Timed Automata Semantics of Spatio-Temporal Consistency Language STeC

Yuanrui Zhang, Frédéric Mallet, Yixiang Chen

Timed Automata Semantics of Spatio-Temporal Consistency Language STeC

Yuanrui Zhang
Institute of Software Engineering,
East China Normal University,
Shanghai, China
Polytech Nice Sophia,
University Nice Sophia Antipolis,
Nice, France
Email: yuanrui1990.zhang@gmail.com

Frederic Mallet
Joint team AOSTE (INRIA/I3S)
(the Anlysis and Optimization of
Systems with real-Time and
Embedding constraints),
University Nice Sophia Antipolis,
Nice, France
Email: Frederic.Mallet@unice.fr

Yixiang Chen
MoE Engineering Research Center for
Software/Hardware Co-design
Technology and Application,
East China Normal University,
Shanghai, China
Email: yxchen@sei.ecnu.edu.cn

***Abstract*—Intelligent Transportation Systems (ITS) are a class of quickly evolving modern safety-critical embedded systems. Dealing with their growing complexity demands a high-level formal modeling language along with adequate verification techniques. STeC has recently been introduced as a process algebra that deals natively with both spatial and temporal properties. Even though STeC has the right expressive power, it does not provide a direct tooled support for verification. We propose to encode STeC specifications as Timed Automata to provide such a support and we illustrate our transformation strategy on a simple example.**

I. Introduction

A great variety of languages have been developed over the last two decades to describe timed or untimed models (like CSP[1], π-calculus[2], Timed-CSP[3], Timed Automata[4]) and timed or untimed properties (LTL, CTL, TCTL or PSL[5]).

Timed Automata, introduced by Alur and Dil, have inspired a lot of works as they introduced real-valued clocks thus making 'time' a first-class citizen of the specification languages. Sound and mature verification tools have also been developed around Timed Automata, which makes them a good 'assembly language' for other high-level modeling languages in search for a verification support.

In embedded systems, time is acknowledged as being of central importance. In the domain of Intelligent Transportation Systems (ITS), which is developing rapidly and is gaining a growing attention, we believe that both models and properties need to consider not only time but also spatial aspects. That is why we consider STeC (Spatio-Temporal Consistence Language) as a good candidate process algebra to build spatio-temporal models([6], [7]). Indeed, STeC makes (physical) location a first-class citizen of the specification. In this paper, we propose to build a high-level verification framework for spatio-temporal specifications. We use STeC to capture the models and we rely on Timed Automata to provide the foundational verification support. After discussing a global high-level modeling framework to capture both models and properties, we focus the discussion on a transformation from STeC to Timed Automata. Expressing high-level properties is not part of the discussion here and we rely on standard temporal logics properties to illustrate our approach on a simple train control system.

In order to provide a direct verification support for the spatio-temporal models built by STeC. We propose a global verification framework to illustrate our goal and main works. In the framework we choose a specification language(called CCSL [10]) for STeC-models and show globally how to provide such a support by transforming them into Timed Automata on which verification techniques can be applied. Then we focus on the details of transformation from STeC to Timed Automata—the main contribution of this paper.

In Section 2, some background knowledge and definitions are given, which are essential for introducing the contribution of this paper. In Section 3 we introduce a possible verification framework for verifying STeC-models. In Section 4, we give our main contribution of this paper. In Section 5, a summary of this paper is given.

II. Background and Definitions

In this section, some basic definitions about STeC and TA are introduced. These knowledge are prerequisite for the contribution part. For the tranditional definitions and more details, see [4], [6], [8] and [9].

A. An Introduction of STeC

Yixiang Chen introduced the Spatio-Temporal Consistence Language in 2010[6] and set up its formal semantics with his colleagues in [7]. Unlike the traditional formal modeling languages, it stresses the location and the time of an event of real-time systems, especially ITS. An event(or action) is executed at a location and time. For example, in a Train Scheduling System(see [6], [7]), trains travel from one platform to another with very strict time restriction. If a train do not arrive at a station at 'correct' time, the collisions might happen. This characteristic of STeC makes it a powerful and an easy-understand formal modeling language for ITS(for more examples modeled by STeC, see [6], [7]). However, despite of its easy-expressing feature, STeC does not support for verification considering there is no specification language used

to describe the spatio-temporal properties and we can not apply traditional verification techniques which is based on automata directly to the STeC-model.

The syntax and operational semantics of STeC are given as below.

1) The Syntax of STeC: The syntax of STeC is as follows, in Backus-Naur Form:

$$A \ ::= \ \mathbf{Send}^{G\rightarrow G'}_{(l,t)}(m) \mid \mathbf{Get}^{G\leftarrow G'}_{(l,t)}(m)$$

$$B \ ::= \ \alpha_{(l,t)}(l',\delta) \mid \beta_{(l,t)}(\delta) \mid B \mid B \wedge B$$

$$P \ ::= \ \mathbf{Stop}^{G}_{(l,t)} \mid \mathbf{Skip}^{G}_{(l,t)} \mid \mathbf{Send}^{G\rightarrow G'}_{(l,t)}(m) \mid \mathbf{Get}^{G\leftarrow G'}_{(l,t)}(m) \mid$$

$$\alpha^{G}_{(l,t)}(l',\delta) \mid \beta^{G}_{(l,t)}(\delta) \mid P;P \mid P \ [] \ P \mid P \parallel P \mid B \rightarrow P \mid$$

$$[]_{i\in I}B_i \rightarrow P_i \mid P \trianglerighteq_{\delta} P \mid P \trianglerighteq ([]_{i\in I}A_i \rightarrow P_i)$$

$\mathbf{Stop}^{G}_{(l,t)}$, $\mathbf{Skip}^{G}_{(l,t)}$, $\mathbf{Send}^{G\rightarrow G'}_{(l,t)}(m)$, $\mathbf{Get}^{G\leftarrow G'}_{(l,t)}(m)$, $\alpha^{G}_{(l,t)}(l',\delta)$ and $\beta^{G}_{(l,t)}(\delta)$ are atomic processes, $P;P$ is the sequence operator and $P \parallel P$ is the parallel operator. $P[]P$ is the choice operator. $[]_{i\in I}B_i \rightarrow P_i$ is a guard choice. $P \trianglerighteq_{\delta} Q$ behaves as P for up to δ time units and then behaves as Q. $P \trianglerighteq ([]_{i\in I}A_i \rightarrow P_i)$ initially proceeds like P and is interrupted on occurrence of the atomic command A_i and then process like P_i. Here A_i is an interrupt command.

For more details about the syntax of STeC, refer to [6] and [7].

