggered by signal events. Within the transition, actions consisting of variable assignments and signal generations may take place. The semantics of Hybrid MARTE statecharts is defined based on labeled transition systems. According to the semantics, there are two kinds of transitions: *Edge* and *Evolution*. The former one specifies when the execution flow passes from one mode to another, and the latter one represents the transitions caused by the flowing of time.

Expressions Hybrid MARTE statecharts use typed variables. In UML, we can only use Integer, String, and Boolean as basic types. To describe continuous changes of variables, we extend the data type to Discrete variables, Event variables, Continuous variables, and Clock variables. Accordingly, there are four kinds of expressions: *Boolean Expression*, *Action Expression*, *Clock Constraint Expression* and *Differential Expression*.

Constraints UML constraints can be used to refine and constrain the semantics of UML elements. Constraints are specified as textual specifications in an external language to express conditions and restrictions on model elements [16]. In this paper we use MARTE clock constraints as well as plain UML constraints. The set of primary/kernel clock constraints that can be used in Hybrid MARTE statecharts is defined in the following subsection in Definition 2.3.

Clocks MARTE Time model improves the UML time model to a large extent. New Time Structure is defined to model both discrete/dense and chronometric/logical time. We use an abstract definition of clocks, which includes two attributes of a clock: the set of time instants and the labeling function. We abstract clocks as infinite sequences of time instants.

Communication mechanism Communications in Hybrid MARTE statecharts are based on shared variables and signals. Shared communication variables are represented as variable-ports. Signal-communications are another way for components to communicate. Similarly signals are visualized as signal ports. In Hybrid MARTE statecharts, signals are carried by SignalEvents, associated with the triggers on transitions from one mode to another.

2.1 Expressions

Like UML statecharts, Hybrid MARTE statecharts allow four kinds of expressions: *Boolean Expression*, *Action Expression*, *Clock Constraint Expression* and *Differential Expression*. *Boolean expressions* can be used to describe guard conditions, invariants and activities. *Action expressions* are used when defining events and actions in Hybrid MARTE statecharts. *Clock Constraint Expressions* are used to define clock-related expressions. *Differential Expressions* are capable to describe activities of continuous variables. In our formalization, every expression occurring in Hybrid MARTE statecharts is considered as a predicate over variables with specific types. Thus, there are four kinds of variables in Hybrid MARTE statecharts:

1. discrete variables with discrete values independent on time, e.g. $x = \{1, 2, 3\}$.
2. event variables which have a boolean value specifying whether the corresponding event occurs or not, e.g. θ, η
3. continuous variables which can be represented as a continuous function dependent on time (with its initial condition), e.g. $F \triangleq \begin{cases} x = x(t) \\ x(0) = 0 \end{cases}$
4. clock variables with a special clock type, e.g. c, d

The syntax of *Boolean Expression* and *Differential Expression* in Hybrid MARTE statecharts is defined as follows.

Definition 2.1 (*Boolean Expression*)

Let x be a variable from the four kinds above.

$\phi ::= \eta \mid A(x) \mid F(x, \frac{dx}{dt}) \mid$

$\neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \Rightarrow \phi_2 \mid \forall x \cdot \phi(x) \mid \exists x \cdot \phi(x)$

η is an event expression with an event variable. The expression is satisfied when the current evaluation of the event variable returns TRUE.

$A(x)$ is an algebraic expression in terms of either an equation or an inequality over algebraic objects. The expression is satisfied when the evaluation of every discrete variable makes the equation or inequality hold. For an algebraic expression $A(x)$, we say an evaluation $v(x)$ satisfies $A(x)$, denoted as $v(x) \models A(x)$ iff $A[v(x)/x] =$ TRUE.

E.g. Providing $A \triangleq x = y + z$, an assignment $v_1(\langle x, y, z\rangle) = \langle 1, 1, 1\rangle$ while $v_2(\langle x, y, z\rangle) = \langle 2, 1, 1\rangle$. Obviously we can obtain the conclusion that $v_1 \not\models A$ and $v_2 \models A$.

$F(x, \frac{dx}{dt})$ is a differential expression with initial condition describing the properties involving continuous variables (x) dependent on time. The semantics of *Differential Expressions* is: $w(t)$ is a continuous variable, the differential expression is satisfied when $w(t)$ is a solution of $F(x, \frac{dx}{dt}) = 0$, and acts on the domain of continuously differentiable functions [17]. For a differential expression F with x being a continuous variable, we say an evaluation $v(x)$ satisfies the expression, denoted as $v \models F$ iff $v(x) = w(t)$ with $t \in D^0$ is the solution of the equation F in the domain D^0.

E.g. Providing $F \triangleq \begin{cases} \dot{x}(t) = 1 + x^2(t) \\ x(0) = 0 \end{cases}$

By solving the above differential equation, we can find $tan(t)$ is a solution. Hence, we can say the assignment $v(x) = tan(t)$ satisfies the differential expression i.e., $v \models F$.

Definition 2.2 (Clock)

A Clock c *can be defined as follows:*
$c ::= (I, \rho)$*, where*

- I *is a set of time instants.*
- ρ *is a sampling function,* $\rho : I \to \mathbb{R}_{\geq 0}$ *maps every time instant in* I *to a non-negative real number.*

A time instant represents an observation point of a clock, either physical or logical, reflecting specific events dependent on time. We call ρ the sampling function that discretizes the continuous time to a series of observation points on demand. Different clocks have different sampling functions. Through the sampling function ρ, a clock c can be abstracted to $\vec{c}$, such that $\rho(c.I) = \vec{c}$, which is an unlimited sequence of time instants. The sequence is a discrete measurement of time and there is a default sequence that contains all the sequences of different clocks in the system.

For every Hybrid MARTE statechart S, we have a default physical clock $\gamma = (I, \rho)$, $\vec{\gamma} = \rho(\gamma.I) = \mathbb{R}_{\geq 0}$. For any clock $c \in S$, it is obviously that $\vec{c} \subseteq \vec{\gamma}$. Since $\rho(\gamma.I) = \mathbb{R}_{\geq 0}$, there is a total order $\leq$ over γ as in $\mathbb{R}$.

Hybrid MARTE statecharts allow to define multiform clocks. These clocks have different presentations, e.g. a clock may use ms as its unit while another uses an angle degree unit. When we unify them in the form of the above clock structure, we can compare them in the same framework by defining a sampling function for each clock.

When designing the model with Hybrid MARTE statecharts, a clock is an instance of a clockType, the unitType that is an enumeration of different units, and the relationships between them. Figure 4 gives a small example after all the definitions are introduced.

Definition 2.3 (*Clock Constraint Expression*)

Let c, d be two clocks,

$ClkC ::= c \subseteq d \mid c \subset d \mid c \sim d \mid c \bowtie d$

$ClkC$ is called a clock constraint. It specifies the relationship among different clocks defined in a system [6]. At different time instants, $ClkC$ can be satisfied or not satisfied.

- $c \subseteq d$ states that clock c precedes clock d, which means that every observation point of clock c is earlier than or equal to the related observation point in d.
  $c \subseteq d =_{df} \forall k \in \mathbb{N}^+ \bullet \vec{C}_k \leq \vec{D}_k$

- $c \subset d$ means that clock c strictly precedes clock d, which means that every observation point of clock c is strictly earlier than the related observation point in d.
  $c \subset d =_{df} \forall k \in \mathbb{N}^+ \bullet \vec{C}_k < \vec{D}_k$

- $c \sim d$ represents that clock c and d alternates with each other.
  $c \sim d =_{df} \forall k \in \mathbb{N}^+ \bullet \vec{C}_k < \vec{D}_k < \vec{C}_{k+1}$

- $c \bowtie d$ represents that the clock c is synchronized with clock d. It evolves at the same rate.
  $c \bowtie d =_{df} \forall k \in \mathbb{N}^+ \bullet \vec{C}_k < \vec{D}_{k+1} \wedge \vec{D}_k < \vec{C}_{k+1}$

Based on the clock structure, the current reading of the clock c can be represented by the first entry of the time instant sequence $\vec{c}$ (denoted as $c\sharp$). A clock tick can be represented as the following clock transformation $c' \triangleq tick(c)$ where $c\sharp$ is removed.

$c\sharp \triangleq first(\vec{c})$
$c.I = c\sharp \cdot tick(c).(c.I)$

As a result, $tick(c).\rho = (\vec{c} \backslash \{c\sharp\}) \lhd c.\rho$

The semantics for logical connectives $\wedge, \vee, \Rightarrow, \forall, \exists$ is defined like the first order language. For example, let $P \triangleq \phi_1 \wedge \phi_2$, an evaluation $v \models P$ iff $v \models \phi_1$ and $v \models \phi_2$.

Action Expression sets the value of system states. The syntax of *Action expression* is as follows.

Definition 2.4 (Syntax of *Action Expression*)

$Act ::= !\vartheta \mid x := e \mid c := \vec{\gamma}.i \mid Act \& Act$

- $!\vartheta$ are signal actions. It means emitting signal ϑ.
- $x := e$ assigns the value of expression e to the mode variable x if they have the same type.
- $c := \vec{\gamma}.i$ sets the clock c to one of default physical clock's instant $\vec{\gamma}.i$.

In this paper, the semantics of *Action Expression* can be represented as a relation over two system modes.

Definition 2.5 (Semantics of *Action Expression*)

Let ϑ, x, c be variables having the value before the action and ϑ', x', c' be variables having the value after the action.