2) The Operational Semantics of STeC: A storage σ is introduced in STeC as the storage of messages between agents. Let $\sigma \subseteq 2^{\mathbb{M}}$, $\mathbb{M}$ is the set of messages of the form $m_{G\rightarrow G'}$. Here we use $\mathbf{Sto}$ to represent the set of storages and it is easy to see that: $\mathbf{Sto} \ \subseteq \ 2^{2^{\mathbb{M}}}$.

Let $\mathbf{Loc}$ be the set of locations. $\mathbb{T} \subseteq \mathbb{R}$ is the set of time space. An environment is a triple (a,u,σ) where σ is a storage, a is a location and u is time. We use the notation $\mathcal{E}$ to represent the set of all the environments: $\mathcal{E} \ = \ \mathbf{Loc} \times \mathbb{T} \times \mathbf{Sto}$.

Let $\mathbb{B} = \{B | B \ is \ a \ 'B' \ type \ formula \ in \ STeC\}$(see the syntax in STeC). A function $\mathcal{T} : \mathbb{B} \rightarrow \{0,1\}$ is defined as the truth valuation function. We have $\mathcal{T}(B) = 1$(resp. $\mathcal{T}(B) = 0$) if B is true (resp. false).

A configuration of process P is defined as a tuple $\langle P,(a,u,\sigma)\rangle$. Define $\mathbf{Stconf} = \{\langle P,(a,u,\sigma)\rangle \mid P \ is \ a \ process \ and \ for \ some \ a, \ u, \ \sigma\}$ as the set of all configurations in STeC.

The transition relation of STeC $\hookrightarrow_{STeC}$ is a function:

$$\hookrightarrow_{STeC}: \mathbf{Stconf} \rightarrow 2^{\mathbf{Stconf}}$$

The transition relation can be denoted as $\langle P,(a,u,\sigma)\rangle \rightarrow \langle P',(a',u',\sigma')\rangle$ iff $\hookrightarrow_{STeC} \ (\langle P,(a,u,\sigma)\rangle) = \langle P',(a',u',\sigma')\rangle$ holds.

The operational semantics of STeC is as follows. For more detail, refer to [6].

- $\frac{a=l,u=t}{\langle \mathbf{Send}^{G\rightarrow G'}_{(l,t)}(m),(a,u,\sigma)\rangle \rightarrow \langle E,(l,t,\sigma\cup\{m_{G\rightarrow G'}\})\rangle}$
- $\frac{a=l,u=t}{\langle \mathbf{Get}^{G\leftarrow G'}_{(l,t)}(m),(a,u,\sigma)\rangle \rightarrow \langle E,(l,t,\sigma\backslash\{m_{G\rightarrow G'}\})\rangle}$
- $\frac{a=l,u=t}{\langle \alpha_{(l,t)}(l',\delta),(a,u,\sigma)\rangle \rightarrow \langle E,(l',t+\delta,\sigma)\rangle}$
- $\frac{a=l,u=t}{\langle \beta_{(l,t)}(\delta),(a,u,\sigma)\rangle \rightarrow \langle E,(l,t+\delta,\sigma)\rangle}$
- $\frac{a=l,u=t}{\langle \mathbf{Stop}^{G}_{(l,t)},(a,u,\sigma)\rangle \rightarrow \langle \mathbf{Stop}^{G}_{(l,t+1)},(l,t+1,\sigma)\rangle}$
- $\frac{a=l,u=t}{\langle \mathbf{Skip}^{G}_{(l,t)},(a,u,\sigma)\rangle \rightarrow \langle E,(l,t,\sigma)\rangle}$
- $\frac{\langle P,(a,u,\sigma)\rangle \rightarrow \langle P',(a',u',\sigma')\rangle}{\langle P;Q,(a,u,\sigma)\rangle \rightarrow \langle P';Q,(a',u',\sigma')\rangle}$
- $\frac{\langle P,(a,u,\sigma)\rangle \rightarrow \langle P',(a',u',\sigma')\rangle}{\langle P;Q,(a,u,\sigma)\rangle \rightarrow \langle P';Q,(a',u',\sigma')\rangle}$
- $\frac{\langle P,(a,u,\sigma)\rangle \rightarrow \langle P',(a_1,u_1,\sigma_1)\rangle, \ \langle Q,(a,u,\sigma)\rangle \rightarrow \langle Q',(a_2,u_2,\sigma_2)\rangle}{\langle P[]Q,(a,u,\sigma)\rangle \rightarrow \{\langle P',(a_1,u_1,\sigma_1)\rangle,\langle Q',(a_2,u_2,\sigma_2)\rangle\}}$
- $\frac{\langle P,(a,u,\sigma)\rangle \rightarrow \langle P',(a',u',\sigma_1)\rangle, \ \langle Q,(a,u,\sigma)\rangle \rightarrow \langle Q',(a',u',\sigma_2)\rangle}{\langle P\|Q,(a,u,\sigma)\rangle \rightarrow \langle P'\|Q',(a',u',\sigma_1 \uplus \sigma_2)\rangle}$
- $\frac{\mathcal{T}(B)=1}{\langle B\rightarrow P,(a,u,\sigma)\rangle \rightarrow \langle P,(a,u,\sigma)\rangle}$ $\frac{\mathcal{T}(B)=0}{\langle B\rightarrow P,(a,u,\sigma)\rangle \rightarrow \langle \mathbf{Stop}^{G}_{(l,t)},(a,u,\sigma)\rangle}$
- $\frac{\mathcal{T}(B_i)=1}{\langle []_{i\in I}B_i\rightarrow P_i,(a,u,\sigma)\rangle \rightarrow \langle P_i,(a,u,\sigma)\rangle}$
- $\frac{\langle P,(a,u,\sigma)\rangle \rightarrow \langle P',(a',u',\sigma')\rangle \ \wedge \ u'\leq\delta}{\langle P\trianglerighteq_{\delta}Q,(a,u,\sigma)\rangle \rightarrow \langle Q,(a',u',\sigma')\rangle}$ $\frac{\langle P,(a,u,\sigma)\rangle \rightarrow \langle P',(a',u',\sigma')\rangle \ \wedge \ u'>\delta}{\langle P\trianglerighteq_{\delta}Q,(a,u,\sigma)\rangle \rightarrow \langle Q,(a_1,\delta,\sigma_1)\rangle}$

B. An Introduction of Timed Automata[4]

1) The Syntax of TA: Let $\mathcal{C}$ be a finite set of non-negative real-valued variables called clocks. The set of guards $G(\mathcal{C})$ is defined by the grammar $g := x \bowtie c \mid g \wedge g$ where $g \in \mathcal{C}$, $c \in \mathbb{N}$ and $\bowtie \in \{<,\leq,>,\geq\}$. A Timed Automata is a tuple $A = (Q,\Sigma,\mathcal{C},q_0,E,I,AP,L,F)$, where:

- Q is a finite set of states,
- Σ is a finite alphabet,
- $\mathcal{C}$ is a finite set of clocks,
- $q_0 \in Q$ is an initial state,
- $E \subseteq Q \times \Sigma \times G(\mathcal{C}) \times 2^{\mathcal{C}} \times 2^{Q}$ is a finite transition relation,
- $I : Q \rightarrow G(\mathcal{C})$ is an invariant-assignment function,
- AP is a finite set of atomic propositions,
- $L : Q \rightarrow 2^{AP}$ is a labeling function for the states,
- $F \subseteq Q$ is a set of accepting states.

2) The Operational Semantics of TA: A clock valuation is a function $\nu : \mathcal{C} \rightarrow \mathbb{T}$. We define $\nu + r$ as: for each clock $x \in \mathcal{C}$, $(\nu + r)(x) = \nu(x) + r$, where $r \in \mathbb{T}$. If $Y \subseteq \mathcal{C}$ then a valuation $\nu[Y := 0]$ is such that for each clock $x \in \mathcal{C}\backslash Y$, $\nu[Y := 0](x) = \nu(x)$ and for each clock $x \in Y$, $\nu[Y := 0](x) = 0$. The satisfaction relation $\nu \models g$ for $g \in G(\mathcal{C})$ is defined in the natural way.

The operational semantics of a TA $A = (Q,\Sigma,\mathcal{C},q_0,E,I,AP,L,F)$ is a labeled transition system (LTS):

$$\hookrightarrow_A (q,\nu) : \ (Q \times \mathbb{T}^{\mathcal{C}})_{\perp} \ \rightarrow \ 2^{(Q\times\mathbb{T}^{\mathcal{C}})_{\perp}}$$

where each mapping defines a transition relation $\langle q,\nu\rangle \rightarrow \langle q',\nu'\rangle$, which is defined as follows:

$$\hookrightarrow_A (q,\nu) = \begin{cases} \{\langle q, \nu+\alpha\rangle\} & \text{if } \exists\alpha \in \mathbb{T} \wedge (\nu+\alpha \models I) \\ \{\langle q', \nu'\rangle \mid q' \in Q'\} & \text{if } \exists\langle q,a,g,Y,Q'\rangle \in E \\ & \text{where } \nu \models g, \nu' = \nu[Y:=0] \\ \{\langle q', \nu'\rangle \mid q' \in Q'\} \cup \{\langle q, \nu+\alpha\rangle\} & \text{if both} \\ \bot & \text{otherwise} \end{cases}$$

3) Transition Relation Function: The transition relation function $\psi : Q \to 2^Q$ is a function over the states of TA. We declare that for any $q \in Q$, there exists a value of $\psi(q)$ if and only if there exists a tuple $\langle q, a, g, Y, Q'\rangle \in E$, denoted as $\psi(q) = Q'$. So we can say for each TA there is a conrrespondent transition relation function.

III. The Verification Framework of Spatio-Temporal Models

One possible way to enhance STeC-model to be directly verifiable is to transform it into Timed Automata. We introduce a standard temporal logic specification language, called CCSL and use it to specify properties of STeC-models. CCSL—The Clock Constraint Specification Language, was firstly proposed by Frederic Mallet in 2008[10]. It is aimed at describing the temporal order of events for real-time systems. In CCSL the concept of 'clock' of each event was introduced and the complexity logical relations between events are expressed by the composition of such 'clock'. Because of this, some properties are stated in a simpler way using CCSL than using other temporal logic specification languages(such as LTL). CCSL is well-defined in its syntax and semantics(see [10], [11]), and can be encoded into Timed Automata in order to support verification(for more information about CCSL, refer to [5], [11], [10], [12], [13], [14] and [15]).

As shown in Fig. 1 below, to verify spatio-temporal specifications of STeC models, we transform STeC and CCSL into Timed Automata(TA). $\mathcal{M}$ is the transformed transition model from STeC, and $\mathcal{A}_\varphi$ is the transformed TA specifying properties. We can verify properties by combining the two TAs—$\mathcal{M} \times \mathcal{A}_\varphi$. ① is the main contribution of this paper, which is discussed in the next Section. For more details about ②, see [12].

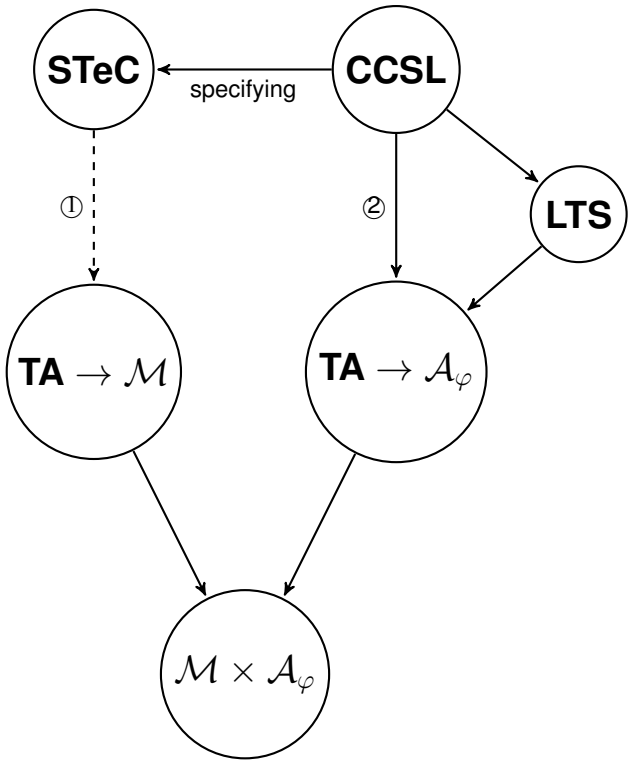

Fig. 1. The Verification Framework of STeC-models

IV. The Transformation From STeC to TA

In this section, we mainly focus the transformation from STeC to TA. We want to build a deductive mapping on the structure of the syntax of STeC. But firstly, we have to introduce three binary operations on TAs in order to form a new TA from the old ones. Later in the Definition IV.6, we see how to apply these three operations on the transformation.

A. The Definition of New Binary Operators

Firstly, we introduce some binary operations in TAs, which are essential for building the transformation from STeC to TA.

1) STeC-Concatenation: We introduce an binary operator in TA, denoted as $\diamond_{STeC}$. Later in Definition IV.6, we can see that for every $R = P; Q$, we have $A_R = A_P \diamond_{STeC} A_Q$, where A_* are the corresponding Timed Automatas of process $*$.