$!\vartheta =_{df} \vartheta' = True$

$x := e =_{df} Type(e) = Type(x) \wedge x' = e$

$c := \vec{\gamma}.i =_{df} c' = tail^{i-1}(\vec{\gamma})$

$Act_1 \& Act_2 =_{df} Act_1 \wedge Act_2$

$!\vartheta$ means that after the action, the value of ϑ is true; $x := e$ suggests that if x and e have the same type, the value of x will be e after the action; $c := \vec{\gamma}.i$ gives the tail of $\vec{\gamma}.i$ to c from the time instant of i; $Act_1 \& Act_2$ means after this action, both the effect of Act_1 and the effect of Act_2 will be satisfied.

2.2 Formal model of Hybrid MARTE Statecharts

A Hybrid MARTE statechart is used to describe the behavior of a system. The behavior of a system is abstracted to the transitions between various states. The states are classified into several modes satisfying specific constraints. The mode is

one of the elementary notion of the statechart. The system execution is considered as traveling among these modes. Time flows only exist in modes, not in any transitions. Hybrid MARTE statecharts can represent hierarchical models by using the function *children*. Function *children* specifies a mode that has several direct submodes.

Definition 2.6 (Hybrid MARTE statechart)

A Hybrid MARTE statechart (HMS) is a tuple,
$HMS = (m, Var, Sub, E, init, grd, evt, act, avt, inv, elf)$
where

- m *is a unique identifier. Given a statechart, we use a set* Modes *to collect all the mode and submode names.*
- *A set of variables* Var *available in this statechart. It includes a set of discrete variables* dVar*, a set of continuous variables* cVar*, a set of clock variables* ckVar *and a set of signals* S*. The variables we mention in this paper are considered to be global variables.*
- Sub *is a set of names of its immediate submodes. This attribute reflects the hierarchical structure of the statechart. Given a statechart, we define a global function*

$$Children : Modes \rightarrow 2^{Modes}$$

  to represent the relationship between a mode and its immediate submodes. The function subjects to the constraints of the tree structure and every mode has one father or none. If a mode has no children, it is called an atomic mode. Otherwise, it is called a composite mode.
- E *is a set of edges connecting its submodes in* Sub*. The labels marked on edges specify the trigger events, guard conditions and actions of a transition behavior.*
- Init *specifies the initial requirements of the mode. It restricts the variables to the appropriate values which guarantee the valid execution. The execution flow is permitted to enter the mode if its initial requirement is satisfied.*
- *grd* : E $\rightarrow$ GUARD *maps each edge with its guard conditions.* GUARD *is a set of guard conditions, which are Boolean Expressions. If and Only if guard conditions are satisfied, the transition can take place.*
- *evt* : E $\rightarrow$ EVENT *maps each edge with its events. It describes the trigger events of transitions, such as signal events.* $EVENT$ *is a set of events.*
- *act* : E $\rightarrow$ ACTION *maps each edge with its Action Expressions.* Act *describes the effects of transitions. It is an atomic operation, and its operation time is zero.* ACTION *is a set of Action Expressions.*
- *avt* : E $\rightarrow$ ACTIVITY *is called the activity of a mode which is the conjunction of several Differential Expressions. The avt specifies the varying patterns of continuous variables dependent on time within the mode. It is the main constraint of the continuous variables in a mode.*
- *inv* : m $\rightarrow$ INVARIANT. INVARIANT *is the set of invariant conditions. Inv specifies the invariant condition which must hold in this mode. The invariant represents the global constraints to the variables. Whenever the invariant condition is violated, the execution flow must exit this mode. If there is no valid transition for the mode, i.e. the* iniv *of target mode is not satisfied, system will transfer to a halt mode, which will be discussed later.*
- *elf* : m $\rightarrow$ ELF. ELF *is a set of guard conditions. Elf specifies a special guard condition. When elf is satisfied, it will trigger a self transition of this mode, and within this transition an event can be generated. Elf is used to describe the effects that the environment brings. In UML 2.0, communication with environment is through ports. In Hybrid MARTE statecharts, we want to model this in the statechart directly. That is why we define an elf notion.*

In general, we require init $\Rightarrow$ inv. The initial requirement init(m) must satisfy the invariant inv(m), so that the execution flow can enter this mode.

The syntax of Hybrid MARTE statecharts draws strongly upon the syntax of statecharts. Hybrid MARTE statecharts associate with every mode a continuous dynamics whose solutions are activities.

2.3 Safe Semantics of Hybrid MARTE Statechart

In this paper, the behavior of a mode is formalized as a labeled transition system.

Definition 2.7 (Labeled Transition System)

A labeled transition system S is a tuple $(\text{States}, \text{Labels}, \rightarrow, S_0)$ *where*

1. States *is a set of states.*
2. S_0 *is the set of initial states and* $S_0 \subseteq$ States*.*
3. Labels *is a set of labels which identify the transitions.*
4. $\rightarrow$: States $\times$ Labels $\times$ States *is a ternary relation over* States *specifying the transitions between states.*

In Hybrid MARTE statecharts, a state of a system can be formalized as a structure $(m, v, \vec{c})$ where

1. $m \in$ *Modes* is a mode name. It states that the execution flow is currently in mode m.
2. v : *Var* $\rightarrow$ *Value* is a valuation function for every state variable of the system. *Value* stands for the evaluation set of *Var*. For a signal set $S \subset$ *Var*, when a signal s is generated, we add s into the set S; if signal s is expended, we remove s from the set S.
3. $\vec{c}$: the sequence of the current clock whose reading is converted from the current clock defined within the transition.

Based on mode attributes mentioned before, a valid mode requires every contained state to be a valid state.

Definition 2.8 (Valid States)

Let $q = (m, v, \vec{c})$ be a state, we say it is a valid state if and only if it satisfies the following conditions.

1. *Let* inv(M) *be the invariant condition of mode M over variable x and clock ck, then we require*

   $(m, v, \vec{c}) \in M.States$ *iff*

   $m \in$ *Children*(M)

   $\wedge$ *inv*$(M)[v/x, \vec{c}/ck] \wedge$ *inv*$(m)[v/x, \vec{c}/ck]$

2. *Let* init(M) *be the initial requirement of mode M over variable x and clock ck, then we require*

   if $(m, v, \vec{c}) \in M.S_0$*, then*

   $m \in Children(M) \wedge$ *init*$(M)[v/x, \vec{c}/ck]$

In the following part of this paper, we only consider valid states.

A transition relation $\rightarrow$: *States* $\times$ *Labels* $\times$ *States* is a relation from a source state to a target state with a specific transition. A concrete transition is usually written as $s \xrightarrow{\lambda} s'$ where s and s' are the source state and target state of this transition and λ is a label identifying this transition. The label λ marked on a transition contains the following three attributes.

- An event evt(λ) represents occurrences in time or space. Events are used to trigger transitions. If a transition has no event, then the transition is an unconditional transition. There are two kinds of events, SignalEvent and TimeEvent. SignalEvent can either be used to trigger transitions inside a component or to make two disjoint system components communicate with each other. TimeEvent is an event triggered by the flow of time. In this paper TimeEvent depends on the clock time defined by the user in the Hybrid MARTE statecharts. If the event is omitted, the transition can be triggered at any time.
- A guard condition grd(λ) specifies when the transition is enabled. If the guard attribute is omitted, it means the transition is always enabled to be triggered.
- An Action act(λ) is performed when the transition occurs, changing the values of variables or sending a SignalEvent. It describes the effects of the transition.

A transition $s \xrightarrow{\lambda} s'$ can be triggered if the source state satisfies its guard condition and a trigger event is observed. Whenever a transition is triggered, its action is performed and the current state of the system becomes the target state of the triggered transition.

Definition 2.9 (Transition)

A transition $(m, v, \vec{c}) \xrightarrow{\lambda} (m', v', \vec{c}')$ is valid if and only if the following conditions hold.

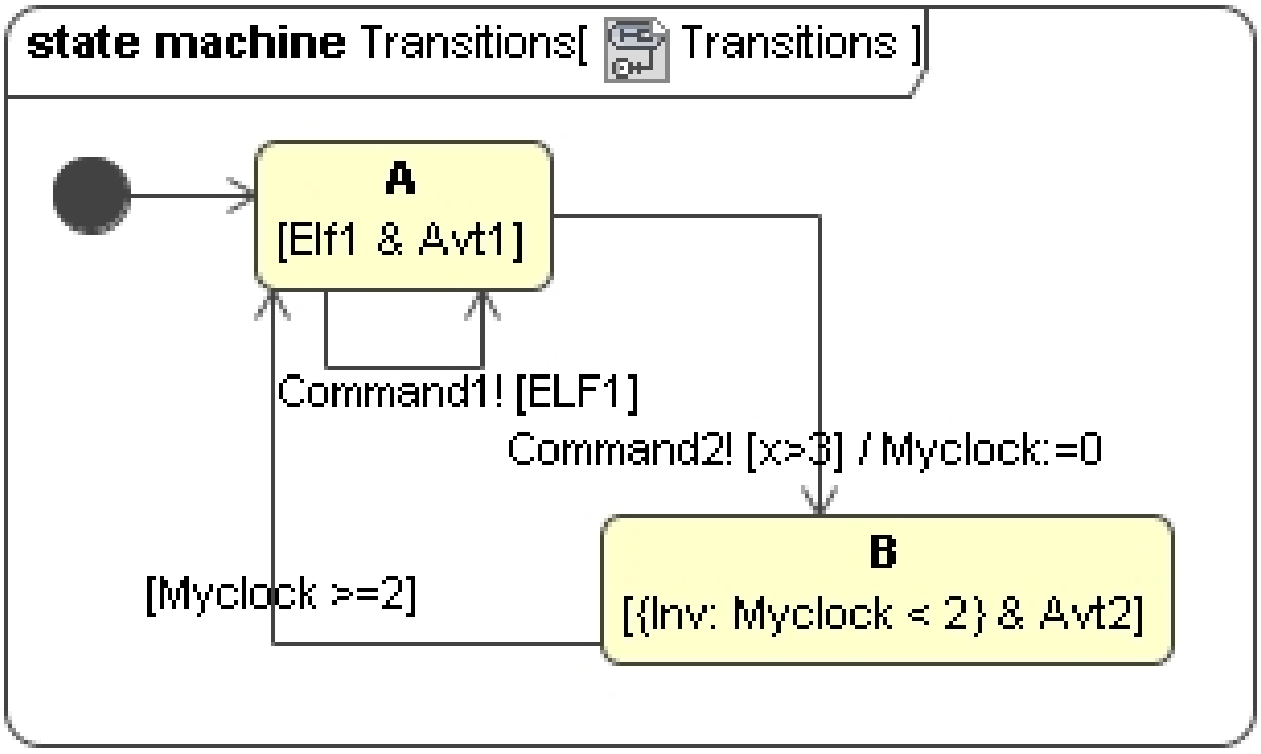

Fig. 2 Example of Transitions

1. $(m, v, \vec{c})$ *and* $(m', v', \vec{c}')$ *are both valid.*
2. $v, \vec{c} \models$ *evt*$(\lambda) \wedge$ *grd*(λ)
3. $v, \vec{c}, v', \vec{c}' \models$ *act*(λ)

There is a default transition for any mode m which takes the system to a halt mode. It can be fired either when:

- None of the normal edges from m can be fired;
- or, the invariant condition is violated.

We define the default transition $dtr = q \times (\epsilon, \neg \text{inv}(m), \epsilon) \times q')$ where $q = m$ and $q' = \perp$.

Transitions can be divided into two kinds: edges and evolutions. Self-transition is a special kind of evolution, and it is used to define a transition triggered by a continuous variable. Whenever a continuous variable changes because time passes, it triggers a continuous self-transition. This is useful to observe continuous changes in the environment. For instance, let us consider an air conditioning system. The specification requires the air conditioner to stop if the room temperature moves below a given temperature. In this case, the continuous variable $temperature$ will trigger a self-transition.

Edges in Hybrid MARTE statecharts can be interpreted as transitions when the execution flow passes from one mode to another.

Definition 2.10 (Edge)

An edge $\xrightarrow{(e,g,a)}$ *from mode* m_1 *to* m_2 *is formalized as the following transition rule*

$$\frac{v, \vec{c} \models e \wedge g}{(m_1, v, \vec{c}) \xrightarrow{(e,g,a)} (m_2, v', \vec{c}')} \qquad a(v, \vec{c}, v', \vec{c}')$$

If the condition $v, \vec{c} \models \text{evt}(\lambda) \wedge \text{grd}(\lambda)$ is true, then the Edge is fired and action $a(v, \vec{c}, v', \vec{c}')$ is executed. The system can remain in a particular mode as long as the mode invariant is satisfied. The mode invariants are related to the transition guards. Contrary to Evolutions, Edges are fired only at discrete time instants. In Fig. 2, the transitions between mode A and mode B are edges. The edge from mode A to mode B is valid as long as
$v, \vec{Myclock} \models$ *Command2*! $\wedge (x > 3)$
and
$v, \vec{Myclock}, v', \vec{Myclock}' \models$ *Myclock* $:= 0$.
The edge from mode B to mode A is valid, as long as
$v, \vec{Myclock} \models$ *Myclock* $>= 2$.

To support hierarchy, if the trigger condition of a transition is satisfied or the invariant condition of the father mode is violated, we require that any state of a submode can also trigger a transition in its father mode. Formally, additional transitions are included in the transition relation.

Definition 2.11 (Transition Closure)

Let mode $m \in Children(M)$*, if* $(M, v, \vec{c}) \xrightarrow{\lambda} (M', v', \vec{c}')$ *then we have* $(m, v, \vec{c}) \xrightarrow{\lambda} (M', v', \vec{c}')$.

The other kind of transitions in Hybrid MARTE statecharts is caused by the flow of time. We call it *Evolution*. An *evolution* allows the value of the continuous variables to evolve with time according to the activity condition of the mode.

Definition 2.12 (Evolution)

Let m *be a mode having a continuous variable* X*, a clock* c *and signal set* S*. An evolution within mode* m *can be divided into three kinds:*

1. *For each nonnegative real* $\delta \in \vec{c} \subset \mathbb{R}_{\geq 0}$*, there is a differentiable function* $f : [0, \delta] \to \mathbb{R}^n$*. Its first derivative* $\dot{f} : [0, \delta] \to \mathbb{R}^n$ *is such that* $f(0) = v$ *and* $f(\delta) = v'$*.* $\forall \varepsilon \in (0, \delta)$*,*
   (a) *inv*$(m)[f(\varepsilon)/X] =$ *TRUE.*
   (b) *avt*$(m)[f(\varepsilon)/X, \dot{f}(\varepsilon)/\dot{X}] =$ *TRUE.*
   (c) *elf*$(m)[f(\varepsilon)/X] =$ *FALSE.*
   An evolution that satisfies the above three conditions is called a basic Evolution.
2. *A self-Transition* $\xrightarrow{(e,g,a)}$ *from mode* m *to* m *is triggered iif the following conditions are satisfied:*
   Let $\delta \in \vec{(c)} \subset \mathbb{R}_{\geq 0}$ *be a nonnegative real,* $\forall \varepsilon \in (0, \delta)$*,*
   (a) *inv*$(m)[f(\varepsilon)/X] =$ *TRUE.*
   (b) *avt*$(m)[f(\varepsilon)/X, \dot{f}(\varepsilon)/\dot{X}] =$ *TRUE.*
   (c) *elf*$(m)[f(\varepsilon)/X] =$ *TRUE.*
   When the transition fires, a signal s *will be added to the set* $S \in v'$*.*
3. *If the following conditions are satisfied, the system is be taken into a halt mode.*
   Let $\delta \in \vec{(c)} \subset \mathbb{R}_{\geq 0}$ *be a nonnegative real,* $\forall \varepsilon \in (0, \delta)$*,*
   (a) *inv*$(m)[f(\varepsilon)/X] =$ *FALSE.*
   (b) *avt*$(m)[f(\varepsilon)/X, \dot{f}(\varepsilon)/\dot{X}] =$ *TRUE.*
   (c) $v, \vec{c} \nvDash e \wedge g$*.*

The continuous variable in a mode varies when time passes. Once the variable satisfies the elf of a mode m, self-transition $(m, v, \vec{c}) \xrightarrow{(e,elf,a)} (m, v', \vec{c}')$ is triggered and a a signal event is generated. If elf is not satisfied, the basic evolution is fired without any action. When the invariants are violated and none of the normal edges from m can be fired, the default transition

dtr takes the system into a halt mode. Edges are fired at discrete time instants, whereas evolutions are fired in the time interval bounded by two discrete time instants.

In Fig.2, Elf1 is the elf of mode A and Avt1 is the activity of mode A. Variables in mode A change according to Avt1. When the variables satisfy Elf1, a self-transition from mode A to itself is triggered and signal Command1! is generated. Variables in mode A evolve with time according to Avt1. That constructs an evolution.

2.4 Live Semantics of Hybrid MARTE Statechart

A liveness property P stipulates that a good thing g eventually happens. If g does not occur within a finite sequence α, then α can be extended to an infinite sequence in which g eventually occurs [18]. When we consider the infinite behavior of Hybrid MARTE statecharts, sometimes we are interested only in whether the transition sequences are convergent or not. The divergence of time is a liveness assumption, and it is the only liveness assumption we need to consider [19, 20]. If the trace of modes transitions cannot prevent time from diverging, it is nonzeno. Clearly, only nonzeno designs of real-time systems can be realized. Thus we need to define live semantics of Hybrid MARTE statecharts and clarify how to justify a nonzeno Hybrid MARTE statechart.

Definition 2.13 (Trajectory)

Consider a labeled transition system S, $(m, v, \vec{c})$ is a state of S and A is a set of labels (denoted λ_i).

A $(m_0, v_0, \vec{c}_0)$-rooted trajectory of S is a finite or infinite sequence of pairs $< \lambda_i, (m_i, v_i, \vec{c}_i) >_{i>=1}$ of labels $\lambda_i \in A$ and states $(m_i, v_i, \vec{c}_i) \in Q$ such that $(m_{i-1}, v_{i-1}, \vec{c}_{i-1}) \xrightarrow{\lambda_i} (m_i, v_i, \vec{c}_i)$ for all $i \geq 1$.

If $(m_0, v_0, \vec{c}_0)$ is an initial state of S, then
$< \lambda_i, (m_i, v_i, \vec{c}_i) >_{i>=1}$ is an initialized trajectory of S.

Definition 2.14 (Live transition systems)

A live transition system (S, L) is a pair consisting of a labeled transition system S and a set L of infinite initialized trajectories of S.

The set of infinite initialized trajectories is machine-closed for S if every finite initialized trajectory of S is a prefix of some trajectory in L. If (S, L) is a live transition system, and $< \lambda_i, (m_i, v_i, \vec{c}_i) >_{i>=1}$ is either a finite initialized trajectory of S or a trajectory in L, then the corresponding sequence $< \lambda_i, \vec{c}_i >_{i>=1}$ of labels is called a (finite or infinite) trace of (S, L).