Definition IV.1. *Set $A = (Q, \Sigma, \mathcal{C}, q_0, E, I, AP, L, F)$, $A' = (Q', \Sigma', \mathcal{C}', q_0', E', I', AP', L', F')$ are two TAs. And Set ψ and ψ' are their transition relation functions correspondingly. We define an operation called "STeC-Concatenation" of two TAs as a special case of the operator "Concatenation" of TA. It is defined as*

$$\diamond_{STeC} : \mathbf{TA} \times \mathbf{TA} \to \mathbf{TA}$$

where **TA** *represents the set of all TAs. Let $A'' = (Q'', \Sigma'', \mathcal{C}'', q_0'', E'', I'', AP'', L'', F'')$ be a TA which satisfies $A'' = A \diamond_{STeC} A'$. We define it as follows:*

- $A'' = (Q \cup Q', \Sigma \cup \Sigma', \{x'', y''\} \cup \mathcal{C}\backslash\{x_A, y_A\} \cup \mathcal{C}'\backslash\{x'_A, y'_A\}, q_0, E'', I \cup I', AP \cup AP', L'', F')$. *$\{x_A, y_A\}$ and $\{x'_A, y'_A\}$ are the clock x and the clock y for A and A' respectively, which has special meanings in the transition of STeC to TA (will be introduced later). In A'', all constraints concerning about clock x and clock y in A or A' will be replaced with new clock x'' and clock y''.*
- *E'' is defined by replacing clock $\{x, y\}$ with the new clock $\{x'', y''\}$ and defining the transition relation function ψ''. $\psi''(q)$ is defined as:*

$$\psi''(q) = \begin{cases} \psi(q) & \text{if } q \in Q \wedge \{q\} \cap F = \emptyset \\ \psi(q) \cup \psi'(q) & \text{if } q \in Q \wedge \{q\} \cap F \neq \emptyset \\ \psi'(q) & \text{if } q \in Q' \end{cases}$$

- *The labelling function L'' is defined as:*

$$L''(q) = \begin{cases} L(q) & \text{if } q \in Q \wedge q \notin F \\ L(q) \cup L'(q) & \text{if } q \in F \\ L'(q) & \text{if } q \in Q' \end{cases}$$

2) STeC-Union: Denoted as $\oplus_{STeC}$, like $\diamond_{STeC}$, it corresponds to the "choice" operator in STeC, for example, $P[]Q$.

Definition IV.2. *Let $A = (Q, \Sigma, \mathcal{C}, q_0, E, I, AP, L, F)$, $A' = (Q', \Sigma', \mathcal{C}', q_0', E', I', AP', L', F')$ be two TAs. Define*

$$\oplus_{STeC} : \ \mathbf{TA} \times \mathbf{TA} \to \mathbf{TA}$$

as the "STeC-union" as a special case of the "union" operator of TA, which satisfies:

- *Let $A'' = A \oplus_{STeC} A'$, then $A'' = (Q \cup Q' \cup \{q_0''\}, \Sigma \cup \Sigma', \{x'', y''\} \cup \mathcal{C}\backslash\{x_A, y_A\} \cup \mathcal{C}'\backslash\{x_A', y_A'\}, q_0'', E'', I'', AP \cup AP', L'', F \cup F')$*
- *We define E'' by replacing clock $\{x, y\}$ with the new clock $\{x'', y''\}$ and defining the transition relation function ψ''. $\psi''(q)$ is defined as:*

$$\psi''(q) = \begin{cases} \psi(q) & \text{if } q \in Q \\ \psi'(q) & \text{if } q \in Q' \\ \{q_0, q_0'\} & \text{if } q = q_0'' \end{cases}$$

- *The labelling function L'' is defined as:*

$$L''(q) = \begin{cases} L(q) & \text{if } q \in Q \\ L'(q) & \text{if } q \in Q' \\ \emptyset & \text{if } q = q_0'' \end{cases}$$

- *The invariant-assignment function I'' is defined as:*

$$I''(q) = \begin{cases} I(q) & \text{if } q \in Q \\ I'(q) & \text{if } q \in Q' \\ \{y \leq 0\} & \text{if } q = q_0'' \end{cases}$$

3) STeC-HandShaking: Denoted as $\otimes_{STeC}$, it corresponds to the "parallel" operator in STeC, such as $P||Q$.

Definition IV.3. *Knowing $A = (Q, \Sigma, \mathcal{C}, q_0, E, I, AP, L, F)$, $A' = (Q', \Sigma', \mathcal{C}', q_0', E', I', AP', L', F')$ are two TAs. The definition of "STeC-Union" is given as follows:*

$$\otimes_{STeC} : \ \mathbf{TA} \times \mathbf{TA} \to \mathbf{TA}$$

set $A'' = A \otimes_{STeC} A'$, it satisfies the following properties:

- *$A'' = (Q \times Q', \Sigma \cup \Sigma', \{x'', y''\} \cup \mathcal{C}\backslash\{x_A, y_A\} \cup \mathcal{C}'\backslash\{x_A', y_A'\}, \langle q_0, q_0'\rangle, E'', I'', AP \cup AP', L'', F \times F')$*
- *The definition of E'' can be deduced by replacing clock $\{x, y\}$ with the new clock $\{x'', y''\}$ and defining the transition relation function ψ''. $\psi''(\langle q_1, q_2\rangle)$ is defined as:*

$$\psi''(\langle q_1, q_2\rangle) =$$

$$\begin{cases} \langle \psi(q_1), \psi'(q_2)\rangle & \text{if } \langle q_1, q_2\rangle \in Q \times Q' \wedge \\ & \exists \langle q_1, m!, g, Y, Q\rangle \in E \text{ and} \\ & \exists \langle q_2, m?, g', Y', Q'\rangle \in E' \\ & \textit{OR} \\ & \exists \langle q_1, m?, g, Y, Q\rangle \in E \text{ and} \\ & \exists \langle q_2, m!, g', Y', Q'\rangle \in E' \\ \{\ \langle \psi(q_1), \psi'(q_2)\rangle, \langle \psi(q_1), q_2\rangle, \langle q_1, \psi(q_2)\rangle\ \} & \\ & \text{if } \langle q_1, q_2\rangle \in Q \times Q' \wedge \\ & \exists \langle q_1, a, g, Y, Q\rangle \in E \text{ and} \\ & \exists \langle q_2, a', g', Y', Q'\rangle \in E' \\ \langle \psi(q_1), q_2\rangle & \text{if } \langle q_1, q_2\rangle \in Q \times Q' \wedge \\ & \exists \langle q_1, a, g, Y, Q\rangle \in E \\ \langle q_1, \psi(q_2)\rangle & \text{if } \langle q_1, q_2\rangle \in Q \times Q' \wedge \\ & \exists \langle q_2, a', g', Y', Q'\rangle \in E' \end{cases}$$

where m is any message in STeC.