For each Hybrid MARTE statechart HMS, its semantics can be formalized as a labeled transition system S. Two kinds of transitions have been defined:

1. For Edges, an event $e \in \Sigma$, the duration of $(m, v, \vec{c}) \rightarrow (m', v', \vec{c}')$ is 0.
   duration $= \vec{c}' - \vec{c} = 0$
2. For Evolutions, a real $\delta \in R_{\geq 0}$, the duration of $(m, v, \vec{c}) \rightarrow (m', v', \vec{c}')$ is δ.
   duration $= \vec{c}' - \vec{c} = \delta$

Definition 2.15 (Trace semantics)

An infinite trajectory $< \lambda_i, (m_i, v_i, \vec{c}_i) >_{i>=1}$ of the labeled transition system S diverges if the infinite sum $\Sigma_{i \geq 1} \delta_i$ diverges, where each δ_i is the duration of the corresponding transition $(m_{i-1}, v_{i-1}, \vec{c}_{i-1}) \xrightarrow{\lambda_i} (m_i, v_i, \vec{c}_i)$. Let L be the set of divergent initialized trajectories of the labeled transition system S. The Hybrid MARTE statechart HMS is nonzeno if L is machine-closed for S.

Each trace $< \lambda_i, \vec{c}_i >_{i>=1}$ of the live transition system (S, L) is called a timed trace of HMS.

2.5 Composition

This section defines the timed semantics of the parallel composition $\text{HMS}_1 \parallel \text{HMS}_2$ of two Hybrid MARTE statecharts HMS_1 and HMS_2. The two Hybrid MARTE statecharts interact via joint events: if event a is both an event of HMS_1 and an event of HMS_2, then HMS_1 and HMS_2 must synchronize on a-transitions; if a is an event of HMS_1 but not an event of HMS_2, then

each a-transitions of HMS_1 synchronizes with a 0-duration time transition of HMS_2, and vice versa. For each real $\delta \geq 0$, an evolution of HMS_1 with duration δ must synchronize with an evolution of HMS_2 with the same duration.

Definition 2.16 (Product of transition systems)

A consistency check for two labeled transition systems S_1 and S_2 is an associative partial function $\otimes$ on pairs consisting of a transition from S_1 and a transition from S_2. The product $S1 \times S2$ with respect to the consistency check $\otimes$ is the labeled transition system with the state space $Q_1 \times Q_2$, the set $Q_1^0 \times Q_2^0$ of initial states, the label set $range(\otimes)$, and the following transition relations: for each label $a \in range(\otimes)$, define $((m_1, v_1, \vec{c}_1), (m_2, v_2, \vec{c}_2)) \xrightarrow{a} ((m_1, v_1, \vec{c}_1)'; (m_2, v_2, \vec{c}_2)')$ if there is a label $\lambda_1 \in A_1$ and a label $\lambda_2 \in A_2$ such that a is the (defined) result of applying $\otimes$ to the two transitions $(m_1, v_1, \vec{c}_1) \xrightarrow{\lambda_1} (m_1, v_1, \vec{c}_1)')$ and $(m_2, v_2, \vec{c}_2) \xrightarrow{\lambda_2} (m_2, v_2, \vec{c}_2)')$.

Definition 2.17 (Composition of Hybrid MARTE statecharts)

Consider two Hybrid MARTE statecharts HMS_1 and HMS_2. A transition $(m_1, v_1, \vec{c}_1) \xrightarrow{\lambda_1} (m_1, v_1, \vec{c}_1)')$ of S_1 and a transition $(m_2, v_2, \vec{c}_2) \xrightarrow{\lambda_2} (m_2, v_2, \vec{c}_2)')$ of S_2 are consistent if one of the following three conditions is true.

1. *$\lambda_1 = \lambda_2$. In this case, the consistency check $\otimes =$ applied to the transitions $(m_1, v_1, \vec{c}_1) \xrightarrow{\lambda_1} (m_1, v_1, \vec{c}_1)')$ and $(m_2, v_2, \vec{c}_2) \xrightarrow{\lambda_2} (m_2, v_2, \vec{c}_2)')$ yields λ_1.*
2. *$\lambda_1 \in \Sigma_1 \setminus \Sigma_2$ and $\lambda_2 = 0$. In this case, the consistency check $\otimes =$ yields λ_1.*
3. *$\lambda_1 = 0$ and $\lambda_2 \in \Sigma_2 \setminus \Sigma_1$. In this case, the consistency check $\otimes =$ yields λ_2.*

The timed transition system $S_{HMS_1 \| HMS_2}$ is defined to be the product $S_1 \times S_2$ with respect to the consistency check $\otimes =$.

3 Modeling ATP system with Hybrid MARTE Statecharts

This section briefly introduces an automatic train control system and the system requirements of the overspeed protection function. Then we develop a formal model of ATP system with Hybrid MARTE statecharts. At the end of this section, we also give a brief introduction to the simulation tool, namely HMSS, for Hybrid MARTE statecharts simulation. The simulation result of the example is showed in Fig.9.

Communication-Based Train Control System is the trend of rail train control system development in the future [21]. The core of CBTC system is Vehicle On-Board Controller (VOBC) Unit. It mainly accomplishes two parts of functional tasks: Automatic Train Protection (ATP) and Automatic Train Operation (ATO). Here we discuss the overspeed protection function in ATP.

ATP is the core subsystem of VOBC system. The train functions of acceleration, coasting, deceleration, stopping, and door opening are under the direct manual control of the Train Operator and are supervised by the ATP system. But its main responsibility, and our main concern, is to protect the system from overspeeding. Figure 3 shows the structure frame of VOBC system.

3.6 Requirements: Scenario of OverSpeed Protection

When a driver is driving the train under ATP mode, the ATP device helps the driver to protect the train from speeding. Here we have two actors: the driver and ATP device. Their interaction can be described as follows:

1. When the train finishes the self-detection, the ATP device is initialized.
2. According to the current mode selected by Mode Direction Handler and commands sent by the Train lines, ATO Processing Unit (PU) computes the current speed limit.
3. After a fixed interval (the period of sending current speed), Speed Sensor sends the current speed to PU.
4. PU computes the difference between the speed limit and the current speed and returns value dif.

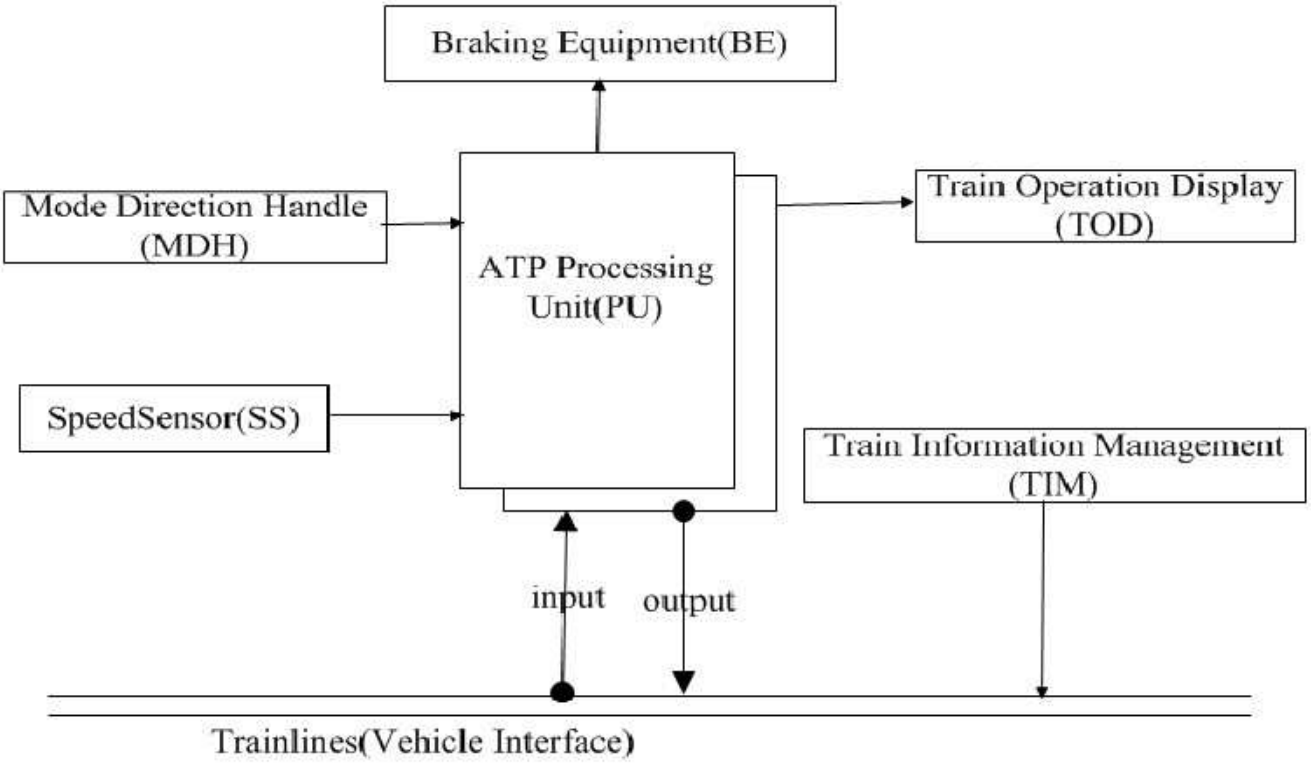

Fig. 3 Brief structure of VOBC system

5. If $0 <$ dif $<$ *criticalSpeedDif*, PU sends a warning message to the Train Operation Display (TOD) to inform the driver. After the warning, PU sends a normal brake message to the Braking Equipment (BE). Then the BE applies the normal brake pressure until $dif >$ *criticalSpeedDif*. All these operations should be done in $150\ ms$.
6. If dif ≤ 0, PU sends an emergency message directly to BE. Then BE engages into an emergency brake until velocity $v = 0$. These operations should be done within $100\ ms$.

3.7 Formal Modeling

Based on the syntax and semantics of Hybrid MARTE statecharts, we can model the behavior of the Overspeed Protection function of ATP system. Each component of VOBC system is a component of the system, but the most important components are Component PU and Component BE. We model the behavior of these two components using Hybrid MARTE statecharts in the following paragraphs.

Time model

As defined before, time is modeled by clocks in Hybrid MARTE statecharts. These clocks could have different time bases so that multiform time is available to be used in reactive synchronous language. In this example, we model two components: PU and BE. Figure 4 shows the time model of the ATP system with two *clockTypes*: ClockPUType and ClockBEType and their *UnitKinds*. Two instances of clocks are defined in the model (ClockPU and ClockBE) with different properties. We consider the two clocks inherit the default values for the resolution and offset from its type. The resolutions of the two clocks are

different, because when the breaking equipment is working, it needs higher precision than PU.

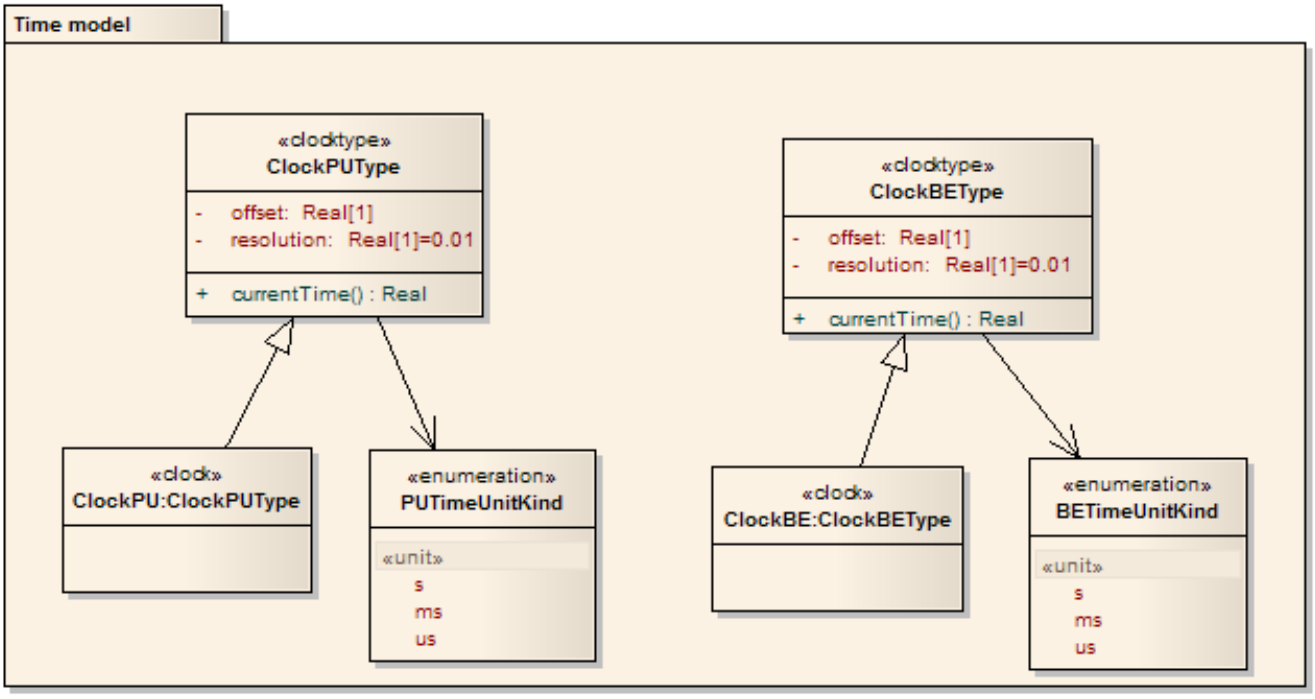

Fig. 4 Time Model of ATP system

System behavior

We use a Hybrid MARTE statechart to model the behavior of Component PU and Component BE. As discussed above, transitions in Hybrid MARTE statecharts can be divided into two types: either discrete (Edges) or continuous (Evolution). Continuous variables change when time passes.

Figure 5 describes the behavior of ATP Processing Unit. Note that the equations in the statechart capture the following equations:

$$\textit{TimeInvariants}(1) \triangleq$$
$$\textit{ClockPU}\sharp < 99.99 = \textit{ClockBE}\sharp < 99.999$$

$$\textit{TimeInvariants}(2) \triangleq$$
$$\textit{ClockPU}\sharp < 149.99 = \textit{ClockBE}\sharp < 149.999$$

$$*\textit{BrakingActivity} \triangleq \begin{cases} -fW = \frac{W}{g} \cdot \frac{dv}{dt} \\ v|_{t=0} = v_0 \end{cases}$$

Component PU is responsible for data computation and sending protection commands. The unique clock of PU is ClockPU. The transition system of PU begins in mode $idle$. Due to the space limitation, we only list the first three steps in the transition system:

- $\text{evt}(tr_1) \wedge \text{grd}(tr_1) = \epsilon \wedge \text{TRUE} = \epsilon$,
  $v, c \models \text{evt}(tr_1) \wedge \text{grd}(tr_1)$,
  $v, c, v, c' \models \text{act}(tr_1) = \emptyset$,
  so $tr_1 \triangleq (idle, v, c) \rightarrow (\textit{BrakingNotRequired}, v, c)$.

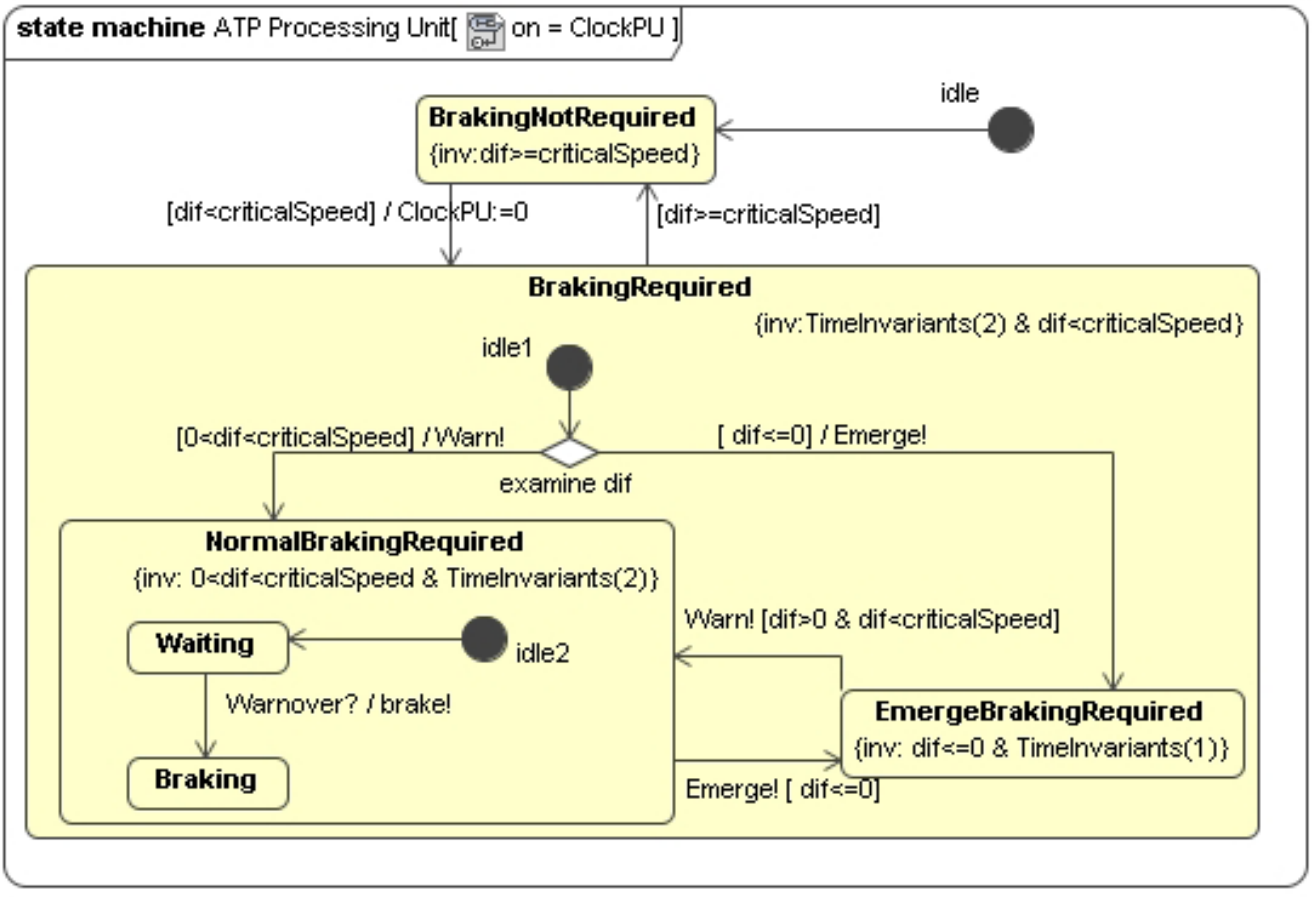

Fig. 5 Behavior of ATP Processing Unit

tr_1 is an edge and executes immediately after entering Component PU. Note that we require

$$v \models \text{inv}(\textit{BrakingNotRequired})$$

because $(\textit{BrakingNotRequired}, v, c)$ must be valid.