- *The labelling function L'' is defined as:*

$$L''(\langle q_1, q_2\rangle) = L(q_1) \cup L'(q_2) \text{ for all } q_1 \in Q \text{ and } q_2 \in Q'$$

- *The invariant-assignment function I'' is defined as:*

$$I''(\langle q_1, q_2\rangle) = I(q_1) \cup I'(q_2) \text{ for all } \langle q_1, q_2\rangle \in Q \times Q'$$

B. From SteC to TA: Induction on the Structure of STeC Specification

After the essential operators are defined, we are trying to define a mapping from STeC to TA. We need to define a morphism from the set of language STeC to the set of TA inductively.

Definition IV.4. *We denote the set of all processes(formulas) of STeC as:*

$$\mathbb{P}_{STeC} = \{\ P \mid \ P \text{ is a process in STeC}\}$$

Definition IV.5. *We denote the set of all TAs as:*

$$\mathbb{M}_{TA} = \{\ A \mid A \text{ is a Timed Automata}\}$$

Definition IV.6. *Knowing that $\mathbb{P}_{STeC}$ and $\mathbb{M}_{TA}$ are the sets of processes of STeC and TAs respectively. We define a morphism $\mathcal{F} : \mathbb{P}_{STeC} \to \mathbb{M}_{TA}$ which can be inductively built according to the following rules:*

i

$$\mathcal{F}(\mathbf{Send}_{(l,t)}^{G \to G'}(m)) = A =$$

$$(\{q_0, q_1\}, \{m!\}, \{x, y\}, q_0, E, I, \{l\}, L, \{q_1\}).$$

where we know that A is a TA. $\langle q_0, m!, \emptyset, \{y\}, q_1\rangle$ is the only element in E. $I = \emptyset$. $AP = \{l\}$ and L is a labelling function which satisfies(Fig. 2 shows the graph of the TA):

$$L(q) = \begin{cases} l & \text{if } q \in \{q_0, q_1\} \\ \emptyset & \text{otherwise} \end{cases}$$

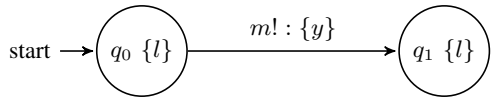

Fig. 2. The TA of $\mathbf{Send}^{G \rightarrow G'}_{(l,t)}(m)$

ii

$$\mathcal{F}(\mathbf{Get}^{G \leftarrow G'}_{(l,t)}(m)) =$$

$$(\{q_0, q_1\}, \{m?\}, \{x, y\}, q_0, E, I, \{l\}, L, \{q_1\})$$

where $\langle q_0, m?, \emptyset, \{y\}, q_1\rangle$ is the only element in E. $I = \emptyset$. $AP = \{l\}$ and L is a labelling function which satisfies:

$$L(q) = \begin{cases} l & \text{if } q \in \{q_0, q_1\} \\ \emptyset & \text{otherwise} \end{cases}$$

Fig. 3 shows the TA of $\mathbf{Get}^{G \leftarrow G'}_{(l,t)}(m)$.

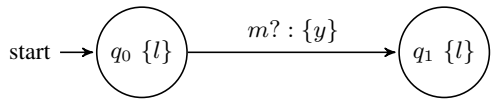

Fig. 3. The TA of $\mathbf{Get}^{G \leftarrow G'}_{(l,t)}(m)$

iii

$$\mathcal{F}(\alpha^G_{(l,t)}(l', \delta)) =$$

$$(\{q_0, q_1\}, \{\alpha\}, \{x, y\}, q_0, E, I, \{l, l'\}, L, \{q_1\})$$

.

where $\langle q_0, \alpha, y = \delta, \{y\}, q_1\rangle$ is the only element in E. $I(q_0) = \{y \leq \delta\}$. $AP = \{l, l'\}$, L is a labelling function which satisfies:

$$L(q) = \begin{cases} l & \text{if } q = q_0 \\ l' & \text{if } q = q_1 \\ \emptyset & \text{otherwise} \end{cases}$$

Fig. 4 shows the TA of atomic commmand $\alpha^G_{(l,t)}(l', \delta)$.

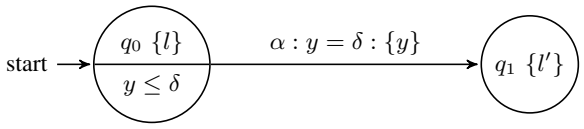

Fig. 4. The TA of $\alpha^G_{(l,t)}(l', \delta)$

Similarly, we have the TA for $\beta^G_{(l,t)}(\delta)(a, u, \sigma)$:

iv $\mathcal{F}(\beta^G_{(l,t)}(\delta)) = (\{q_0\}, \emptyset, \{x, y\}, q_0, E, I, \{l\}, L, \{q_0\})$. *The TA has only 1 state—q_0. $E = \emptyset$ and $I(q_0) = \{y \leq \delta\}$. $L(q_0) = l$ (see Fig. 5).*

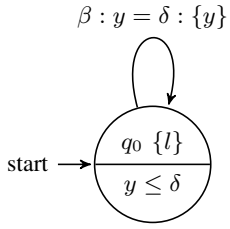

Fig. 5. The TA of $\beta^G_{(l,t)}(\delta)$

v For all $R \in \mathbb{P}_{STeC}$, if $R = P\ ;\ Q$, then

$$\mathcal{F}(R) = \mathcal{F}(P; Q) = \mathcal{F}(Q) \diamond_{STeC} \mathcal{F}(P) = A_Q \diamond_{STeC} A_P$$

if and only if $\mathcal{F}(P) = A_P$ and $\mathcal{F}(Q) = A_Q$ hold. It is according to the of $\mathcal{F}$'s preservation of sequence operator in $\mathbb{P}_{STeC}$.

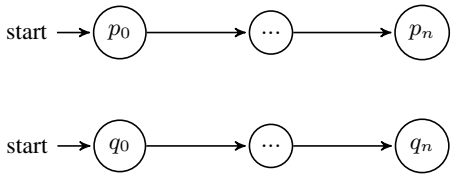

Fig. 6. The TA of P and Q

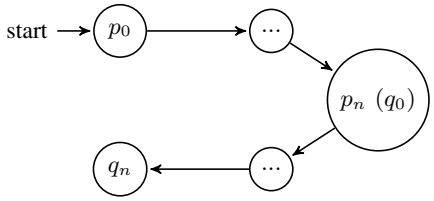

Fig. 7. The TA of $P; Q$

vi For all $R \in \mathbb{P}_{STeC}$, if $R = P$ [] Q, then

$$\mathcal{F}(R) = \mathcal{F}(P \text{ [] } Q) = \mathcal{F}(Q) \oplus_{STeC} \mathcal{F}(P)$$
$$= A_Q \oplus_{STeC} A_P$$

if and only if $\mathcal{F}(P) = A_P$ and $\mathcal{F}(Q) = A_Q$ hold. In other words, $\mathcal{F}$ preserve choise operator in $\mathbb{P}_{STeC}$.