- $\text{evt}(tr_2) \wedge \text{grd}(tr_2) = \epsilon \wedge (\text{dif} < \textit{criticalSpeedDif})$,
  $\text{act}(tr_2) = \textit{ClockPU} := 0$,
  When $v, c \models \text{grd}(tr_2)$ and $(v, c, v, 0) \models \text{act}(tr_2)$,
  $tr_2 \triangleq (\textit{BrakingNotRequired}, v, c)$
  $\rightarrow (\textit{BrakingRequired}, v, 0)$,
  $\lambda(tr_2) = (\epsilon, \text{dif} < \textit{criticalSpeedDif}, \textit{ClockPU} := 0)$.
  When braking is required, no matter what kind of braking, the braking duration should be less than 150 ms. Hence in this mode
  $\text{inv}(\textit{BrakingRequired}) = (\textit{ClockPU}\sharp < 149.99)$
  $\wedge (\text{dif} < \textit{criticalSpeedDif})$
- $\text{evt}(tr_3) \wedge \text{grd}(tr_3) = \epsilon \wedge (0 < \text{dif} < \textit{criticalSpeedDif})$,

act(tr_3) = $Warn!$

When $v, c \models$ grd(tr_3) and $(v, c, v', c') \models$ act(tr_3),
$tr_3 \triangleq (idle_1, v, 0)$
$\rightarrow$ (*NormalBrakingRequired*, v, 0),
$\lambda(tr_3)$ = (ϵ, 0 < dif < *criticalSpeedDif*, $Warn!$).
When tr_3 is fired from $idle_1$ to submode
NormalBrakingRequired, PU sends a warning signal to TOD.

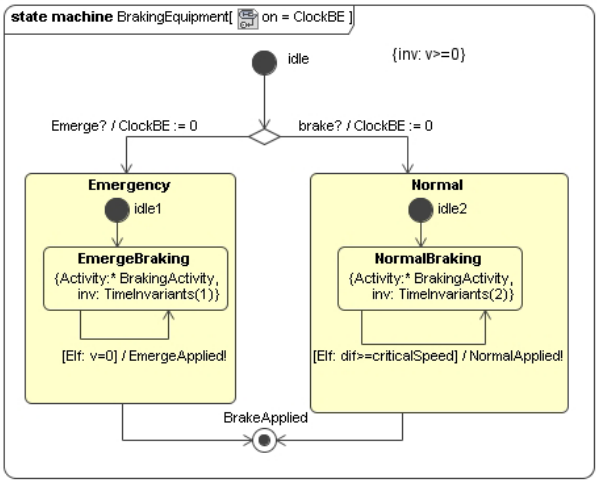

Fig. 6 Behavior of Component BrakingEquipment

Only when the velocity $v \geq 0$, the Braking Equipment state machine can be entered as Fig.6 shows. The transition system of BE is illustrated with several examples below:

- evt(tr_1) $\wedge$ grd(tr_1) = $Emerge?$ $\wedge$ TRUE,
  act(tr_1) = *ClockBE* := 0.

  When $v, c \models$ evt(tr_1) and $(v, c, v, 0) \models$ act(tr_1),
  $tr_1 \triangleq (idle, v, c) \rightarrow (Emergency, v, 0)$,

  $\lambda(tr_1) = (Emerge?,$ TRUE, *ClockBE* := 0).
  Edge tr_1 is triggered from *idle* to *Emergency*, and
  ClockBE is reset to 0.

- $tr_2 \triangleq (idle_1, v, 0) \rightarrow (EmergeBraking, v, 0)$
  Edge tr_2 is fired immediately after entering submode *Emergency*.

- inv(*EmergeBraking*) = (*ClockBE* < 100) $\wedge$ ($v \geqslant 0$)
  avt(*EmergeBraking*) = $\begin{cases} -fW = \frac{W}{g} \cdot \frac{\mathrm{d}v}{\mathrm{d}t} \\ v \mid_{t=0} = v_0 \end{cases}$.
  If f is a solution of equation avt(*EmergeBraking*), f satisties inv(*EmergeBraking*)$[f/v]$ = TRUE$\wedge$
  avt(*EmergeBraking*)$[f/v, \dot{f}/\dot{v}]$ = TRUE,
  then,
  $tr_3 \triangleq$ (*EmergeBraking*, v, 0)
  $\rightarrow$ (*EmergeBraking*, v', c')

tr_3 is an evolution. If emergency signal is received, *ClockBE* is reset and the emergency braking should be complete within 100ms. This is a continuous step of BE. In the submode *EmergeBraking*, the continuous variable v is changing when time passes, following the
avt(*EmergeBraking*). According to Newton second law of motion, fW is the external force to brake, $\frac{W}{g}$ is the weight of the train and $\frac{dv}{dt}$ is the acceleration of the train.

Time model is used to model the system time specification. Hybrid MARTE statecharts are used to model the behavior of components in the system. This case study shows how Hybrid MARTE statecharts can be applied to the modeling of real-time embedded system. The invariants and activities strengthen the description ability of statecharts. In this case the time-continuous variable velocity $v(t)$ is involved in the differential equation.

3.8 System Simulation

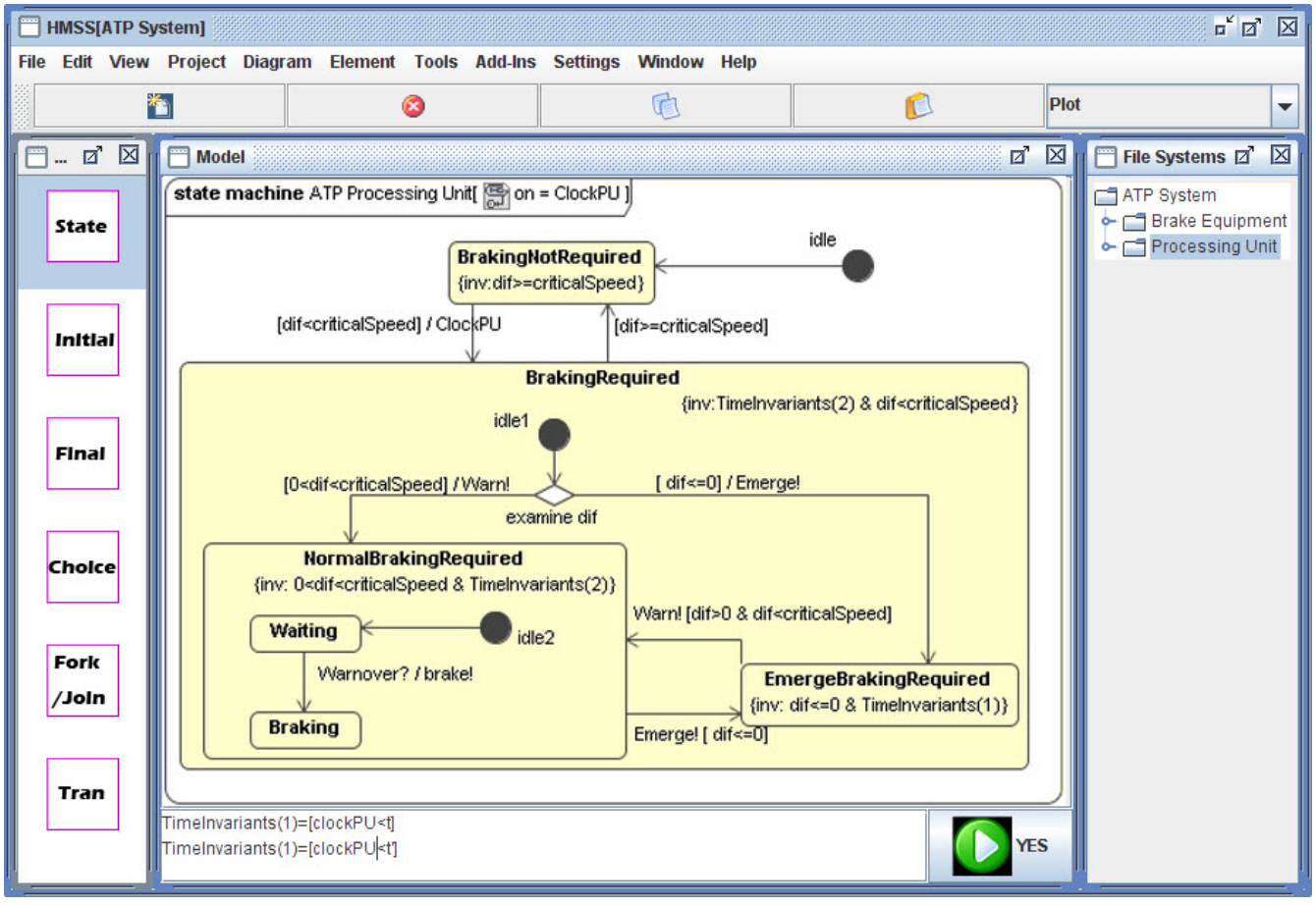

Fig. 7 Model the ATP System in HMSS: PU

Since we have developed a new modeling method for RTES, we also work on developing a new tool to support our model design and simulation. Hybrid MARTE statecharts are different from all other statecharts for its multiform time profile and its continuous activity characteristics. Thus, we have developed a tool, HMSS, for Hybrid MARTE statecharts simulation. Our tool is an extension of open source tool Ptolemy [22]. HMSS is an extensible modeling and simulation framework. It

offers modeling capabilities for Hybrid MARTE statecharts and it supports multiform time modeling. The tool will output the simulation result of the system, which is modeled in the graphic input area. Through the simulation result, we can see the variation process of the variables and can adjust the system to satisfy the requirements.

The ATP system is modeled in Fig.7 and Fig.8. We can draw the statecharts in the center of the tool, and below is the formula of the modes. After modeling the system, double click the initial mode to input the initial conditions. Then the tool will automatically output the simulation result (Fig.9).

Figure 9 shows the variations on the train velocity. When the train reaches the speed limit v, PU computes what kind of braking is needed and sends the corresponding signal to BE. BE applies the command accordingly. The acceleration of emergency brake is greater than the acceleration of normal brake. Acceleration and speed limit are adjusted until the system satisfies the TimeInvariants mentioned in Sec. 3.7.

4 Related Work

The UML profile for System Engineering (SysML) [23] is used at the system level. SysML extends UML in three main aspects: behavior, flow and structure allocations. Being used for creating object-oriented models of the systems, it includes software, people, material, and other physical resources. ModelicaML, the integration of UML/SysML and Modelica, was proposed by Fritzson and Pop [24]. ModelicaML is a UML profile that enables users to describe a Modelica simulation model graphically with UML/SysML information models. UML^{H} and is a special formation of UML for the modeling process of hybrid systems [25]. The motivation of UML^{H} is to give a graphical documentation of the object-oriented constructions of Modelica-models and to support the users to use the Modelica models [26].

Although SysML, ModelicaML and UML^{H}have improved many aspects compared to UML, they do not support strict time modeling and real-time resource management. However, UML profiles can be combined, and therefore Hybrid MARTE statecharts can be used as part of SysML models. Hybrid MARTE statecharts builds on MARTE time model, which is a specific time model for RTE systems. Moreover, Hybrid MARTE statecharts includes hybrid automata elements that can model continuous variables.

HybridUML [27, 28] is a profile extension of UML 2.0 for the specification of hybrid systems. This profile is based on the CHARON language of Alur et. al. [29, 30], so the profile inherits the formal CHARON semantics. A transformation is defined from HybridUML specifications to the programs written in a low-level language, which constructs the formal semantics of HybridUML.

Contrary those four profiles, Hybrid MARTE statecharts provides a different framework for time modeling that combines logical multiform time from MARTE and continuous time from hybrid automata. At the system level, components are often loosely coupled, without any strong synchronization mechanism. Assuming a common homogeneous time leads to an overspecification.

In [31], Frédéric Mallet and Robert de Simone extended UML time model in order to provide a generic timed interpretation. In [6, 8, 9, 32], they also expanded UML time description ability to multiform time, and proposed a new clock constraint language (CCSL) to specify the logical and chronometric time constraints. MARTE and SysML standardization task forces are closely related so that both profiles can be used jointly to benefit from their respective strengths.

Our work is based on the MARTE profile and hybrid automata. Contrary to the MARTE profile, Hybrid MARTE statecharts not only introduces a set of formal models able to model both logical and chronometric time, but also defines a formal method, using sampling functions, to model both continuous and discrete time. Hybrid MARTE statecharts enhances the modeling ability of MARTE. Based on MARTE time structure, we redefine the semantics of clock constraints in our Hybrid MARTE statecharts model, which is different from definitions in CCSL for the following aspects. Firstly, we introduce an operator $c\sharp$ to represent the current reading of a clock c by the first entry of the time instant sequence $\vec{c}$. Secondly, for continuous time modeling, a sampling function ρ (E.g. Taylor Series) is defined to discretize the continuous time to a series of observation points on demand. This unifies the continuous time and discrete time. Besides, Hybrid MARTE statecharts relies on Labeled Transition Systems to express the semantics. The wild acceptance of Labeled Transition Systems is likely to facilitate the

adoption of Hybrid MARTE statecharts.

5 Conclusions

Hybrid Real-Time and Embedded systems requires formalisms able to deal with both continuous and discrete time. Current researches on this issue are limited and require considerable manual effort. We have constructed an extension of UML state machines based on MARTE and hybrid automata, namely Hybrid Statecharts, for the hybrid Modeling and Analysis of Real-time and Embedded systems (Hybrid MARTE Statecharts). Dedicated to Real-Time and Embedded systems, Hybrid MARTE Statecharts offers a time structure to support multiform time modeling. The clock in the system has special time units in Hybrid MARTE Statecharts. Formal syntax and semantics of clock constraints (between different clocks) are given. A Hybrid MARTE statechart can specify the behavior of components, where the behavior is abstracted to the transitions between system states. States are classified into several modes satisfying specific constraints. The mode is one of the basic units of the statechart. The system execution is considered as the execution flow traveling among these modes. The formal model of mode behaviors is a labeled transition system with two kinds of transitions: *Edge* and *Evolution*. *Edge* specifies when the execution flow moves from one mode to another and *Evolution* represents transitions caused by the flow of time. The trace semantics and the composition of Hybrid MARTE statecharts are given. We have applied our model to the example from the Train Control System to display the ability to model the behavior of RTES. The simulation tool of Hybrid MARTE statecharts is also developed to show the evolution process of system. With the simulation results, designers can modify and improve their models.

Hybrid MARTE statecharts build a connection between MARTE and hybrid automata, which gives MARTE strong description ability. On one hand, users can use not only time-continuous variables but also discrete variables to model real time embedded systems at high level. On the other hand, we can do refinement step by step at low level modeling and implementation.

In the future we intend to work on the verification algorithm and tool support of Hybrid MARTE Statecharts. To facilitate modeling and analysis functions of Hybrid MARTE statecharts, we are currently working on the verification issues such as time invariant specification issues. We will also integrate our simulation tool with UML tools for the performance analysis.

6 *References

[1] OMG. *UML Superstructure, v2.2*. Object Management Group, February 2009. formal/2009-02-02.

[2] B. Selic. A systematic approach to domain-specific language design using UML. In *Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing*, pages 2–9. IEEE Computer Society, 2007.

[3] F. Thoen and F. Catthoor. *Modeling, verification, and exploration of task-level concurrency of real-time embedded systems*. Kluwer Academic Publishers, 2000.

[4] OMG. *UML Profile for MARTE, v1.0*. Object Management Group, November 2009. formal/2009-11-02.

[5] OMG. *UML Profile for Schedulability, Performance, and Time Specification, v1.1*. Object Management Group, January 2005. formal/05-01-02.

[6] Charles André, Frédéric Mallet, and Robert de Simone. Modeling time(s). In G. Engels, B. Opdyke, D. C. Schmidt, and F. Weil, editors, *Model Driven Engineering Languages and Systems*, volume 4735 of *Lecture Notes in Computer Science*, pages 559–573. Springer, 2007.

[7] Albert Benveniste, Paul Caspi, Stephen A. Edwards, Nicolas Halbwachs, Paul Le Guernic, and Robert de Simone. The synchronous languages 12 years later. *Proceedings of the IEEE*, 91(1):64–83, 2003.

[8] C. André, F. Mallet, and M.A. Peraldi-Frati. A multiform time approach to real-time system modeling; application to an automotive system. In *Industrial Embedded Systems, 2007. SIES'07. International Symposium on*, pages 234–241. IEEE.

[9] C. André. Syntax and semantics of the clock constraint specification language (CCSL). 2009.

[10] F. Mallet. CCSL: specifying clock constraints with UML/MARTE. *Innovations in Systems and Software Engineering*, 4(3):309–314, 2008.

[11] A. van der Schaft. An introduction to hybrid dynamical systems. 2000.

[12] A. Bemporad, A. Bicchi, and G.C. Buttazzo. *Hybrid Systems: Computation and Control*. Springer, 2007.

[13] A. Casagrande, C. Piazza, A. Policriti, and B. Mishra. Inclusion dynamics hybrid automata. *Information and Computation*, 206(12):1394–1424, 2008.

[14] H. Lu and S. Yu. Are statecharts finite automata? *Implementation and Application of Automata*, pages 258–261, 2009.

[15] D. Harel. Statecharts: A visual formalism for complex systems. *Science of computer programming*, 8(3):231–274, 1987.

[16] J.L. Sourrouille and G. Caplat. Constraint checking in UML modeling. In *Proc. of the 14th Int. Conf. on Software engineering and knowledge engineering*, pages 217–224. ACM, 2002.

[17] A. Edalat and D. Pattinson. Domain-theoretic solution of differential equations. 2006.

[18] R. Grosu and S.A. Smolka. Safety-liveness semantics for uml 2.0 sequence diagrams. In *Application of Concurrency to System Design, 2005. ACSD 2005. Fifth International Conference on*, pages 6–14. IEEE, 2005.

[19] T. Henzinger, P. Kopke, and H. Wong-Toi. The expressive power of clocks. *Automata, Languages and Programming*, pages 417–428, 1995.

[20] T.A. Henzinger. Sooner is safer than later. *Information Processing Letters*, 43(3):135–141, 1992.

[21] IEEE. *IEEE Recommended Practice for Communications-Based Train Control (CBTC) System Design and Functional Allocations*. IEEE Std 1474.3-2008, 2008.

[22] Edward A. Lee and Stavros Tripakis. Modal models in ptolemy. In Peter Fritzson, Edward A. Lee, François E. Cellier, and David Broman, editors, *EOOLT*, volume 47 of *Linköping Electronic Conference Proceedings*, pages 11–21. Linköping University Electronic Press, 2010.

[23] The Object Management Group. Systems modeling language (sysml) specification, ptc/08-05-07, 2008.