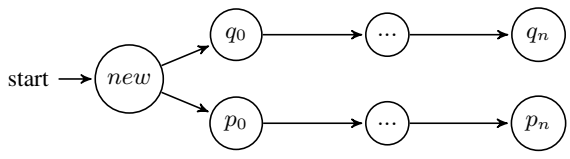

Fig. 8. The TA of $P[]Q$

vii For all $R \in \mathbb{P}_{STeC}$, if $R = P \parallel Q$, then

$$\mathcal{F}(R) = \mathcal{F}(P \parallel Q) = \mathcal{F}(Q) \otimes_{STeC} \mathcal{F}(P)$$
$$= A_Q \otimes_{STeC} A_P$$

if and only if $\mathcal{F}(P) = A_P$ and $\mathcal{F}(Q) = A_Q$ hold. $\mathcal{F}$ preserve parallel operator in $\mathbb{P}_{STeC}$.

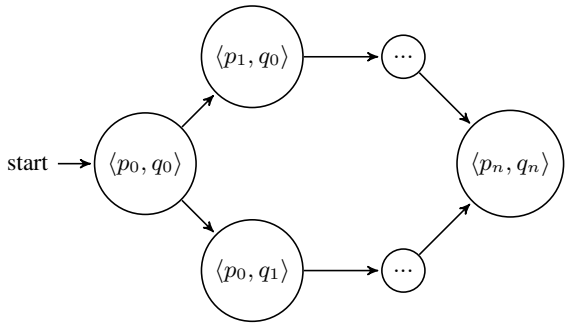

Fig. 9. The TA of $P \parallel Q$

viii

$$\mathcal{F}(B \to P) = \begin{cases} \mathcal{F}(P) & \text{if } \mathcal{T}(B) = 1 \\ \emptyset & \text{otherwise} \end{cases}$$

ix For any process $R = P \trianglerighteq_\delta Q$, we consider to convert it into a form which contains only the operator "[]" and ";".
Let $P = P_i \ ; \ P'$ where P_i always satisfies:

$$u \le \delta \wedge u' > \delta$$

for any transition $\langle P_i, (a, u, \sigma)\rangle \to \langle P_i', (a', u', \sigma')\rangle$, then

$$R = P \trianglerighteq_\delta Q = []_{i \in X} P_i; Q$$

According to this, we have:

$$\mathcal{F}(P \trianglerighteq_\delta Q) = \mathcal{F}([]_{i \in X} P_i; Q) = \oplus_{i \in X}(\mathcal{F}(P_i; Q)) =$$
$$\oplus_{i \in X}(A_{P_i} \diamond_{STeC} A_Q)$$

Theorem IV.1. *Let $\mathcal{F} : \mathbb{P}_{STeC} \to \mathbb{M}_{TA}$ be a morphism descriped in Definition IV.6, then*

$$\mathbf{dom}(\mathcal{F}) = \mathbb{P}_{STeC}$$

Proof: It is obvious according to the definition of $\mathcal{F}$. ■

C. One Example—Railroad Crossing Problem

Here is a classical example to show that how a STeC model of system can be transformed into TA. This example is descriped by STeC, we don't explain in details how this system can be modeled by STeC, for more information, refer to [6].

Imagine a railroad crossing with several train tracks and a common gate(here we only consider there is only one track), such as the one depicted in Fig. 10. In order to avoid collision, the gate must be closed when a train passes the crossing point. This is a very classical problem for comparing various specification and modeling language. It certainly can be modeled by TA. Here, we use STeC to model it and then transform it into TA using the morphism introduced in Definition IV.6.

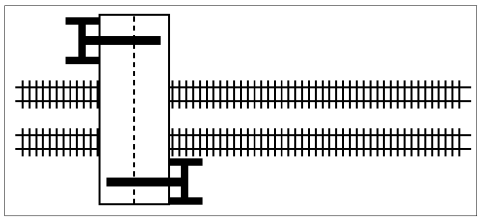

Fig. 10. Railroad Crossing Problem

1) Train Agent: The system model descripted by STeC is as follows. Let P_{Train} and P_{Gate} be the process of train agent and gate agent, A_{Train} and A_{Gate} be the corresponding TA. Then we have(for more details about how to model this example using STeC, refer to [6]):

$$P_{Train} = Run(\infty); (Send_{(Lapp,t,G)}(Appr) \parallel Approach_{(Laap,t)}(\Gamma))\triangleright$$

$$\left\{ \begin{array}{l} Get_{(Lpass,t+\Gamma,G)}(Cross) \to (Pass_{(Lpass,t+\Gamma)}(\triangle); \\ \quad (Send_{(Lleav,t+\Gamma+\triangle,G)}(Lleav) \parallel P_{Train})) \\ [] \\ Get_{(Lpass,t+\Gamma,G)}(Noncross) \to (Stop_{(Lpass,t+\Gamma)}(\Upsilon); \\ Wait_{(Lstop,t+\Gamma+\Upsilon,G)}(Cross); Pass_{(Lstop,t\prime+\Omega,G)}(\Omega); \\ \quad (Send_{(Lleav,t\prime+\Omega,G)}(Lleav) \parallel P_{Train})) \end{array} \right\}$$

Easy to see that P_{Train} is a recursive process. The train is firstly approaching the crossing road, it send a message to the gate, informing that there is a train coming. After Γ time it tries to get a message from the gate. If it gets the "Cross" message, then the gate is now closed and it is safe for train to pass the crossing road. After the train passed, it send a "Lleav" message to inform the gate and continuely run again. If it gets the "NonCross" message, then the train will stop and wait for an inform by gate to pass again.