[24] A. Pop, D. Akhlevidiani, and P. Fritzson. Towards unified system modeling with the modelicaml uml profile. *EOOLT*, page 13, 2007.

[25] C. Nytsch-Geusen. The use of the uml within the modelling process of modelica-models. *EOOLT*, page 1, 2007.

[26] T. Johnson, C.J.J. Paredis, and R. Burkhart. Integrating models and simulations of continuous dynamics into sysml. In *Modelica Conference 2008*, 2008.

[27] K. Berkenkotter, S. Bisanz, U. Hannemann, and J. Peleska. Executable HybridUML and its application to train control systems. *Integration of Software Specification Techniques forApplications in Engineering*, pages 145–173, 2004.

[28] K. Berkenkotter, S. Bisanz, U. Hannemann, and J. Peleska. The HybridUML profile for UML 2.0. *International Journal on Software Tools for Technology Transfer (STTT)*, 8(2):167–176, 2006.

[29] R. Alur, R. Grosu, I. Lee, and O. Sokolsky. Compositional refinement for hierarchical hybrid systems. *Hybrid Systems: Computation and Control*, pages 33–48, 2001.

[30] R. Alur, T. Dang, J.M. Esposito, R.B. Fierro, Y. Hur, F. Ivancic, V. Kumar, I. Lee, P. Mishra, G.J. Pappas, et al. Hierarchical hybrid modeling of embedded systems. In *Proceedings of the First International Workshop on Embedded Software*, pages 14–31. Springer-Verlag, 2001.

[31] F. Mallet and R. De Simone. MARTE: a profile for RT/E systems modeling, analysis and simulation? In *Proc. of the 1st Int. Conf. on Simulation tools and techniques for communications, networks and systems*, pages 1–8. ICST, 2008.

[32] F. Mallet and C. André. On the semantics of UML/MARTE clock constraints. In *Object/Component/Service-Oriented Real-Time Distributed Computing, 2009. ISORC'09. IEEE International Symposium on*, pages 305–312. IEEE.

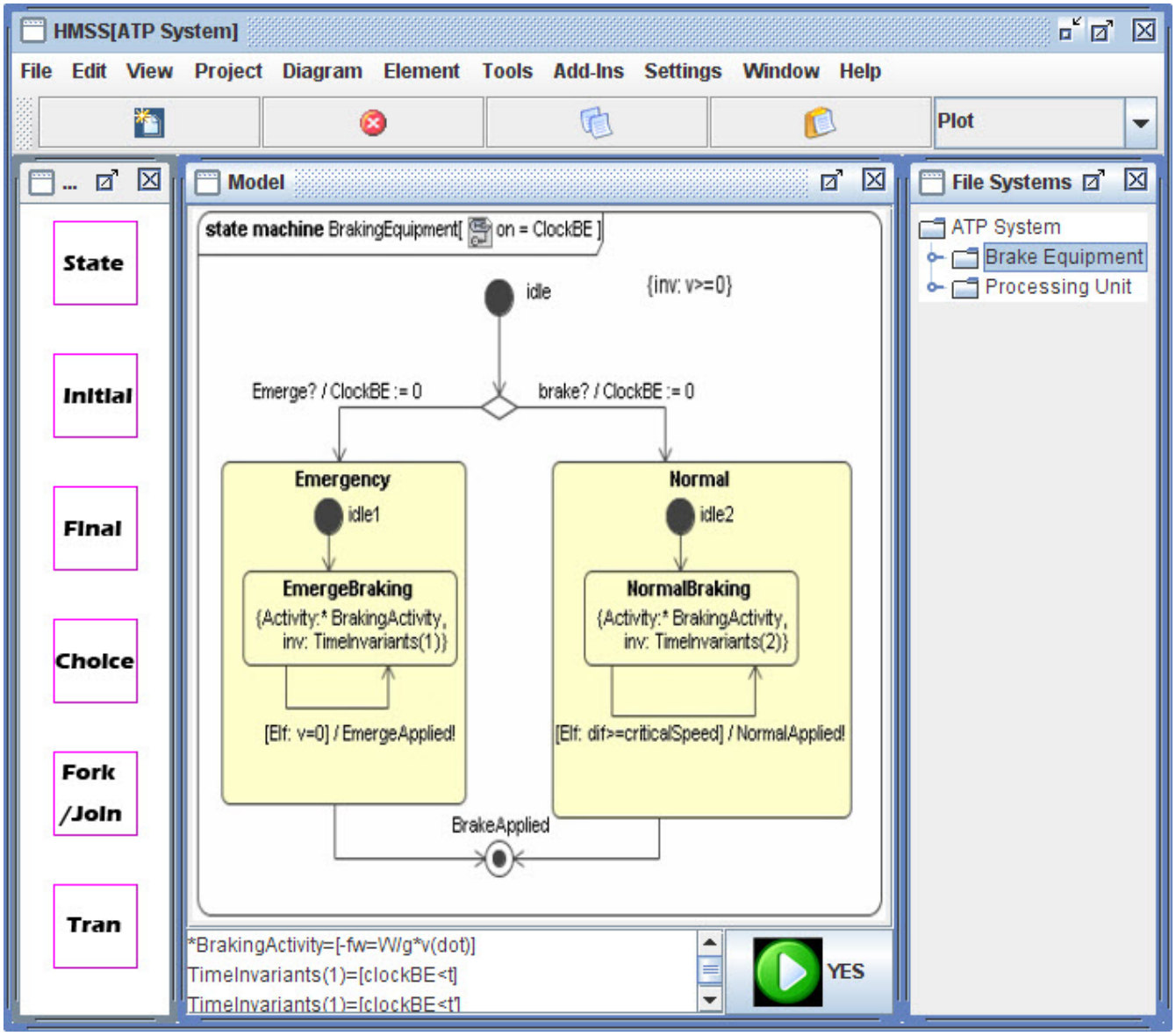

Fig. 8 Model the ATP System in HMSS: BE

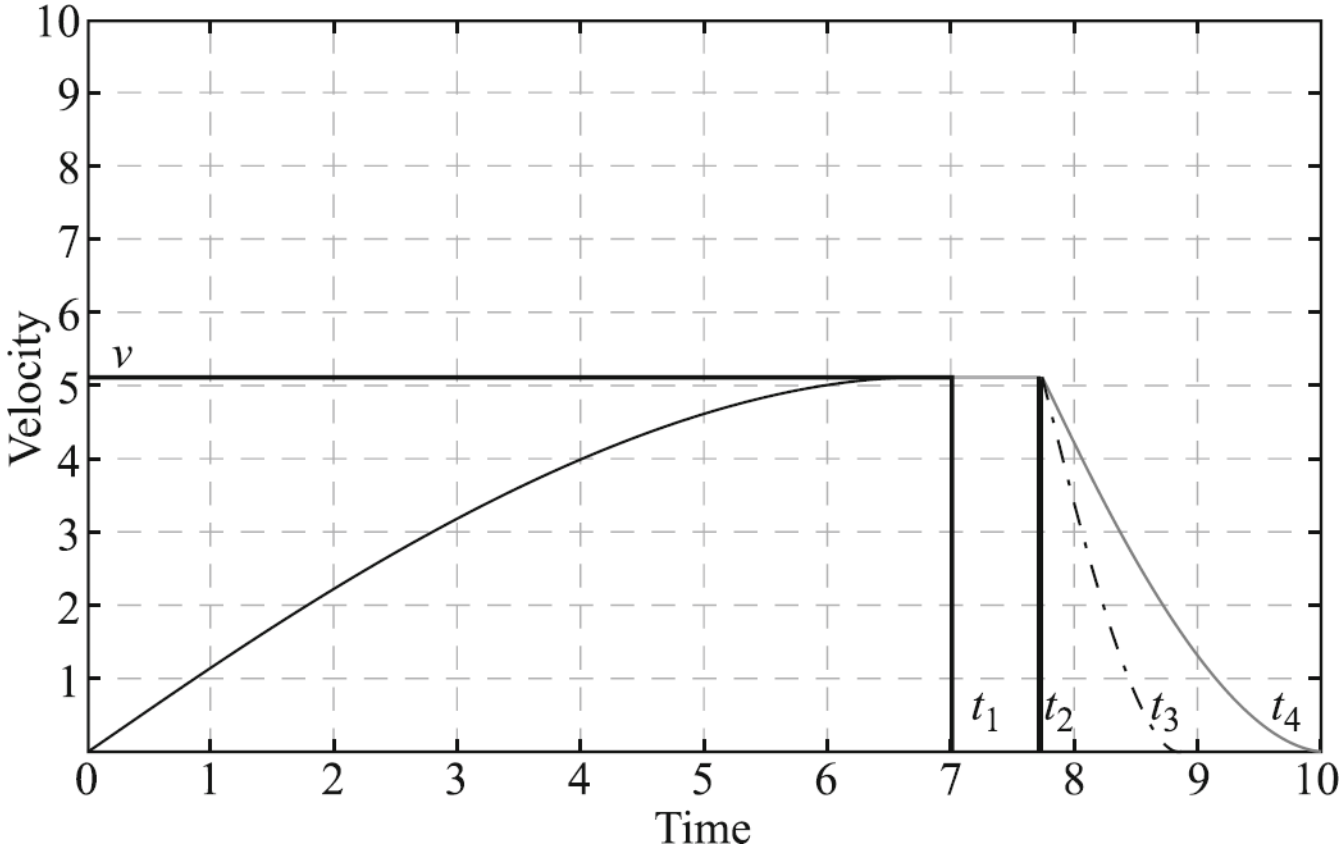

Fig. 9 Result of simulation