Applying Definition IV.6, we can use the function $\mathcal{F}(P_{Train})$ to transform the model P_{Train} into a TA inductively as follows:

$$\mathcal{F}(P_{Train}) = \mathcal{F}(Run(\infty); P'_{Train})$$
$$= \mathcal{F}(Run(\infty)) \diamond_{STeC} \mathcal{F}(P'_{Train})$$
$$= A_{Run(\infty)} \diamond_{STeC} \mathcal{F}(\ (Send_{(Lapp,t,G)}(Appr) \parallel Approach_{(Laap,t)}(\Gamma))\triangleright$$

$$\left\{ \begin{array}{l} Get_{(Lpass,t+\Gamma,G)}(Cross) \to (Pass_{(Lpass,t+\Gamma)}(\triangle); \\ \quad (Send_{(Lleav,t+\Gamma+\triangle,G)}(Lleav) \parallel P_{Train})) \\ [] \\ Get_{(Lpass,t+\Gamma,G)}(Noncross) \to (Stop_{(Lpass,t+\Gamma)}(\Upsilon); \\ Wait_{(Lstop,t+\Gamma+\Upsilon,G)}(Cross); Pass_{(Lstop,t\prime+\Omega,G)}(\Omega); \\ \quad (Send_{(Lleav,t\prime+\Omega,G)}(Lleav) \parallel P_{Train})) \end{array} \right\}$$

$$)$$

$$= A_{Run(\infty)} \diamond_{STeC} \mathcal{F}(P_1 \triangleright P_2) = A_{Run(\infty)} \diamond_{STeC} \mathcal{F}(P_1; P_2)$$

$$= A_{Run(\infty)} \diamond_{STeC} \mathcal{F}(P_1) \diamond_{STeC} \mathcal{F}(P_2) = ... = A_{Train}.$$

where $P_1 = Send_{(Lapp,t,G)}(Appr) \parallel Approach_{(Laap,t)}(\Gamma)$ and

$$P_2 = \begin{cases} Get_{(Lpass,t+\Gamma,G)}(Cross) \rightarrow (Pass_{(Lpass,t+\Gamma)}(\triangle); \\ \quad (Send_{(Lleav,t+\Gamma+\triangle,G)}(Lleav) \parallel P_{Train})) \\ [] \\ Get_{(Lpass,t+\Gamma,G)}(Noncross) \rightarrow (Stop_{(Lpass,t+\Gamma)}(\Upsilon); \\ Wait_{(Lstop,t+\Gamma+\Upsilon,G)}(Cross); Pass_{(Lstop,t\prime+\Omega,G)}(\Omega); \\ \quad (Send_{(Lleav,t\prime+\Omega,G)}(Lleav) \parallel P_{Train})) \end{cases}$$

Finally we can get the A_{Train} as the TA of the process P_{Train}. The result of A_{Train} is shown in the following figure(Fig. 11).

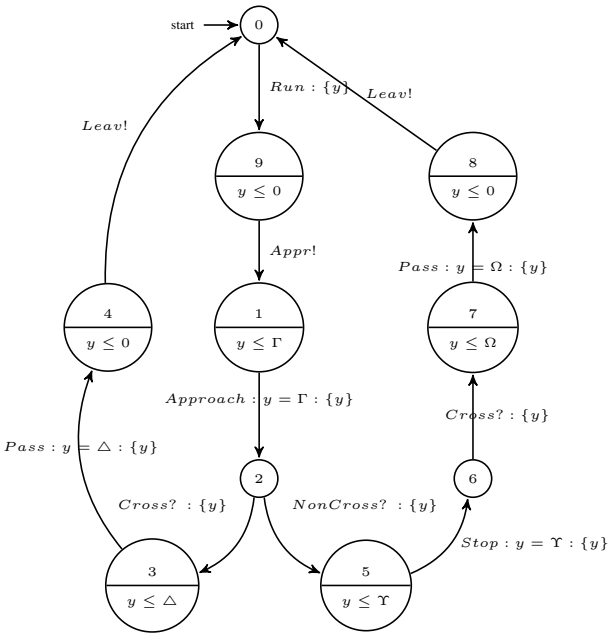

Fig. 11. The TA of Train Agent

Set $A_{Train} = (Q, \Sigma, \mathcal{C}, q_0, E, I, AP, L, F)$ be the corresponding timed-automata translated from STeC, the interpretations of each part of the tuple are listed as follows, from which we can see the difference in the way of modeling behaviors of a system by STeC and TA:

- $Q = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ is the set of state of TA.
- $\Sigma = \{Run, Appr!, Cross?, Approach, Pass, Noncross?, Stop, Cross?\}$ corresponds to all the actions in STeC.
- $\mathcal{C} = \{x, y\}$. Two clocks, one for recording total time from the beginning and one for recording the delay of actions in STeC.
- $E \subseteq Q \times \Sigma \times G(\mathcal{C}) \times 2^{\mathcal{C}} \times 2^{Q}$ is a transition relation(as shown above). In timed-automata, the delay of time only happens in a state, action is executed instantaneously when the guard is satisfied. For example, transition which transit from state 1 to state 2 can be denoted as the tuple—$\langle 1, Approach, \{y = \Gamma\}, \{y\}, \{2\}\rangle$. Remember that $G(\mathcal{C})$ is the set of constraints under the set of clocks $\mathcal{C}$, a clock constraint can be defined as a BNF form:

$$g := x \bowtie c \mid g \wedge g$$

where $g \in \mathcal{C}$, $c \in \mathbb{N}$ and $\bowtie \in \{<, \leq, >, \geq\}$. $2^{\mathcal{C}}$ represents the clocks that should be set to 0 once after transition happened(here,in this example,clock y should be reset).
- $q_0 = 0$ is the initial state.
- $I : Q \rightarrow G(\mathcal{C})$ is an invariant-assignment function. It maps each state(or location) to a subset of $G(\mathcal{C})$. It specifies that how long the timed-automata can stay in one state. For example,in our A_{Train},we can only stay at most 5 units of time and after that, any transition which satisfies the guard condition must happen, or(if no such transitions exists) no further progress is possible. In some states, one transition must happen right after the state is reached, for example, state 4 and state 8. Since the invariant is $y \leq 0$, so the action $Send$ is executed immediately which mimic the behavior of the actions that take no time in STeC language.
- $L : Q \rightarrow 2^{AP}$ is a labelling function which relates a set $L(s) \in 2^{AP}$ of atomic propositions to any state s. In this example, the set of atomic propositions represents the set of locations in STeC, we have $L(4) = \{Lleav\}$,$L(1) = \{Lapp\}$,$L(2) = \{Lpass\}$,$L(0) = \emptyset$, etc.

2) Gate Agent: Follow the same procedure, we can get the A_{Gate} from P_{Gate}.

$$P_{Gate} = (Get_{(Open,t,G)}(Appr) \rightarrow Closing_{(Open,t+\theta_0)}(\Pi));$$

$$\left\{ \begin{array}{l} (Closed_{(Closed,t+\theta_0+\Pi)}(1) \rightarrow \\ (Send_{(Closed,t+\theta_1,G)}(Cross) \parallel (Closed_{(Closed,t+\theta_1)}(\infty) \\ \triangleright (Get_{(Closed,t+\theta_2)}(Leav) \\ \rightarrow Opening_{(Closed,t+\theta_2)}(\zeta)))); P_{Gate} \\ [] \\ (Unclosed_{(Unclosed,t+\theta_0+\Pi)}(0) \rightarrow \\ (Send_{(Unclosed,t+\theta_0+\Pi,G)}(NonCross) \parallel \\ Clossing_{(Unclosed,t+\theta_0+\Pi)}(\pi)) \\ ; (Closed_{(Closed,t+\theta 3)}(0) \parallel (Send_{(Closed,t+\theta_3,G)}(\\ Cross)) \triangleright (Get_{(Closed,t\prime+\Omega,G)}(Leav) \\ \rightarrow Opening_{(Closed,t\prime+\theta_4)}(\zeta)); P_{Gate} \end{array} \right\}$$

We can get $\mathcal{F}(P_{Gate}) = A_{Gate}$ which is shown as follows:

3) Railroad Crossing System: The corresponding TA of railroad crossing system is:

$$A_{Sys} = \mathcal{F}(P_{Sys}) = \mathcal{F}(P_{Train} \parallel P_{Gate})$$

$$= A_{Train} \otimes_{STeC} A_{Gate}.$$

D. Summary of Results

In this section, in order to encode STeC into TA, we define 3 operations on TAs(Definition IV.1, IV.2 and IV.3), corresponding to the three binary operators in STeC— sequence, choice and parallel. Definition IV.4 and IV.5 define a two set representing the set of language of STeC and TA. The main

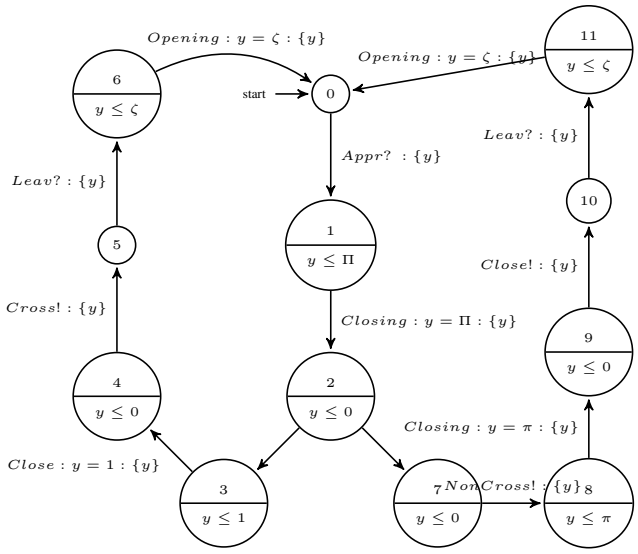

Fig. 12. the TA of Gate Agent

result is Definition IV.6, where a transformation from STeC to TA is inductively built on the structure of STeC specification. At last, an example is given to explain the transformation strategy.

V. Conclusion

In this paper, we introduce a spatio-temporal modeling language—STeC and propose a possible verification framework of STeC-models. We provide a support for verification by encoding STeC into TA. We define the transformation in a theoretical way and show how to transform through an example.

This paper does not give more details about the verification on STeC-models, and how to define CCSL as a specification language of STeC-models. In the future, more works are needed to give the details of that.

Acknowledgment

This work is supported by the INRIA Associated Team DAESD between INRIA and ECNU, NSFC (No. 61021004 and No. 61370100) and Shanghai Knowledge Service Platform Project (No. ZF1213).

References

[1] C.A.R Hoare. *Communicating Sequential Processes*. Communications of ACM, Volume 21 No. 8, 1978.

[2] Robin Milner. *A Calculus of Communicating Systems* . Computer Science, Vol.92, 1986.

[3] Steve Schneider . *An Operational Semantics for Timed CSP* . Information and Computation, 116:193-213, 1995.

[4] R. Alur and D. L. Dill . *A theory of timed automata* . Theoretical Computer Science, 126(2):183-235, 1994.

[5] Regis Gascon, Frederic Mallet, Julien DeAntoni. *Logical time and temporal logics: comparing UML MARTE/CCSL and PSL.* Temporal Representation and Reasoning (TIME), pp 141-148, 2011.

[6] Yixiang Chen. *STeC: A Location-Triggered Specification Language For Real-Time Systems*. Science China, Vol.53, No.1-18, 2010.

[7] Hengyang Wu, Yixiang Chen and Min Zhang *On Denotational Semantics of Spatio-Temporal Consistency Language STeC*. International Symposium on Theoretical Aspects of Software Engineering, pages 113-120, 2013.

[8] Stefano Cattani and Marta Kwiatkowska. *A Refinement-based Process Algebra for Timed Automata*. Under consideration for publication in Formal Aspects of Computing, 17(2), pages 138-159, Springer. August 2005.

[9] Parosh Aziz Abdulla, Pavek Krcal and Yi Wang. *Sampled semantics of times automata*. Logical Methods in Computer Science, Vol. 6 (3:14): 1-37, 2010.

[10] Charles Andre, Frederic Mallet. *Clock Constraints in UML/MARTE CCSL*. Research Report of Research Unit of INRIA Sophia Antipolis, 2008.

[11] Charles Andre. *Syntax and Semantics of the Clock Constraint Specification Language(CCSL)*. Research Report of Research Unit of INRIA Sophia Antipolis, 2009.

[12] Jagadish Suryadevara, Cristina Seceleanu, Frederic Mallet and Paul Pettersson. *Verifying MARTE/CCSL Mode Behaviors using UPPAAL*. 11th International Conference on Software Engineering and Formal Methods, Volume 8137, pp 1-15, 2013.

[13] Frederic Mallet, Jean-Vivien Millo, Yuliia Romenska. *State-based representation of CCSL operators*. Research Report of Project-Team Aoste of INRIA Sophia Antipolis, 2013.

[14] Frederic Mallet. *Logical Time in Model-Driven Engineering*. University of Sophia Antipolis, 2010.

[15] Iryna Zaretska, Galyna Zholtkevych, Grygoriy Zholtkevych. *Clocks Model for Specification and Analysis of Timing in Real-Time Embedded Systems*.

[16] Rajeev Alur, David Dill. *Automata-theoretic Verification of Real-time Systems*. 1995.

[17] Joel Ouaknine,Steve Schneider. *Timed CSP:A Retrospective*. Electronic Notes in Theoretical Computer Science, 162 (2006) 273C276.

[18] A.W.Roscoe. *A CSP solution to the "trains" problem*. Programming Research Group, University of Oxford.

[19] Volker Diekert, Paul Gastin. *First-order definable languages*. Logic and Automata: History and Perspectives, Texts in Logic and Games, 2008.